WhaTap Monitoring

通知のアラートから アクションのアラートへ

WhaTap Monitoringアラート機能の活用実践ガイド

～ Zabbixユーザー向けの『インテリジェントアラート運用』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、IT現場でエンジニアの皆様が直面しがちな4つのアラート体制の課題を取り上げ、WhaTap Monitoringのアラート機能がどのようにこれらを補完/高度化するかを、実際の設定手順を交えてご説明します。

章	テーマ	解決する課題
第1章	5分で始めるメトリクス監視アラート	初期設定の複雑さとしきい値設定の属人化
第2章	アラートノイズを抑止するインテリジェントイベント	アラート疲れと誤検知の常態化
第3章	チーム別アラート振り分けとAIイベント分析	通知の一極集中と原因特定の遅延
第4章	主なアラート機能との比較	導入判断の材料

ZabbixとWhaTapの役割分担（棲み分け）

本書の冒頭で、まず重要な前提を明確にしておきます。**WhaTapは、既存のZabbixを置き換えるだけのツールではありません。**Zabbixの強みを活かしながら、Zabbixの基本機能では構築、維持に膨大な工数がかかる領域を、SaaSサービスとして補完/高度化することもWhaTapの役割です。

Zabbixが得意な領域（既存資産として継続活用）	WhaTapが補完・高度化する領域（上位レイヤー）
・インフラ、ハードウェアの監視 ・ネットワーク機器の監視 ・OS基本リソース（CPU/Memory/Disk）の監視 ・オンプレ環境を含むエージェント分散監視 **On-premise**	・アプリケーション層（APM）の詳細監視 ・複合条件、MXQLによる多次元アラート ・AI異常検知、AIによる原因分析 ・チーム別ルーティング、通知品質の最適化 ・SaaS型による運用保守工数の削減 **SaaS/Cloud**

そのため、本書では**「Zabbixからの移行」ではなく「Zabbixと併用したアラート運用の高度化」**を主な内容として構成しています。

はじめに：アラートは「鳴らすだけで終わり」ではありません

多くのITエンジニアの皆様は、Zabbixのトリガー機能を基盤として長年にわたりアラート体制を構築してこられました。「CPU使用率が80%を超えたらメール通知」「ディスク使用率が90%に達したらSlack通知」―こうした基本的なしきい値アラートは、あらゆる監視体制の出発点です。

しかしながら、運用規模の拡大とシステムの複雑化に伴い、現場では以下のような声をしばしば耳にします。

> 「Zabbixからの大量アラートでメールが溢れている。アラートをExcelに転記してチケットのように管理しているが、運用工数がかかりすぎる。」

> 「夜間バッチ中にネットワーク瞬断が起きるたびに、同じトリガーから障害 → 復旧、障害 → 復旧…と通知が繰り返し届く。朝になって確認すると、バッチも正常に終わったし、システムも復旧されていた。」

> 「アラートメールの件名と本文がデフォルトのままで、何が起きているのか把握しづらい。結局、毎回Zabbixの画面にログインして、トリガーの詳細とグラフを自分で確認するしかない。」

> 「プロジェクトに参加しているメンバー全員に全アラートが届くため、自分に関係のないアラートが日常的に流れてきて、チーム全体がアラートを無視する習慣がついてしまった。」

これらの課題は「Zabbixの設定ミス」や運用不足によるものではありません。Zabbixのアラートモデルは**「単一メトリクスのしきい値超過をトリガーで判定し、アクションで即時通知する」**という、シンプルかつ確実な1対1の構造に最適化されています。

しかし、現代の複雑化したシステム運用においては、インフラだけでなくアプリケーション層までを含めた**「複数の指標を時間軸で相関分析し、本質的な障害のみを検出し、適切な担当者に知らせる」**という多層的なアラート設計が求められるようになっています。つまり、既存の監視基盤を活かしつつ、いかにアラート運用をインテリジェント化していくかが重要です。

第1章：5分で始めるメトリクス監視アラート

～ テンプレートとシミュレーションで属人化を解消する ～

1.1 現場で遭遇する一般的なシナリオ

状況：新規プロジェクトのJavaアプリケーションサーバーを本番環境にデプロイした翌日。運用リーダーから「基本的なアラートを設定しておいて！」と指示を受けました。

ZabbixではOS監視向けの標準テンプレートを使えば基本的なリソース監視はすぐに開始できますが、本番運用に必要な以下の項目までを含めると、相応の設定工数が発生します。

