Engineer

Larm

ett meddelande från ett system när något kräver en människa

Vad det är

Ett larm är en automatisk signal om att ett villkor som någon bevakar har gällt så länge att det är värt att avbryta en människa för det. På engelska heter det alert eller alarm, och ett larm som är allvarligt nog att väcka den som har jour kallas ofta en page.

Ordet täcker två olika saker, och Prometheus håller isär dem på ett sätt som gör skillnaden lätt att se. En larmregel (alerting rule) i Prometheus skapar larm: ett för varje unik uppsättning etiketter som uttrycket ger ett resultat för. Larmen går vidare till ett separat program, Alertmanager, som gör om dem till aviseringar: ett mejl, ett chattmeddelande, ett anrop till en jourtjänst. Projektet skriver rakt ut att larmregler ensamma inte är ett komplett system för aviseringar. Uppdelningen spelar roll eftersom antalen går isär. Femtio larm kan komma fram som ett enda meddelande, ett larm kan ge en påminnelse med några timmars mellanrum, och ett larm som har tystats når ingen alls fast det fortfarande är utlöst.

Livscykeln är kort. Så länge uttrycket inte returnerar något är larmet inaktivt (inactive). När det för första gången ger ett resultat blir larmet väntande (pending), och det förblir väntande så länge regelns `for` -klausul kräver. Gäller villkoret fortfarande när väntan är över utlöses larmet (firing) och lämnas vidare för leverans. När villkoret upphör att gälla återgår larmet (resolved). En nyare klausul, `keep_firing_for` , håller ett utlöst larm öppet en stund efter att villkoret har släppt, och dokumentationen beskriver den som ett skydd mot fladder och falska återgångar. Prometheus sparar dessutom tillstånden väntande och utlöst som en tidsserie med namnet `ALERTS` , så att själva larmen går att söka i och rita grafer över.

En regel som väntar tjugo minuter innan den utlöses, och tio minuter till innan den släpper, ser ut så här:

```
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"
```

Etiketter (labels) och annoteringar (annotations) ser likadana ut men gör olika jobb. Etiketterna är en del av det som gör ett larm till just det larmet och inget annat, och Alertmanager läser dem för att avgöra vart det ska: `severity: page` kan nå en telefon medan `severity: ticket` hamnar i en kö för ärenden. Annoteringarna bär orden som en människa behöver när meddelandet väl har kommit fram: en sammanfattning på en rad, en längre beskrivning, en länk till runbooken som säger vad som ska göras först. Annoteringar kan byggas med mallar av larmets etiketter och värdet som utlöste det, och det är så sammanfattningen ovan kan citera en siffra.

Larm och incident är olika ord av ett skäl. Ett larm rapporterar att ett villkor gällde. Att avgöra om villkoret utgör en incident är en människas beslut.

Använd begreppet i ett beslut

Använd ”Vad är ett larm?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Övervakning och larm som säger till medan ett fel fortfarande är litet, och siffrorna som visar hur ofta det förblir så.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Byggde företagets första observerbarhetsdashboard — företagets första överblick över de egna systemen

Byggde företagets första observerbarhetsdashboard Hör av er →

Varför det spelar roll

Larmet är den enda delen av övervakningen som inte väntar på att någon ska titta. Systemmätvärden ligger i lagring, en dashboard väntar på en skärm, en logg väntar på en sökning, och ett larm tränger sig in i någons kväll. Det gör varje larmregel till ett litet stående anspråk på en människas uppmärksamhet, och anspråket är skäligt bara om människan kan göra något åt saken. Prometheus egna råd om larm uttrycker det som att undvika att väcka någon när det inte finns något att göra, och att hålla larmen enkla.

Det andra skälet är ren aritmetik. En enda orsak utlöser oftast många regler på en gång. När en nätverkspartition skär av hälften av en tjänsts instanser från deras databas kan varje instans utlösa ett eget larm, och dokumentationen för Alertmanager använder just det fallet för att visa hur hundratals larm blir en enda avisering. Alertmanager samlar larm som delar etiketterna i `group_by` och väntar `group_wait` (30 sekunder som standard), så att det första meddelandet rymmer hela omgången. Därefter skickar den uppdateringar för gruppen högst en gång per `group_interval` (5 minuter) och upprepar ett meddelande som fortfarande gäller efter `repeat_interval` (4 timmar). Undertryckning (inhibition) går ett steg längre och håller tillbaka vissa larm medan ett annat är utlöst, så att ett larm om att ett helt kluster är onåbart kan tysta larmen från varje tjänst därinne. En silence stänger av larm som matchar en uppsättning etiketter under en bestämd tid, oftast under planerat underhåll.

Det tredje skälet gäller slutet på historien. De flesta mottagare skickar ett meddelande om återgång när larmet släpper, men e-postmottagarens inställning `send_resolved` är som standard false. Ett team som larmar via e-post kan alltså se problemet annonseras och aldrig se det ta slut.

