Engineer

アラート

人の対応が必要なときにシステムが送るメッセージ

どのようなものか

アラートは、監視対象の条件が、人の作業を中断してでも伝えるべき程度に長く続いたことを示す、自動の信号です。アラームと呼ぶチームもあり、オンコール担当者を起こすほどの強い通知は、通常pageと呼びます。

この語は二種類のものを指し、Prometheusは、その違いが分かるように分離しています。Prometheusのアラートルールは、式が結果を返す、異なるラベルの組み合わせごとに、一つのアラートを生成します。それらを別のプログラムであるAlertmanagerへ送り、メール、チャット、当番サービスへの呼び出しといった通知に変えます。プロジェクトは、アラートルール単体は完全な通知システムではないと明言しています。分離が重要なのは、数が一致しないからです。五十のアラートが一通にまとまり、一つのアラートが数時間ごとの再通知を生み、ミュートしたアラートは、発火し続けても誰にも届きません。

アラートには短いライフサイクルがあります。式が何も返さない間は非アクティブです。最初に結果を返すと保留になり、ルールの `for` 句が指定する間、そのまま待ちます。待ち時間の終わりにも条件が成り立っていれば、発火して配信へ渡します。条件が成り立たなくなれば、解消します。新しい `keep_firing_for` 句は、条件が解消しても一定時間発火を維持します。文書では、頻繁な状態の切り替わりや、誤った解消通知への対策としています。Prometheusは、保留と発火の状態を `ALERTS` という時系列にも記録するため、アラート自体を問い合わせ、グラフにできます。

発火まで20分待ち、解消にもさらに10分待つルールは、次のようになります。

```
groups:
    - name: exports
      rules:
          - alert: ExportQueueStalled
```

```
            expr: exports_waiting > 0 and rate(exports_done_total[15m]) == 0
            for: 20m
            keep_firing_for: 10m
            labels:
                severity: page
                team: billing
            annotations:
                summary: "{{ $value }} exports queued and none finishing"
```

ラベルと注釈は見た目が似ていますが、役割は違います。ラベルは、そのアラートを他と区別するIDの一部で、Alertmanagerはそれを読んで送り先を決めます。`severity: page` は電話へ、`severity: ticket` は対応待ちへ送る、といった使い方です。注釈には、届いた後に人が必要とする言葉を入れます。一行の要約、詳しい説明、最初の対応を示す手順書へのリンクなどです。注釈のテンプレートには、アラートのラベルや、発火の原因となった値を使えるため、上の要約では数値を示せます。

アラートとインシデントが別の語なのには理由があります。アラートは、条件が成り立ったと報告します。それをインシデントと判断するのは、人です。

概念を判断に生かす

「アラートとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 障害が小さいうちに知らせる監視とアラートを整え、安定した状態をどれだけ維持できているかを数値で示します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 社内初のオブザーバビリティダッシュボードを構築 — 社内システムの状態を初めて可視化

社内初のオブザーバビリティダッシュボードを構築 お問い合わせ →

なぜ重要なのか

監視の中で、見に来てもらうのを待たないのはアラートだけです。指標はストレージに、ダッシュボードは画面に、ログは検索されるまで留まりますが、アラートは誰かの夜の時間に割り込みます。各ルールは、人の注意を常時少しずつ要求するものであり、受け手が何かをできる場合にだけ、その要求は妥当です。Prometheusの指針も、何もすることがない緊急通知を避け、アラートを単純に保つよう勧めています。

二つ目の理由は、件数です。一つの原因で多くのルールが同時に反応します。ネットワークの分断により、サービスの半分のインスタンスがデータベースから切り離されると、それぞれがアラートを発火できます。Alertmanagerの文書も、ま

さにその例で、何百ものアラートを一通にする仕組みを説明します。`group_by`で指定したラベルを共有するアラートを集め、`group_wait`（標準は30秒）だけ待って、最初の通知にまとめます。グループの更新は`group_interval`（5分）より頻繁には送らず、発火が続く通知は`repeat_interval`（4時間）の後に繰り返します。抑止はさらに進み、別のアラートが発火している間、一部を抑えます。例えばクラスタ全体に到達できない通知で、その内側の各サービスの通知を止めます。サイレンスは、ラベルに一致するアラートを一定期間ミュートし、通常は計画保守で使います。