- アプリケーション固有メトリクス向けのアイテム → トリガー作成
- 各メトリクスのしきい値を対象サービス特性に合わせて個別に調整
- アクション設定でSlack/Teamsへの通知先を組み込み（Webhookスクリプトの作成とテスト）
- エスカレーション設定（Warning → Critical の段階設定）
- 通知メールの件名/本文をマクロでカスタマイズ
- テスト通知の送信と検証

経験豊富なエンジニアでも、このプロセス全体を網羅すると相当な構築/検証時間を要します。しかも、しきい値の「適切な値」は対象サービスの特性に依存するため、経験の浅い担当者が設定すると誤検知（False Positive）や検知漏れを頻発させるリスクがあります。

1.2 アラート設定における構造的課題

Zabbixのアラートモデルは、アイテム（データ収集）→ トリガー（条件判定）→ アクション（通知実行）という3層構造で設計されています。この設計は柔軟性に富む一方で、以下の構造的課題を抱えています。

- アイテムとトリガーが分離しているため、「このメトリクスでアラートを出したい」という単純な要望に対しても、最低2つの設定オブジェクトを作成、関連付ける必要がある
- トリガー式が独自の関数構文（例：{host:cpu.util.avg(5m)}>80）を使用するため、SQLやプログラミングに慣れていないエンジニアには学習コストが発生する
- アクションの条件フィルタリング（特定のホストグループのCriticalのみSlackへ送信する等）の設定が複雑で、意図しない通知漏れが発生しやすい
- 過去の時系列データに照らし合わせて発報回数を予測するシミュレーション機能がなく、Expression testingによる条件式の静的なテストにとどまる。本番環境で実際に発報するまで「この設定が妥当なノイズ量に収まるか」を検証できない

1.3 メトリクスイベントの設定

WhaTap Monitoringでは、エージェントをインストールするだけで、基本的なアラートテンプレートが自動的にセットアップされます。CPU使用率、メモリ使用率、ディスクI/O、レスポンスタイムといった基本メトリクスのアラートルールが、業界標準のしきい値でプリセットされた状態で用意されています。初期状態ではOFF（無効）になっているため、運用チームが対象環境に応じて必要なルールをONにするだけで即時にアラートが開始します。しきい値を一から設計する必要はありません。さらに、カスタムイベントの追加はすべてWeb画面上の操作で完結し、シミュレーション機能を使って設定前に過去データで動作結果をプレビューできます。

📌 WhaTapが提供するメトリクスイベントのメリット

- エージェントインストール時点で基本アラートテンプレートが自動セットアップ（必要なルールをONにするだけ）
- Web画面上でメトリクス選択 → 条件選択 → 保存の3ステップで完了（5分以内）
- シミュレーション機能で「この設定なら過去1週間で何回発報するか」を時系列データに基づき事前検証
- テンプレートの共有、一括インポートで、プロジェクト間の設定統一が容易

🛠️ 実践ガイド① ：基本的アラートを5分で設定する

ステップ1：アラート受信設定の確認

設定メニューへのアクセス：アラート通知 > 通知設定

WhaTapでは、プロジェクトにメンバーを追加した時点で、自動的にそのメンバーに対するメールでのアラート受信が有効化されます。特別な設定なしに、プロジェクトメンバー全員がアラートをメールで受信できる状態になっています。

Slack、Teamsなどの3rd Partyと連携する場合

1. イベント受信設定画面の3rd Partyプラグイン機能から追加ボタンをクリックします。
2. 連携したいサービス（Slack、Teams、Telegram、Line、Pagerduty等）を選択します。
3. 画面の案内に従って連携設定を完了します（例：SlackはIncoming Webhook URL、TeamsはWebhook URL等を入力）。

WhaTapはEmail通知やモバイルアプリ通知の他にもSlack、Teams、Telegram、Line、Pagerduty、Opsjenie、ilert等に標準対応しています。その他の社内メッセンジャーや運用管理ツールとも標準WebhookまたはWebhook JSON形式で連携可能です。

ステップ2：メトリクスイベントの追加

設定メニューへのアクセス：アラート通知 > イベント設定 > メトリクス

1. ＋アラート通知追加ボタンをクリックします。
2. 各項目をドロップダウンおよび選択肢から設定します。

設定項目	設定方法	設定例
イベント名	テキスト入力	CPU使用率のしきい値超過
カテゴリ	ドロップダウンから選択	server_cpu
フィールド	ドロップダウンから選択	cpu
アラート条件	演算子をドロップダウンで選択し、しきい値を入力	>= 80
持続時間	ドロップダウンから選択	1分（短時間のスパイクを除外）
レベル	ドロップダウンから選択	Warning

