Engineer

Alarm

eine Systemmeldung, wenn ein Mensch gebraucht wird

Was es ist

Ein Alarm ist ein automatisches Signal, dass ein überwachter Zustand lange genug angehalten hat, um jemanden dafür zu unterbrechen. Im Englischen lautet die Bezeichnung alert, in manchen Teams auch alarm. Ein Alarm, der laut genug ist, die Person in Rufbereitschaft zu wecken, heißt im Englischen meist Page.

Das Wort bezeichnet zwei verschiedene Dinge, die Prometheus anschaulich trennt. Eine Alarmregel in Prometheus erzeugt Alarme: einen pro unterschiedlicher Labelkombination, für die der Ausdruck ein Ergebnis liefert. Diese Alarme gehen an ein separates Programm, den Alertmanager. Er macht daraus Benachrichtigungen wie eine E-Mail, eine Chatnachricht oder einen Aufruf eines Rufbereitschaftsdienstes. Das Projekt sagt ausdrücklich, dass Alarmregeln allein kein vollständiges Benachrichtigungssystem sind. Die Trennung ist wichtig, weil die Zahlen auseinandergehen. Fünfzig Alarme können als eine Nachricht ankommen, ein Alarm kann alle paar Stunden eine Erinnerung erzeugen, und ein stummgeschalteter Alarm erreicht niemanden, obwohl er weiterhin aktiv ist.

Ein Alarm hat einen kurzen Lebenszyklus. Solange der Ausdruck nichts zurückgibt, ist er inaktiv. Beim ersten Ergebnis ist er ausstehend und bleibt das so lange, wie die `for`-Klausel verlangt. Ist die Bedingung nach dieser Wartezeit weiterhin erfüllt, wird er aktiv und zur Zustellung weitergereicht. Gilt die Bedingung nicht mehr, ist der Alarm behoben. Eine neuere Klausel, `keep_firing_for`, hält einen aktiven Alarm nach dem Wegfall seiner Bedingung noch eine Weile offen. Die Dokumentation beschreibt dies als Schutz gegen ständiges Umschalten und fälschliche Entwarnungen. Prometheus erfasst ausstehende und aktive Zustände außerdem in einer Zeitreihe namens `ALERTS`, sodass auch die Alarme selbst abgefragt und grafisch dargestellt werden können.

Eine Regel, die vor der Aktivierung zwanzig Minuten wartet und vor der Entwarnung weitere zehn, sieht so aus:

```
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"
```

Labels und Annotationen sehen ähnlich aus, erfüllen aber verschiedene Aufgaben. Labels gehören zur Identität eines Alarms. Der Alertmanager verwendet sie zur Wahl des Empfängers: `severity: page` könnte ein Telefon erreichen, während `severity: ticket` in einer Warteschlange landet. Annotationen enthalten die Wörter, die ein Mensch nach dem Empfang braucht: eine einzeilige Zusammenfassung, eine längere Beschreibung oder einen Link zur Handlungsanweisung für den ersten Schritt. Vorlagen können dabei die Labels und den auslösenden Wert einsetzen. So kann die Zusammenfassung oben eine Zahl nennen.

Alarm und Incident sind aus gutem Grund unterschiedliche Wörter. Ein Alarm meldet, dass eine Bedingung vorlag. Ob dieser Zustand als Incident zu gelten hat, entscheidet ein Mensch.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Alarm?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Monitoring und Alarme melden sich, solange ein Fehler noch klein ist; Kennzahlen zeigen, wie oft das gelingt.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Das erste Observability-Dashboard aufgebaut — der erste Einblick des Unternehmens in die eigenen Systeme

Das erste Observability-Dashboard aufgebaut Kontakt aufnehmen →

Warum es wichtig ist

Ein Alarm ist der einzige Teil der Überwachung, der nicht darauf wartet, angesehen zu werden. Eine Metrik liegt im Speicher, ein Dashboard wartet auf einem Bildschirm, ein Protokoll auf eine Suche. Ein Alarm greift in den Abend eines Menschen ein. Jede Alarmregel beansprucht damit dauerhaft ein kleines Stück Aufmerksamkeit. Gerechtfertigt ist das nur, wenn der Empfänger etwas tun kann. Die Prometheus-Empfehlungen raten deshalb von dringlichen Benachrichtigungen ohne Handlungsmöglichkeit ab und empfehlen einfache Alarmierung.