三つ目は、終わり方です。多くの通知先の設定では解消時にも通知を送りますが、メール通知の`send_resolved`は既定でfalseです。メールを使うチームでは、問題の開始だけが通知され、終わりが届かない場合があります。

実際の活用例

自社ホストは、信号が届かないときに警報を出す仕組みを使います。464 MBのメモリーでウェブサーバーとフォージを動かしており、メモリー不足とディスク容量不足は重要な二つの障害リスクです。ホストのタイマーは、利用可能メモリーと空きディスク容量が設定した閾値の条件を満たす間だけ外部サービスへ信号を送ります。十五分ごとにランダムな時間のずれを加えて送り、期待した信号が届かなくなると、外部サービスが警報を出します。ネットワーク喪失や起動失敗など、スクリプトの実行や、そのサービスへの接続を妨げる障害によって、信号が届かなくなります。信号の受信は、スクリプトが動作し、利用可能メモリーと空きディスク容量が設定した閾値の条件を満たしたことを確認します。ホスト全体の健全性や、サイトが配信できていることを証明するものではありません。監視によって兆候を捉えられますが、その時期は負荷や観測する条件によって異なります。

同じ反転は、定期作業も守れます。バックアップの実装は、バックアップと復元の成功信号を分けています。スクリプトとsystemdユニットは、暗号化したホスト外のスナップショット、整合性検査、毎月の既知ファイルの復元テストに対応し、各処理が成功した後だけ信号を送ります。これは通知経路の設計であり、稼働していた証明ではありません。2026-10-27付の実装記録は、リポジトリ未設定のためタイマーを無効にしていたことを述べ、実環境で完了した復元を記録していません。処理と監視を有効にした後は、信号の途絶で外部通知を出せますが、設定だけでは稼働中とは分かりません。

反転したアラートにも固有の故障があり、プラットフォームを一台から二台へ増やす直前に見つかりました。二台で一つのスイッチを共有すると、拡張するはずの通知を弱めてしまいます。一方が信号を送り続ければ、もう一方が停止しても

正常のままになるためです。各スイッチをホスト固有の値にし、既存ホストのスイッチ項目は意図して空欄にしました。設定は配線の存在を示し、値の用意がまだ必要であることを示します。

問題が起きるところ

測定値をラベルに入れてしまいます。 ラベルはアラートを識別するため、現在の数値を含めると、変わるたびに新しいアラートとなり、前のものを解消します。一つの問題で、発火と解消が次々と届きます。ラベルはIDと経路のために使い、数値は注釈に置きます。

発火と解消を繰り返します。 しきい値付近の条件が、数分おきに発火、解消、再発火し、解消通知のたびに誤った情報を伝えます。`for` を延ばして、一時的な変化が通知前に戻るようにし、`keep_firing_for` で、一瞬しきい値を下回っても終了しないようにします。

ミュートが理由より長く残ります。 保守のため、広い一致条件と余裕を持たせた終了時刻でミュートすると、翌週の本当の障害まで届かなくなります。作業で許す限り対象を絞り、作業終了予定に近い期限を設け、Alertmanagerのコメント欄に、設定した人と理由を書きます。

広い経路が個別の経路を吸収します。 Alertmanagerは経路ツリーを順番にたどり、`continue: true` がなければ、最初に一致する子で止まります。チーム専用経路より先に全件対象の経路を置くと、そのチームのアラートも吸い込みます。最も狭い経路から並べ、頼る前にサンプルのラベルでテストします。

すべてを緊急扱いします。 ディスク70%、翌月に切れる証明書、決済サービスのエラーが、すべて同じ電話を鳴らすと、やがて無視されます。今すぐ人が動く必要がある条件だけを緊急通知し、残りはチケットや日次の要約へ送り、何か月も発火しているのに誰も対応しないルールは削除します。

参考資料

- アラートの概要：PrometheusのルールとAlertmanagerの役割分担、および両者の設定手順です。
- アラートルール：`for` 、 `keep_firing_for` 、ラベル、注釈、テンプレートを含む構文を説明します。
- Alertmanager：グループ化、抑止、サイレンス、高可用性のための複数構成を説明します。

- Alertmanagerの設定：経路ツリー、グループ化のタイマー、抑止ルール、`send_resolved` を含む受信設定を説明します。
- アラートの実践：何を緊急通知し、何を別に扱うべきかについて、プロジェクト自身の助言です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「アラートとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/alert/