3. シミュレーションボタンをクリックして、過去データに基づく発報量予測を確認します。
4. 保存ボタンをクリックします。

メトリクスの名称や条件値を手動で入力する必要はなく、あらかじめ用意されたカテゴリ、フィールド一覧から選択するだけで設定が完了します。

ステップ3：テスト通知の確認

設定が完了すると、WhaTapは指定したしきい値を超過した時点でリアルタイムに通知を送信します。リアルタイムアラート画面（画面右上のベルアイコン）で、発報されたイベントのタイトル、発生時刻、メッセージを直ちに確認できます。

1.4 基本的アラート設定テンプレート

以下は、Javaアプリケーションサーバーにおいて最初に設定すべきアラートの推奨テンプレートです。

イベント名	メトリクス	条件	持続時間	レベル
CPU使用率の超過	cpu_usage	> 80%	1分	Warning
CPU使用率の危険	cpu_usage	> 95%	30秒	Critical
ヒープメモリ使用率の超過	heap_used / heap_max	> 85%	2分	Warning
レスポンスタイム異常	resp_time	> 3000ms	1分	Warning
エラー率の上昇	error_count / tx_count	> 5%	3分	Critical
エージェント未応答	agent_status	非アクティブ	1分	Critical

これらはすべて**Web画面上で5分以内に設定可能**です。Zabbixでも標準テンプレートを使えばリソース向けの基本的監視は短時間で開始できますが、Slack/Teamsへのスクリプト組込み、深刻度に応じたエスカレーションのアクション設計、通知文面のカスタマイズまでを含めると、設定と検証に相当な時間を要します。

第2章：アラートノイズを抑止するインテリジェントイベント

～ 複合メトリクスイベント、ヒットマップパターン、異常検知を活用する ～

本章の位置づけ：別文書の「WhaTap Log Monitoring活用実践ガイド（第2章）」では、「複合ログイベント」を活用したアラートノイズ制御について解説しました。本章では、メトリクス領域における同様の機能―**複合メトリクスイベント、ヒットマップパターン検知、AI異常検知**―について解説します。

2.1 しきい値アラートが引き起こす「アラート疲れ」

一般的な監視ツールにおけるアラート設定の多くは、「メトリクスXがしきい値Yを超えたら通知」という1対1の単一条件です。しかし昨今のシステム障害は、単一メトリクスの異常だけではその詳細をキャッチできないケースが増えています。

例：ECサイトの決済処理遅延

- CPU使用率：45%（正常範囲）
- メモリ使用率：62%（正常範囲）
- DBコネクションプール：使用率98%（しきい値超過！）
- トランザクション応答時間：平均5,200ms（通常の10倍）

この場合、CPUとメモリに対するしきい値アラートは発報されず、DBコネクションプールのしきい値アラートだけが発報されます。しかし、この場合の問題の本質は「DBコネクション枯渇によりトランザクションが滞留し、エンドユーザーが決済完了を待ち続けている」という複合的な障害です。

Zabbixの標準機能でこの複合条件を検知するには、カスタムトリガー式のネスト({A} and {B} and {C})や相関トリガーの設計が必要で、設定とメンテナンスの工数が大きくなる傾向があります。

2.2 3層のインテリジェントアラート

WhaTapは、単純なメトリクスしきい値アラートに加えて、以下の3種類のインテリジェン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
複合メトリクスイベント	複数メトリクスの時間軸相関性、MXQL条件	「CPU急増＋レスポンス遅延＋エラー率上昇」が同時発生した場合のみ通知
ヒットマップパターン検知	トランザクション分布の形状変化	「ヒットマップに縦線パターン（一斉遅延）が出現」した場合を自動検知
AI異常検知 (Anomaly Detection)	機械学習(ML)による正常パターンからのはずれ	「しきい値では定義できない普段と違う振る舞い」を自動検出

これらの機能はすべてコーディング不要のWeb画面での設定で利用できます。Zabbixでは設定とメンテナンスに膨大な工数がかかるような多次元的、動的な障害検知をWhaTapでは手軽に実現します。

