Engineer

Postmortem

a written review of an incident, made to stop it recurring

What it is

A postmortem is the written account a team produces after an incident: what happened, what it cost, what was done to stop it, why it happened, and what will change so that it does not happen the same way again. The word is also written post-mortem, with a hyphen; the meaning is the same. Google's Site Reliability Engineering book defines it in those terms, as an account that connects an incident and its consequences to the response, the causes and the work needed to prevent a repeat.

The book gives the exercise three aims: to document the incident, to understand every contributing root cause rather than the first one found, and to put preventive actions in place that actually work. The plural matters. Incidents in systems of any size rarely have a single cause, and a postmortem that stops at the first plausible explanation tends to fix the trigger and leave the conditions that let the trigger do damage.

What separates the practice from an ordinary report is that it is blameless. A blameless postmortem looks for contributing causes without accusing any person or team of bad behaviour, and it assumes that the people involved meant well and made decisions using the information available to them then. The SRE book traces the idea to healthcare and aviation, two fields where a mistake can be fatal and where every error is treated as a chance to make the system stronger. Its reasoning is practical rather than kind: a person cannot be patched, but a system or a process can.

A postmortem is not written for every hiccup, since it costs time. The book lists the usual triggers, and teams adjust the thresholds to suit them:

- user-visible downtime or degradation beyond an agreed limit;

- data loss of any kind;
- an on-call engineer having to intervene, by rolling back a release or rerouting traffic, for instance;
- a resolution time above an agreed limit;
- a failure of monitoring, which usually means a human found the problem first.

The criteria are agreed before anything breaks, so that nobody has to argue during an outage about whether this one counts, and any stakeholder may ask for a postmortem when an event falls outside them.

Use the concept in a decision

Put “What is a postmortem?” 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 incident that is fixed but not written up is likely to happen again. The SRE Workbook works through a postmortem that was published four months after the incident it described; in the meantime the same incident recurred, and the team had to reconstruct what it had learned the first time. A record written promptly, and kept where people can find it, turns one team’s bad afternoon into knowledge the whole organisation can use.

The record also forces the follow-up to exist. Ben Treynor Sloss, quoted in the Workbook, puts it bluntly: users gain nothing from a postmortem unless the team acts on its findings. Google therefore requires every postmortem that follows a user-affecting outage to have at least one tracking ticket at one of the two highest priorities. The postmortem is where the argument for that work is made, with evidence, and where its progress can be checked later.

Blamelessness is what makes the evidence trustworthy. Where people are shamed for mistakes, the SRE book observes, they stop bringing problems to light, and the Workbook adds that they may hide exactly the facts needed to prevent a recurrence. A blameless culture gets the full timeline, including the

command someone ran by mistake, because nobody is punished for telling it. The book goes further: a team or a person who writes many postmortems should not be treated as a problem, since the alternative is incidents nobody writes about.

Taken together, postmortems become a data set. Once each one carries the same metadata, an organisation can follow trends such as time to detect, time to resolve and how far an incident spread, and see whether reliability work is paying off. The same records serve as training: Google reenacts old postmortems as role-play exercises for new engineers, and runs reading clubs and a newsletter featuring a postmortem of the month.

What it looks like in practice

The SRE book publishes a complete example, Appendix D, written for a fictional search service that went down for 66 minutes under a sudden surge of traffic. Its sections are a good default: status, summary, impact, root causes, trigger, resolution, detection, action items, lessons learned (what went well, what went wrong, where the team got lucky) and a timeline in UTC. The timeline marks four moments separately: when the outage began, when the incident was declared, when the outage ended, and when the incident was formally closed. They are different moments, and the gaps between them are where the lessons usually sit.