Der zweite Grund ist rechnerisch. Eine Ursache löst meist mehrere Regeln zugleich aus. Trennt eine Netzwerkpartition die Hälfte der Dienstinstanzen von ihrer Datenbank, kann jede Instanz einen eigenen Alarm auslösen. Die Alertmanager-Dokumentation verwendet genau diesen Fall, um zu zeigen, wie aus Hunderten von Alarmen eine Benachrichtigung wird. Der Alertmanager sammelt Alarme mit gemeinsamen, in `group_by` benannten Labels und wartet `group_wait`, standardmäßig 30 Sekunden, damit die erste Nachricht die ganze Gruppe enthält. Aktualisierungen gehen nicht häufiger als alle `group_interval`, also 5 Minuten, hinaus. Eine noch aktive Meldung wird nach `repeat_interval`, also 4 Stunden, wiederholt. Inhibition geht weiter und unterdrückt bestimmte Alarme, solange ein anderer aktiv ist. Ein Alarm über einen unerreichbaren Cluster kann so die Meldungen seiner einzelnen Dienste unterdrücken. Silences schalten Alarme mit passenden Labels für eine feste Zeit stumm, typischerweise während geplanter Wartung.

Der dritte Grund ist das Ende der Geschichte. Die meisten Empfänger senden nach der Behebung eine Entwarnung. Beim E-Mail-Empfänger steht `send_resolved` jedoch standardmäßig auf false. Ein Team mit E-Mail-Alarmierung kann deshalb vom Beginn erfahren, ohne je vom Ende zu hören.

Wie es in der Praxis aussieht

Der eigene Host des Unternehmens verwendet einen Alarm bei ausbleibendem Signal. Seine 464 MB Speicher tragen einen Webserver und eine Forge; erschöpfter Arbeitsspeicher und eine volle Festplatte sind zwei wichtige Ausfallrisiken. Ein Timer auf dem Host sendet nur dann ein Lebenszeichen an einen externen Dienst, wenn seine Prüfungen der Grenzwerte für verfügbaren Arbeitsspeicher und freien Plattenplatz bestanden werden, alle fünfzehn Minuten mit etwas zeitlicher Streuung. Der externe Dienst löst Alarm aus, wenn die erwarteten Signale ausbleiben. Fehler, die die Ausführung des Skripts oder

den Kontakt zu diesem Dienst verhindern, darunter Netzverlust oder ein fehlgeschlagener Start, führen zu einem ausbleibenden Signal. Ein empfangenes Signal bestätigt, dass das Skript lief und seine Prüfungen der Grenzwerte für verfügbaren Arbeitsspeicher und freien Plattenplatz bestanden wurden. Es bescheinigt weder den allgemeinen einwandfreien Zustand des Hosts noch, dass die Website ausgeliefert wird. Die Überwachung kann Warnzeichen sichtbar machen; wann diese auftreten, hängt jedoch von der Arbeitslast und den beobachteten Bedingungen ab.

Dieselbe Umkehrung kann geplante Arbeit schützen. Die Sicherungsimplementierung trennt Erfolgssignale für Sicherung und Wiederherstellung. Skripte und systemd-Units unterstützen verschlüsselte Snapshots außerhalb des Hosts, Integritätsprüfungen und einen monatlichen Wiederherstellungstest mit einer bekannten Datei. Ein Erfolgssignal wird erst nach erfolgreichem Abschluss des jeweiligen Ablaufs gesendet. Das ist der Entwurf des Alarmwegs, kein Beleg dafür, dass er aktiv war. Der Implementierungsstand vom 2026-10-27 hält fest, dass die Timer ohne konfiguriertes Repository deaktiviert waren, und dokumentiert keinen abgeschlossenen Wiederherstellungslauf in der realen Umgebung. Sobald Ablauf und Überwachung aktiviert sind, kann Schweigen einen externen Alarm auslösen. Die Konfiguration allein belegt nicht, dass beides läuft.