🛠️ 実践ガイド② ：複合メトリクスイベントを設定する

設定メニューへのアクセス：アラート通知 > イベント設定 > 複合指標 > ＋アラート通知追加

シナリオ：DBコネクション枯渇によるトランザクション遅延の検知

ここでは、WhaTap Java APMが収集する実メトリクスを用いて、「DBコネクション取得待ちのトランザクションが滞留し、かつ全体の応答時間が悪化している」状態を検知します。使用するメトリクスは以下の2つです。

使用メトリクス	カテゴリ	フィールド	意味
DBC段階の滞留トランザクション数	app_active_stat	dbc	DBコネクションプールから接続を取得しようとしている（取得待ちを含む）トランザクション件数。プール枯渇時にここが急増する
トランザクション平均応答時間	app_counter	resp_time	全トランザクションの平均応答時間（ミリ秒）

メトリクス選定の根拠：WhaTap Java APMでは「DB接続プールの使用率(%)」を直接表す単一フィールドはありません。代わりに、APMエージェントがトランザクションの実行ステップを METHOD / SQL / HTTPC / DBC / SOCKET の5段階で常時計測しており、**app_active_stat.dbcが増加する＝コネクション取得待ちでトランザクションが詰まっている**という現場の事象を最も正確に表す指標になります。

複合メトリクスイベントの動作フロー

複合メトリクスイベントは、以下の3ステップで動作します。

※ MXQL：WhaTap Monitoringが収集、保存しているデータを自由に照会できるクエリー
※ 単一カテゴリのメトリクスであれば「メトリクスイベント」設定画面でマウス操作だけで完結します。但し、異なるカテゴリにまたがる2つ以上の指標を結合して条件を組む場合は、MXQLによる記述が必須です。本書の「DBコネクション×応答時間」の事例も、app_active_stat と app_counter という2つの異なるカテゴリを跨ぐため、MXQLを用います。

1) MXQLを自動生成する

MXQLは強力ですが、文法を一から覚えるのは負担です。WhaTap AIにメトリクス名とやりたいことを指示するだけで、すぐに使えるMXQLを生成してもらえます。

WhaTap AIの利用例：

> **質問：** DBコネクション待ち件数 (app_active_stat.dbc) とトランザクションの応答時間 (app_counter.resp_time) の指標を、同時にSELECTできるMXQLを作成してください。

WhaTap AIの回答例： app_active_stat.dbc（DBコネクション待ち件数）と app_counter.resp_time（応答時間）を oname × 時間軸でJOINします。

```
CATEGORY app_counter
TAGLOAD
GROUP { timeunit: "1m", pk: [oname], merge: [resp_time] }
UPDATE { key: [resp_time], value: avg }
RENAME { src: resp_time, dst: avg_resp_time_ms }

SUB dbc_stat
  CATEGORY app_active_stat
  TAGLOAD
  GROUP { timeunit: "1m", pk: [oname], merge: [dbc] }
  UPDATE { key: [dbc], value: avg }
  RENAME { src: dbc, dst: dbc_wait_count }
END

JoinByTime { pk: [oname], query: dbc_stat, timeunit: "1m" }

SELECT [time, oname, avg_resp_time_ms, dbc_wait_count]
```

このMXQLは次の処理を行っています。

区分	MXQL	意味
メイン処理	CATEGORY app_counter + TAGLOAD	app_counter カテゴリのデータをロード
	GROUP {...} + UPDATE {...}	エージェント名×1分単位で resp_time を平均集計
	RENAME	集計結果を avg_resp_time_ms という名前に変更
サブ処理	SUB dbc_stat ... END	サブクエリとして app_active_stat のデータをロードし、dbc_wait_count という名前で1分平均を集計
結合	JoinByTime {...}	メイン処理とサブ処理を oname ×時間軸で結合
出力	SELECT [...]	最終的に時間、エージェント名、応答時間、DBコネクション待ち件数の4列を抽出

2) MXQLでデータを照会する

1. 複合メトリクスイベント設定画面のイベントデータ照会テキスト欄に、生成したMXQLを貼り付けます。
2. 照会ボタンをクリックします。
3. 過去データに基づいて、time / oname / avg_resp_time_ms / dbc_wait_count の4列のテーブルが表示されることを確認します。

ここで、意図したデータが出力されない場合は、MXQLを修正する必要があります。WhaTap AIに「このMXQLで oname ごとに2つの指標が同時に出るようにしてほしい」など追加の指示を出して修正を行います。

