Engineer

Alert

a message a system sends when something needs a person

What it is

An alert is an automatic signal that some watched condition has stayed true long enough to be worth interrupting somebody. Some teams call it an alarm, and an alert loud enough to wake the person on call is usually called a page.

The word covers two different objects, and Prometheus keeps them apart in a way that makes the distinction easy to see. An alerting rule in Prometheus produces alerts: one per distinct set of labels for which the expression returns a result. Those alerts go to a separate program, the Alertmanager, which turns them into notifications: an email, a chat message, a call to an on-call service. The project says plainly that alerting rules on their own are not a full notification system. The split matters because the numbers diverge. Fifty alerts can arrive as one message, one alert can produce a reminder every few hours, and a silenced alert reaches nobody at all while still firing.

An alert has a short life cycle. While the expression returns nothing, the alert is inactive. When it first returns a result, the alert is pending, and it stays pending for as long as the rule's `for` clause asks. If the condition is still true at the end of that wait, the alert fires and is handed on for delivery. When the condition stops being true, the alert resolves. A newer clause, `keep_firing_for`, holds a firing alert open for a while after its condition clears, which the documentation offers as a guard against flapping and false resolutions. Prometheus also records the pending and firing states as a time series named `ALERTS`, so the alerts themselves can be queried and graphed.

A rule that waits twenty minutes before firing, and ten more before it lets go, looks like this:

```
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 and annotations look alike and do different jobs. Labels are part of what makes an alert this alert and not another one, and the Alertmanager reads them to decide where it goes: `severity: page` might reach a phone while `severity: ticket` lands in a queue. Annotations carry the words a person needs once the message arrives: a one-line summary, a longer description, a link to the runbook that says what to do first. Annotations can be templated with the alert's labels and the value that triggered it, which is how the summary above can quote a number.

Alert and incident are separate words for a reason. An alert reports that a condition held; deciding that the condition amounts to an incident is a person's call.

Use the concept in a decision

Put "What is an alert?" to work in a real decision. We can help relate the concept to the system you are building or buying. Monitoring and alerts that speak up while a fault is still small, and the figures that show how often it stays that way.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Built the company's first observability dashboard — the company's first view of its own systems

Built the company's first observability dashboard Get in touch →

Why it matters

An alert is the one piece of monitoring that does not wait to be looked at. A metric sits in storage, a dashboard waits on a screen, a log waits for a search, and an alert reaches into someone's evening. That makes every alerting rule a small

standing claim on a person's attention, and the claim is only fair if the person can do something with it. The Prometheus guidance on alerting puts it as advice to avoid pages where there is nothing to do, and to keep alerting simple.

The second reason is arithmetic. A single cause usually trips many rules at once. When a network partition cuts half of a service's instances off from their database, each instance can fire its own alert, and the Alertmanager documentation uses exactly that case to show hundreds of alerts becoming a single notification. The Alertmanager collects alerts that share the labels named in `group_by`, waits `group_wait` (30 seconds by default) so the first message holds the whole batch, sends updates for the group no more often than `group_interval` (5 minutes), and repeats a still-firing message after `repeat_interval` (4 hours). Inhibition goes a step further and suppresses some alerts while another is firing, so an alert saying a whole cluster is unreachable can quiet the per-service alerts from inside it. Silences mute alerts that match a set of labels for a fixed period, typically during planned maintenance.

The third reason is the end of the story. Most receivers send a resolved message when the alert clears, but the email receiver's `send_resolved` setting defaults to false, so a team that alerts by email can see the problem announced and never see it end.

What it looks like in practice

The company's own host uses a missing-signal alarm. Its 464 MB of memory serves a web server and a forge, and memory exhaustion and a full disk are two important failure risks. A timer on the host pings an outside service only while its available-memory and free-disk thresholds pass, every fifteen minutes with some jitter, and the outside service raises the alarm when the expected pings stop. Failures that prevent the script from running or reaching that service, including a lost network or a failed boot, produce a missing ping. A received signal confirms that the script ran and its available-memory and free-disk thresholds passed; it does not certify general host health or that the site is serving. Monitoring can expose warning signs, although the timing depends on the workload and the conditions observed.

The same inversion can guard scheduled work. The backup implementation separates success signals for backup and restore. Its scripts and systemd units support encrypted off-host snapshots, integrity checks and a monthly known-file restore test; a success signal is sent only after its workflow succeeds. That is the

design of the alert path, not proof that it was active. The implementation record dated 2026-10-27 says the timers were disabled without a configured repository and records no completed live restore. Silence can raise an external alert once the workflow and its monitor are activated; configuration alone does not establish that they are running.

Inverted alerts have a failure mode of their own, and it showed up when the platform was about to grow from one host to two. Sharing one switch between both hosts would have weakened the alerting it was meant to extend: as long as either host keeps pinging, the check stays green while the other one is dead. Each switch became a per-host value, and the existing host's switch entries were left blank on purpose, so the configuration says the wiring exists and the value still has to be supplied.

Where it goes wrong

A measured value ends up in a label. Because labels identify an alert, a label that carries the current number turns each change of that number into a new alert and resolves the previous one. The receiver sees a stream of fire-and-resolve pairs for one problem. Keep labels for identity and routing, and put numbers in annotations.

The alert flaps. A condition that hovers around its threshold fires, resolves and fires again every few minutes, and each resolved message tells the reader something false. Lengthen `for` so a blip clears before anyone hears of it, and use `keep_firing_for` so a brief dip below the threshold does not end the alert.

A silence outlives its reason. Someone mutes an alert for maintenance with a broad matcher and a generous end time, and next week's real failure arrives muted. Match silences as narrowly as the work allows, give them an end time close to when the work should finish, and write who set them and why in the comment field the Alertmanager provides.

A broad route swallows the specific ones. The Alertmanager walks the routing tree in order and, unless a route sets `continue: true`, stops at the first child that matches. A catch-all route placed above a team's own route takes that team's alerts with it. Put the narrowest routes first and test the tree with sample labels before relying on it.

Everything is urgent. Disk at 70%, a certificate expiring next month and a payment service returning errors all page the same phone, and the phone soon gets ignored. Page only for conditions a person has to act on now, send the rest to a ticket queue or a daily summary, and delete a rule that has fired for months without anyone acting on it.

Further reading

- Alerting overview — how Prometheus alerting rules and the Alertmanager divide the work, and the steps to set the two up.
- Alerting rules — the rule syntax, including `for`, `keep_firing_for`, labels, annotations and templating.
- Alertmanager — grouping, inhibition, silences and running more than one Alertmanager for high availability.
- Alertmanager configuration — the routing tree, the grouping timers, inhibit rules and each receiver's settings, `send_resolved` among them.
- Alerting practices — the project's own advice on what deserves a page and what belongs elsewhere.



Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is an alert?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

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

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/alert/