Så ser det ut i praktiken

Företagets egen värd använder ett larm som reagerar på en utebliven signal. Dess 464 MB minne betjänar en webbserver och en forge, och minnesbrist och en full disk är två viktiga felrisker. En timer på värden pingar en extern tjänst bara så länge dess tröskelkontroller för tillgängligt minne och ledigt diskutrymme godkänns, var femtonde minut med slumpmässig förskjutning, och den externa tjänsten larmar när de förväntade pingarna uteblir. Fel som hindrar skriptet från att köras eller nå tjänsten, däribland ett tappat nätverk eller en misslyckad start, ger en utebliven ping. En mottagen signal bekräftar att skriptet kördes och att dess tröskelkontroller för tillgängligt minne och ledigt diskutrymme godkändes;

den intygar inte värdens allmänna tillstånd eller att webbplatsen svarar. Övervakning kan visa varningssignaler, men tidpunkten beror på belastningen och de förhållanden som observeras.

Samma omvändning kan skydda schemalagda jobb. I backupimplementeringen har säkerhetskopiering och återställning var sin framgångssignal. Skripten och systemd-enheterna stöder krypterade ögonblicksbilder utanför värden, integritetskontroller och ett månatligt test som återställer en känd fil; en framgångssignal skickas först när respektive arbetsflöde har lyckats. Det är larmvägens utformning, inte bevis för att den var aktiv. Implementationsunderlaget daterat 2026-10-27 anger att tidsstyrningarna var avaktiverade utan ett konfigurerat lager och redovisar ingen genomförd återställning under verkliga driftförhållanden. Tystnad kan utlösa ett externt larm när arbetsflödet och dess övervakning har aktiverats; konfigurationen ensam visar inte att de körs.

Omvända larm har ett eget felläge, och det visade sig när plattformen skulle växa från en värd till två. Att låta båda värdarna dela ett dödmansgrepp hade försvagat larmen som det var tänkt att utvidga: så länge någon av värdarna fortsätter att pinga förblir kontrollen grön medan den andra är död. Dödmansgreppen delades upp med en post per värd, och den befintliga värdens poster lämnades tomma med avsikt, så att konfigurationen säger att kopplingen finns och att värdet återstår att fylla i.

Var det går fel

Ett uppmätt värde hamnar i en etikett. Eftersom etiketterna identifierar ett larm gör en etikett som bär det aktuella talet varje ändring av talet till ett nytt larm, och det förra återgår. Mottagaren ser en ström av par, utlöst och återgått, för ett och samma problem. Använd etiketter för identitet och routning, och lägg siffror i annoteringar.

Larmet fladdrar. Ett villkor som pendlar kring sitt tröskelvärde utlöses, återgår och utlöses igen med några minuters mellanrum, och varje meddelande om återgång säger något falskt till den som läser. Förläng `for` så att en tillfällig topp hinner försvinna innan någon hör om den, och använd `keep_firing_for` så att en kort dipp under tröskeln inte avslutar larmet.

En silence lever längre än sitt skäl. Någon tystar ett larm inför underhåll med en bred matchning och en generös sluttid, och nästa veckas verkliga fel kommer fram tystat. Matcha silences så snävt som arbetet tillåter, ge dem en sluttid nära den tidpunkt då arbetet borde vara klart, och skriv vem som satte dem och varför i kommentarsfältet som Alertmanager erbjuder.

En bred rutt sväljer de specifika. Alertmanager går igenom routningsträdet i ordning och stannar vid det första barnet som matchar, om inte en rutt sätter `continue: true`. En samlingsrutt som ligger ovanför ett teams egen rutt tar med sig teamets larm. Lägg de snävaste rutterna först och testa trädet med exempeletiketter innan ni förlitar er på det.

Allt är bråttom. Disk på 70 %, ett certifikat som går ut nästa månad och en betaltjänst som returnerar fel går alla till samma telefon, och snart börjar telefonen ignoreras. Larma bara för villkor som en människa måste agera på nu, skicka resten till en kö för ärenden eller en daglig sammanfattning, och radera en regel som har utlösts i månader utan att någon har agerat på den.

Läs mer

- Alerting overview — hur larmregler i Prometheus och Alertmanager delar på arbetet, och stegen för att sätta upp båda.
- Alerting rules — syntaxen för regler, med `for`, `keep_firing_for`, etiketter, annoteringar och mallar.
- Alertmanager — gruppering, undertryckning, silences och hur fler än en Alertmanager körs för hög tillgänglighet.
- Alertmanager configuration — routningsträdet, grupperingens timers, regler för undertryckning och varje mottagares inställningar, bl.a. `send_resolved`.
- Alerting practices — projektets egna råd om vad som är värt att väcka någon för och vad som hör hemma någon annanstans.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är ett larm?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/alert/