3) アラート条件を入力する

MXQL照会で取得した各列名（avg_resp_time_ms、dbc_wait_count）を使って、設定画面左側の条件式欄にアラート発報の条件を入力します。

⇒「1分平均応答時間が3,000ms（3秒）を超え、かつDBコネクション待ち件数の5分平均が、5件を超えた場合にアラートを発生する」

```
イベント名        : 決済サービス DBコネクション枯渇 + 応答遅延
イベント活性化    : ON
レベル            : Critical（初期は警告レベルで運用→確認後に昇格を推奨）
メッセージ        : DBコネクション取得待ちが滞留し、応答時間も悪化しています。
データ照会範囲    : 5分
条件式            : avg_resp_time_ms > 3000 && dbc_wait_count > 5
```

条件式で使用できる主な記号は以下のとおりです。

記号	意味
>、<、>=、<=、==、!=	比較演算子
&&	AND（両方の条件を同時に満たす）
||	OR（いずれかの条件を満たす）

4) アラートの発報量を予測する

設定を保存する前に、アラート条件テストで過去データに対するシミュレーションを実行します。

1. アラート条件テストセクションで、対象とする日付を選択します。
2. 実行ボタンをクリックします。
3. 結果として、「1日に何件のアラートが発報されるか」が表示されます。

【テスト結果の判断例】
・直近1週間で500件発報 → 条件が緩すぎる（毎時数件のノイズ）
→ しきい値を上げる（avg_resp_time_ms > 5000、dbc_wait_count > 10 など）
・直近1週間で0件発報 → 条件が厳しすぎる（実際の障害の際も検知できない可能性がある）
→ しきい値を下げる、または時間単位を見直す
・直近1週間で3〜10件発報 → 適切なアラート頻度
→ このまま保存

5) アラートを適用する

適切な発報量になるよう条件を調整したら、画面下部の保存ボタンをクリックして設定を確定します。これでアラートが稼働を開始し、条件を満たすイベントが発生するたびに登録済みの通知チャンネル（Email/Slack等）に通知が届きます。

本番適用のコツ：最初はイベントレベルを情報または警告として設定し、1〜2週間運用してみて誤検知がほとんどないことを確認してから Critical に昇格させると、夜間呼び出しのリスクを最小化できます。

🛠️ 実践ガイド③ ：ヒットマップパターンで自動検知する

WhaTap APMの特徴的な機能の一つが、ヒットマップ（トランザクション応答時間の分布図）のパターンを自動検知するアラート設定です。

設定メニューへのアクセス：アラート通知 > イベント設定 > ヒットマップパターン

ヒットマップパターン検知は、以下のような「目視でしか気づけなかった異常」をシステムが自動的に検出し、アラートを発報します。

パターン	意味	従来の検知方法
縦線パターン	特定の時刻に全トランザクションが一斉に遅延	エンジニアがヒットマップを常時監視
横線パターン	特定のレスポンスタイムに集中（外部API遅延の疑い）	手動でのトレース分析
密集パターン	短時間にトランザクション数が急増（突発的負荷）	リアルタイムダッシュボードの目視

2.3 AI異常検知（Anomaly Detection）

WhaTapのAI異常検知は、過去のメトリクスデータから機械学習モデルが正常パターンを自動学習し、「いつもと違う振る舞い」を検知するとアラートを発報します。

従来の固定しきい値アラートとの根本的な違いは、時間帯やビジネスサイクルを考慮した動的な判定が可能な点です。

設定メニューへのアクセス：アラート通知 > イベント設定 > 異常検知

AI異常検知は学習なしで利用開始可能です。エージェントがデータを収集し始めると、WhaTapのAIエンジンがバックグラウンドで自動的にパターン学習を開始します。

アラートイベントタイプの使い分けガイド

状況	推奨イベントタイプ	理由
「CPU 95%超過は絶対に即時に知りたい」	メトリクスイベント	明確なしきい値があり、即時対応が必要
「複数の兆候が同時に出たときだけ通知」	複合メトリクスイベント	単一指標では誤報が多いので、複合条件で精度を向上
「トランザクションが突然遅くなったら」	ヒットマップパターン	しきい値では定義しがたい分布パターンの変化
「普段と何か違う、を自動で拾いたい」	AI異常検知	しきい値を人が定義できない動的な異常

第3章：チーム別アラート振り分けとAIイベント分析

～ 適切な人に、適切なタイミングで、アラート原因まで届ける ～

3.1 アラート通知における運用上の課題

Zabbixのアクション機能は、ホストグループ、トリガーの重要度、時間帯に応じた条件付き通知を可能にします。しかし、実際の運用現場では、以下の問題が繰り返し発生します。