Auch umgekehrte Alarme haben eine eigene Fehlerart. Sie zeigte sich, als die Plattform von einem Host auf zwei wachsen sollte. Ein gemeinsamer Schalter für beide hätte die Alarmierung geschwächt, die er erweitern sollte: Solange einer weiter Lebenszeichen sendet, bleibt die Anzeige grün, während der andere tot ist. Jeder Schalter wurde deshalb zu einem hostbezogenen Wert. Die Einträge des bestehenden Hosts blieben absichtlich leer, damit die Konfiguration zeigt: Die Verdrahtung ist vorhanden, der Wert muss noch geliefert werden.

Wo es schiefgeht

Ein Messwert landet in einem Label. Weil Labels einen Alarm identifizieren, erzeugt ein Label mit dem aktuellen Zahlenwert bei jeder Änderung einen neuen Alarm und beendet den vorherigen. Der Empfänger sieht für ein einziges Problem einen Strom aus Alarm und Entwarnung. Verwenden Sie Labels für Identität und Zustellung und Annotationen für Zahlen.

Der Alarm pendelt ständig zwischen aktiv und behoben. Eine Bedingung pendelt um ihren Schwellwert, wird alle paar Minuten aktiv, verschwindet und kommt wieder. Jede Entwarnung vermittelt dabei etwas Falsches. Verlängern Sie `for`, damit ein kurzer Ausschlag vor der Benachrichtigung verschwindet. Mit `keep_firing_for` beendet ein kurzes Unterschreiten des Schwellwerts den Alarm nicht sofort.

Eine Stummschaltung überlebt ihren Anlass. Jemand unterdrückt während einer Wartung mit einem breiten Matcher und großzügigem Enddatum einen Alarm. Der echte Fehler der nächsten Woche kommt dann ebenfalls stumm an. Begrenzen Sie die Auswahl so eng, wie die Arbeit erlaubt, legen Sie das Ende nahe an das geplante Arbeitsende und schreiben Sie Verantwortlichen und Grund in das Kommentarfeld des Alertmanagers.

Eine breite Route verschluckt die besonderen. Der Alertmanager durchläuft den Routingbaum der Reihe nach und stoppt beim ersten passenden Kind, sofern die Route nicht `continue: true` setzt. Eine allgemeine Route oberhalb der Teamroute übernimmt deren Alarme. Stellen Sie die engsten Routen zuerst und testen Sie den Baum mit Beispiellabels, bevor Sie sich darauf verlassen.

Alles ist dringend. Eine zu 70 % gefüllte Platte, ein nächsten Monat ablaufendes Zertifikat und Fehler des Zahlungsdienstes rufen dasselbe Telefon an. Bald wird es ignoriert. Alarmieren Sie dringlich nur bei Zuständen, auf die jetzt jemand reagieren muss. Der Rest gehört in Tickets oder eine tägliche Zusammenfassung. Entfernen Sie Regeln, die seit Monaten auslösen, ohne dass jemand handelt.

Weiterführende Lektüre

- Überblick zur Alarmierung — die Arbeitsteilung zwischen Prometheus-Regeln und Alertmanager sowie die Einrichtung beider Teile.
- Alarmregeln — die Syntax einschließlich `for`, `keep_firing_for`, Labels, Annotationen und Vorlagen.
- Alertmanager — Gruppierung, Inhibition, Silences und mehrere Alertmanager für hohe Verfügbarkeit.
- Alertmanager-Konfiguration — Routingbaum, Gruppierungszeiten, Unterdrückungsregeln und Empfängereinstellungen einschließlich `send_resolved`.
- Empfehlungen zur Alarmierung — die eigenen Hinweise des Projekts dazu, was eine dringliche Benachrichtigung verdient und was anders behandelt werden sollte.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Alarm?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/alert/