Here is a postmortem in that shape for an invented incident, cut down to what fits on a screen:

```
Postmortem: checkout failures after a timeout change
Status:      final, reviewed
Owner:       on-call engineer, with the incident commander
Summary:     checkout failed for about a third of requests for 31 min
Impact:      about 4,100 failed checkouts, no data lost; the 28-day
             error budget for checkout spent in one evening
Root causes: payment timeout cut from 10 s to 1 s in a change that
             was reviewed as "config only"; nothing exercised it
             under peak load before release
Trigger:     the evening peak slowed the payment provider past 1 s
Detection:   error-rate alert paged on-call 6 min after the first error
Resolution:  change rolled back; failures stopped 4 min later

Action items
  Type      Priority  Owner          Done when
  prevent   P1        platform lead  config changes run the load test
  detect    P1        on-call lead   alert fires on payment timeouts
```

```
  mitigate  P2        checkout lead  rollback of config takes one step
  process   P2        incident lead  config changes appear in deploy log

Lessons
  Went well:   rollback worked first time
  Went wrong:  a timeout change was not treated as a code change
  Got lucky:   the peak was short; a sale day would have tripled it

Timeline (UTC)
  18:02  change merged and deployed
  18:40  traffic peak; first checkout failures        outage begins
  18:46  error-rate alert pages on-call
  18:52  incident declared, commander named           incident begins
  19:07  timeout change identified, rollback started
  19:11  failures stop                                outage ends
  19:41  thirty minutes at normal error rate          incident ends
```

Notice what is absent: no names in the causes, and no action item that asks anyone to be more careful. Each action has one owner, a priority and an end state that someone else can verify. Most of the raw material does not have to be written from memory either. The SRE book’s chapter on managing incidents has the incident commander keep a living document during the incident and asks for it to be kept for the postmortem, and the Workbook describes tooling that fills in the roles, the timeline and the affected services automatically.

On this site, the macOS event-buffer incident record, from September 2026, has much of a postmortem’s anatomy. It distinguishes diagnostic reports of 111 GB of compressed pages on a 36 GB machine from a separate footprint sample showing growth of about 41 MB per second. It identifies an unbounded main event stream and log events copied to subscribers that did not use them, then bounds buffers, filters subscriptions by event type and gives logging a rate budget. A full test-run comparison records 610,996 core log lines falling to 1,411. Architecture checks and budget tests defend the design, while the account states what the evidence does not establish: a measured post-fix memory figure or a production benchmark. Separate shutdown timeouts needed their own fix, and the budget deliberately allows repetitive diagnostic events to be dropped. The account of three locks on the door, none of them locked describes a review rather than an outage, but it ends where a good postmortem should: the three fixes were small, and what came out of the work was two rules that now govern the repository, so that the whole class of failure is caught rather than the three instances.

Where it goes wrong

The document names a culprit. A postmortem that says one person was solely responsible, or that describes their “careless ignorance”, teaches everyone else to keep quiet next time. Write about what happened and what the person knew at the time, not about who they are, and keep the language factual; the Workbook flags dramatic phrasing and exclamation marks as a distraction from the facts.

The action items ask humans to try harder. “Be more careful” and “improve the runbook” cannot be closed, and changing behaviour is less reliable than changing the system. Give every item a type (prevent, mitigate, detect), one owner, a priority and a verifiable end state, such as an alert that fires above a stated threshold, and open a ticket for each one in the tracker the team already watches.

The root cause stops at the trigger. “A config change caused the outage” names the trigger. The root causes are why the change could do that much harm: why the review missed it, why no test caught it, why the alert came late. Keep the two headings separate, as the SRE book’s example does.

The impact is a shrug. If the only number in the document is the duration, nobody can tell whether the fix worked. Estimate the users affected, the requests lost and the error budget spent; the Workbook’s advice is that a well-informed estimate beats having no data at all.

Four people own it, so nobody does. The postmortem needs one owner who is the single point of contact for writing it and seeing the follow-up through, with others helping. The same goes for every action item.

It is written late, or read by nobody. Memories fade within days, and a document circulated months later arrives after the next occurrence. The Workbook’s good example went out within a week of closing the incident, to a wide audience. Review it, too: “An unreviewed postmortem might as well never have existed,” as the SRE book has it. Hold regular review sessions, then keep the result in a searchable archive.

The actions never close. When the same incident keeps coming back, check whether the action items are taking too long and whether feature work keeps winning over the reliability fixes. Track closure, not authorship: Google holds FixIt weeks twice a year and rewards the engineers who close the most postmortem action items.

Further reading

- Postmortem Culture: Learning from Failure — chapter 15 of Google's Site Reliability Engineering book: triggers, blamelessness, review and the culture around it.
- Postmortem Culture: Learning from Failure — the SRE Workbook's chapter, with a bad and a good postmortem of the same incident and the reasons each is what it is.
- Example Postmortem — a complete postmortem for a fictional outage, with action items, lessons and a timeline.
- Managing Incidents — chapter 14 of the SRE book: the incident commander, the live incident document and the handoffs that the postmortem later draws on.

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 a postmortem?” 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/postmortem/