課題①：メンバー全員へ全ての通知が届く「アラートの洪水」

Zabbixのデフォルト設定では、監視に参加している全ユーザーへ全てのアラートが届きます。インフラチーム向けのディスクアラートがアプリケーション開発者にも届き、逆にアプリケーションのエラー率アラートがインフラ担当者にも届きます。アクションの条件フィルタリングで振り分けは可能ですが、設定が煩雑化しやすく、結果として全員が「自分には関係ないアラート」を日常的に無視する習慣がつくことがあります。これにより、本当に対応が必要なアラートも見逃してしまうアラート疲れ（オオカミ少年効果）現象が生じます。

課題②：アラートを受け取っても「次に何をすべきか」が分からない

Zabbixのアラート通知は、メトリクス名、しきい値、現在値、ホスト名を伝えますが、**「なぜこの値になったのか」「どのような対処を取るべきか」**までは提示してくれません。結果として、アラートを受け取ったエンジニアは、自らダッシュボードやログを渡り歩いて原因を調査する必要があります。

3.2 チームや担当別のアラート振り分け

WhaTapのイベント受信タグ機能は、「特定のアラートは特定のチームや担当者にのみ配信する」というルーティングを、コーディング不要で実現します。

> **🛠️ 実践ガイド④ ：チーム別にアラートを振り分ける**

Step 1：受信タグの作成

設定メニューへのアクセス：アラート通知 > 通知設定

タグ名：infra-team
対象メンバー：○○（インフラリーダー）、△△（インフラ担当）
3rd Partyプラグイン：#infra-alerts（Slackチャンネル）

タグ名：app-team
対象メンバー：□□（アプリリーダー）、◎◎（アプリ開発者）
3rd Partyプラグイン：#app-alerts（Slackチャンネル）

Step 2：イベントへのタグ紐付け

各イベント設定画面のイベント受信タグフィールドから、作成したタグを選択します。

イベント名	受信タグ	配信先
CPU使用率の超過	infra-team	インフラチームのみ
ディスクI/O異常	infra-team	インフラチームのみ
レスポンスタイム異常	app-team	アプリチームのみ
エラー率の上昇	app-team	アプリチームのみ
DBコネクション枯渇	infra-team、app-team	両チーム

イベント受信タグが未設定の場合は、プロジェクトの全てのメンバーへ配信されます。既存のアラート設定を維持しながら、段階的にルーティングを導入できます。

3.3 イベント履歴とAIによる原因分析

WhaTapのイベント記録（Event History）画面では、過去最大1年間のアラート発報履歴を時系列で確認、検索できます。

設定メニューへのアクセス：アラート通知 > イベント履歴

イベント記録画面では、以下の項目が確認できます。

- **イベント名：**発報されたアラートのタイトル
- **イベント発生時刻：**アラートが発報された日時
- **イベント状態：**進行中 / 解消済み
- **イベント解消時刻：**アラートが自動解消された日時
- **処理内容：**担当者が入力した対応メモ

🛠️ 実践ガイド⑤ ：AIでイベント原因を自動分析する

WhaTap Monitoringでは、発生したイベント毎の原因調査をAIで自動実行できます。運用管理者が発生したイベントの原因を迅速に把握する必要がある場合、イベント履歴の [AI 分析] ボタンをクリックするだけで、WhaTap AIがイベント発生時点のシステムコンテキストを自動分析し、イベント発生原因を推定、その対処策までを提示します。

従来の調査フローとの比較

比較項目	従来の調査フロー	WhaTap AIによる調査
原因調査の起点	アラート受信 → 手動でダッシュボードを確認	アラート受信 → [AI 分析]ボタンをクリック
調査に要する時間	15〜60分（複数画面の渡り歩き）	数秒〜数十秒（AIによる自動分析）
必要な専門知識	対象システムへの知見	なし
対応可能な人員	ベテランエンジニア(属人性)	誰でも初動対応可能

WhaTap AIによるアラート原因の調査機能は、WhaTap Monitoringの全製品（APM、サーバー、Kubernetes、データベース、RUMなど）のイベント記録画面で利用できます。イベントタイプを問わずメトリクスイベント、複合イベント、異常検知イベントのいずれでもAI分析が実行できます。

3.4 アラートノイズを抑止するオプション

第2章では複合メトリクスイベント、ヒットマップパターン、AI異常検知という「アラート発生自体を減らす」方法について説明しました。ここでは、異常検知後の通知段階で過剰なアラート通知を制御するための3つの実用的なオプションを紹介します。これらはすべて、メトリクスイベント等のイベント設定画面の基本オプションとして用意されています。

① イベント通知の一時中止（ミュート）

短時間で同一イベントが繰り返し発生する場合に、最初の通知以降、指定した時間内のアラート通知を抑止する機能です。夜間バッチの瞬断などで同じエラーが連続発報されるケースで効果的です。

> **機能の仕様：**「過剰なアラート通知が発生しないようにするオプションです。最初のアラート通知の後、選択された時間にアラート通知は送信されません。また、イベント履歴メニューにも記録されません。」

設定例	効果
ミュート時間：5分	同じ条件でイベントが連続発生しても、5分間は通知を1件にする
ミュート時間：10分	バッチ起動時の連鎖エラー（10分以内に収束するもの）を1通に集約

設定箇所：イベント設定画面 > ミュート または 一時中止

② イベント状態が解消された時のイベントステータス通知

「異常発生」のアラートだけでなく、「異常が自然回復した／解消された」タイミングでも通知を受け取れるオプションです。担当者がアラート受信後、対応をしている間に問題が解消されたかどうかを正確に把握できます。なお、異常の状態が続いている場合なら、追加のアラートは発生しません。

オプションを有効化しない場合、解消通知は送信されないため、担当者は「まだ異常が続いているのか、自然回復したのか」を能動的に確認しに行く必要があります。夜間対応の負担軽減などに直結する重要な設定です。

設定箇所：イベント設定画面 > 解消された通知

③ アラート受信の曜日、時間帯の制限

ユーザーごとに「どの曜日、時間帯にアラートを受信したいか」をきめ細かく設定できます。「平日昼間はインフラチーム全員に通知、夜間と休日は当日担当のみ」といった運用に対応します。

設定例	適用シーン
平日の09:00〜18:00のみ受信	日中の就業時間のみ通知を受け取りたい開発メンバー
24時間×7日間のすべて受信	夜間のオンコール担当者
土日には受信を停止	週末オフの担当者

設定箇所：イベント設定画面 > イベント動作時間

3つのオプションの使い分け

状況	推奨オプション
「同じエラーが連続で何十件も届く」	① ミュート（イベント発生の一時中止）
「アラート対応中に問題が解消したか分からない」	② 解消通知（イベント状態が解消されたら追加に通知）
「夜間や週末は限られたメンバーだけに通知したい」	③ 受信曜日、時間帯の制限

これらのオプションは互いに干渉せず併用可能です。①でアラートノイズを抑え、②で対応完了などのステータスを知らせ、③で適切なメンバーに適切な時間だけ通知する―この3層構造で、運用チームの疲弊を最小限に抑えられます。

第4章：主なアラート機能との比較

4.1 アラート機能の比較表

項目	Zabbix	Prometheus + Alertmanager	PagerDuty	WhaTap
初期アラート設定	○ 標準テンプレートあり、カスタム項目は手動	○ YAMLファイルの記述が必要	○ 連携ツールとの組み合わせ	◎ テンプレート自動セットアップ、ONにするだけ
メトリクスアラート	○ トリガー式で柔軟	○ PromQLで高機能	△ 外部連携が前提	◎ Web画面で直感的に設定
複合条件アラート	△ トリガー式のネストが必要	△ Recording Rulesの設計が必要	△ Event Rulesで設定	◎ Web画面で設定(MXQL)
ヒットマップパターン検知	×	×	×	◎ 自動検知(WhaTapの特許技術)
AI異常検知	△ Baseline関数による簡易的な動的閾値のみ	×	△ ML機能あり(高額プラン)	◎ MLモデル標準搭載(学習不要)
AIによる原因分析	×	×	△ Generative AI(限定機能)	◎ WhaTap AI
チーム別振り分け	△ アクション条件設定が煩雑	○ Routing設定(YAML)	◎ Escalation Policy	◎ Web画面での設定のみ
シミュレーション機能	△ 条件式の静的テストのみ	×	×	◎ 過去時系列データでの発報量予測
通知チャンネル	○ Email、Slack等、Webhook設定要	△ Webhook中心	◎ 多チャンネル対応	◎ Email/Slack/Teams/Line/Pagerduty/Webhook等を標準対応
導入/維持管理の工数	中〜大 手動チューニング	大 YAML + Helm管理	小 ただし、高額	◎ 5分で設定、フルマネージドSaaS

4.2 WhaTapアラート機能の主な仕様

項目	内容	参照ドキュメント
対応イベントタイプ	メトリクス、複合メトリクス、ヒットマップパターン、異常検知、ログイベント、複合ログ、トランザクション	docs.whatap.io/ja/java/alert/event-configuration
対応通知チャンネル	Email、モバイルアプリ、Slack、Teams、Telegram、LINE、Pagerduty、Opsjenie、ilert、Webhook、Webhook JSON等	docs.whatap.io/ja/java/alert/event-notification
イベント履歴保持期間	最大1年間	docs.whatap.io/ja/java/alert/event-history
シミュレーション機能	過去時系列データに基づくイベント発報量の予測	docs.whatap.io/ja/java/alert/event-configuration/metrics
AI異常検知	機械学習による自動パターン学習、逸脱検知	docs.whatap.io/ja/java/alert/event-configuration/anomaly-detection
AIイベント分析	AIによる原因分析と対処策の提示（WhaTapのイベント記録画面で利用可能）	docs.whatap.io/ja/ai/introduction
WhaTap AI 対応言語	日本語、英語、韓国語	docs.whatap.io/ja/ai/introduction
イベント発生の一時中止(ミュート)	連続して通知する同一条件のイベントを指定時間内には1件にする	docs.whatap.io/ja/java/alert/event-configuration/metrics
解消通知	イベント状態が解消された際のステータス通知(ON/OFF切替)	docs.whatap.io/ja/java/alert/event-configuration/metrics
受信曜日、時間帯制限	ユーザー別に通知を受け取る曜日、時間帯をきめ細かく設定	docs.whatap.io/ja/java/alert/event-notification

おわりに：アラートを「通知」から「アクション」へ進化させる

Zabbixは今日においても、多くの企業のインフラ健全性を支え続けている信頼性の高い監視ツールです。WhaTapの目指すところは、既存のZabbixアラート体制をすべて置き換えることだけではありません。

Zabbixが長年培ってきたインフラ、ネットワーク、OS層の監視基盤はそのまま維持しながら、その先のアラート運用で課題となっていた—アラートノイズの氾濫、しきい値設定の属人性、原因調査の遅延、通知の一極集中—「アラート運用の空白地帯」を、WhaTapがインテリジェントに補完することも可能です。

本ガイドでご紹介した中核的な機能—5分で完了するWeb画面でのアラート設定、複合メトリクス、ヒットマップパターン、AI異常検知による多層防御、チーム別のスマートルーティング、そしてAIによるワンクリ

ック原因分析―は、既存の監視体制を変更することなく、WhaTapエージェントを追加するだけで今日からすぐご活用いただけます。

アラートを「届いたら確認する」受動的な通知から、「原因と対処が一緒に届く」能動的なアクションへ。WhaTapのアラート機能が、貴社の運用チームの夜間対応負荷を軽減し、障害対応のスピードを次のステージへと引き上げます。

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
最初のアラートを設定する	https://docs.whatap.io/ja/best-practice-guides/quick-win-first-alert
チーム別アラート振り分け	https://docs.whatap.io/ja/best-practice-guides/quick-win-alert-routing
イベント設定の概要	https://docs.whatap.io/ja/java/alert/event-configuration
メトリクスイベント	https://docs.whatap.io/ja/java/alert/event-configuration/metrics
複合メトリクスイベント	https://docs.whatap.io/ja/java/alert/event-configuration/composite-metrics
ヒットマップパターン検知	https://docs.whatap.io/ja/java/alert/event-configuration/hitmap-pattern
異常検知	https://docs.whatap.io/ja/java/alert/event-configuration/anomaly-detection
ログイベント	https://docs.whatap.io/ja/java/alert/event-configuration/log-event
複合ログイベント	https://docs.whatap.io/ja/java/alert/event-configuration/log-composite
トランザクションイベント	https://docs.whatap.io/ja/java/alert/event-configuration/transaction
イベント受信設定	https://docs.whatap.io/ja/java/alert/event-notification
イベント記録	https://docs.whatap.io/ja/java/alert/event-history
リアルタイムアラート	https://docs.whatap.io/ja/java/alert/real-time-notifications
システムイベント	https://docs.whatap.io/ja/java/alert/system-events
WhaTap AI	https://docs.whatap.io/ja/ai/introduction