

Incident Management, SLOs and Postmortems

Version 2026 · verified 2026-11-02

SLOs and error budgets with real numbers, burn-rate alerts, severities, incident roles, status updates and blameless postmortems for product teams.

Contents

Mental model 3

SLI, SLO, SLA 3

Error budgets in numbers 4

Alerting on symptoms 6

Severity levels 7

Roles and communication 8

Blameless postmortems 10

Action items and on-call hygiene 11

Further reading 13

Mental model

An SLO is a promise about user experience that you can measure; the error budget is the unreliability that promise still allows, and spending it is permission to ship fast until the promise is at risk. An incident is the moment the budget burns faster than planned, and it is run by roles, not by whoever is loudest. A postmortem then fixes the system that let a reasonable person make the mistake — never the person.

SLI, SLO, SLA

Term	Is	Example
SLI	A measured ratio of good events	Non-5xx checkouts / all
SLO	A target for an SLI over a window	99.9% over 30 days
SLA	A contract with consequences	Credits below 99.5%

Measure an SLI as good events divided by valid events, as close to the user as you can — the load balancer, not a host metric. Google's quick test for SLO versus SLA: ask what happens when it is missed. If nothing explicit happens, it is an SLO. Keep the SLO stricter than any SLA so you get warned before you owe anyone money.

```
SLI = good requests / valid requests
SLO = SLI >= 99.9% over rolling 30 days
SLA = refund if monthly SLI < 99.5%
```

Error budgets in numbers

SLO	Budget	Per 30 days
99%	1%	7 h 12 min
99.5%	0.5%	3 h 36 min
99.9%	0.1%	43.2 min
99.95%	0.05%	21.6 min
99.99%	0.01%	4.32 min

The budget is simply `1 - SLO`. For request-based SLIs, count events, not minutes: at 99.9%, three million requests in a window allow 3,000 failures. Agree up front, in writing, what happens when the budget runs out — otherwise the policy gets negotiated during the outage.

```
Error budget policy (example)
  Budget left       -> ship normally
  Budget exhausted  -> freeze risky releases;
                       next sprint: reliability
  One incident used
  > 20% of budget   -> mandatory postmortem
```

> **Gotcha:** 100% is the wrong target. Past a point users cannot notice the difference through their own networks and devices, and a zero budget forbids every release. Pick the lowest target users would still be happy with.

Alerting on symptoms

Burn rate	Long / short window	Budget spent → action
14.4×	1 h / 5 min	2% → page
6×	6 h / 30 min	5% → page
1×	3 d / 6 h	10% → ticket

Burn rate is how fast you spend the budget: at 1× it lasts exactly the SLO window. The SRE Workbook pairs each long window with a short one, firing only when both breach, so the alert stops minutes after the errors do. Page on symptoms users feel (error ratio, latency), not causes (CPU, disk) — causes go to dashboards and tickets.

```
SLO 99.9%, 30-day window = 720 h
Page if error ratio > 14.4 x 0.001 = 1.44%
  over the last 1 h AND the last 5 min
14.4 x 1 h / 720 h = 2% of budget gone
```

> **Tip:** Anything that wakes a human at night must be immediately actionable by that human. If the right response is “look at it tomorrow”, route it to business hours or a ticket.

Severity levels

Level	Means	Response
SEV-1	Critical, many users, public	All hands, IC, status page
SEV-2	Major feature broken for many	IC, incident channel
SEV-3	Partial or at-risk, few users	Owning team, top priority
SEV-4	Minor, no user impact	Ticket

Lower numbers are more urgent. Define each level by user impact in your own words before you need it, and treat SEV-2 and above as major incidents with the full

role structure. PagerDuty's public scale runs SEV-1 to SEV-5; four levels are enough for most product teams.

```
Declare when ANY is true:
  - an SLO page fired
  - customers report it
  - you need help from another team
Unsure between two levels? Pick the higher.
```

> **Gotcha:** Teams lose the most valuable minutes debating whether it is "really" an incident. Declaring is cheap and downgrading is free; argue about severity in the postmortem, not on the call.

Roles and communication

Role	Owns
Incident Commander (IC)	Decisions, roles, status; does not fix

Role	Owns
Ops lead / SMEs	The only people changing production
Comms lead	Stakeholder and status-page updates
Scribe	Timeline and decisions log

The IC is the single source of truth and delegates every repair action. On a small team one person may hold several roles, but the IC should never also be debugging. Hand command over explicitly, and the outgoing IC stays until the new one acknowledges. PagerDuty posts a first public notice within 5 minutes and updates at least every 20 minutes for the first two hours.

```
[Investigating] Checkout errors - 14:05 UTC
Impact: ~30% of card payments fail in EU.
Doing: rolled back release 412; watching.
```

```
Workaround: PayPal is unaffected.
Next update: 14:25 UTC or sooner.
```

> **Tip:** Always state when the next update will come. Silence during an incident reads as “nobody is working on it”.

Blameless postmortems

Section	Answers
Summary and impact	What happened, to whom, how long
Timeline	Detection, key actions, recovery
Contributing factors	Why the system allowed it
What went well / badly	Response, not people
Action items	Owner, ticket, priority

Write one for every major incident while memory is fresh — PagerDuty schedules the review within 3

calendar days for a SEV-1 and 5 business days for a SEV-2. Assume everyone acted reasonably on what they knew; the useful question is why the system made the wrong action easy and the failure hard to see.

```
Not:  "Alex ran the wrong migration."
Yes:  "The migration ran against prod because
       the CLI defaults to the prod profile and
       nothing asks for confirmation."
```

> **Warning:** “Human error” as a root cause ends the investigation exactly where it should start. It also teaches people to hide mistakes, which removes your best early-warning signal.

Action items and on-call hygiene

Practice	Why
Few action items, each with owner and ticket	Unowned items never ship

Practice	Why
Track them like product work	Postmortems without follow-up repeat
Primary plus backup on-call	Nobody is a single point of failure
Handover notes at shift end	Open issues survive the rotation
New members shadow first	Learn before carrying the pager

Prefer action items that remove a class of failure (a guardrail, a test, an alert on the right symptom) over “be more careful”. Review paging volume regularly: alerts that fire and need no action should be fixed or deleted, because alert fatigue is how real pages get missed.

```
AI-1  Add confirm prompt for prod profile
      owner: @dana  ticket: OPS-311  P1
AI-2  Alert on checkout error ratio by region
      owner: @lee   ticket: OPS-312  P1
```

Further reading

- Google SRE Book
- Google SRE Workbook — Alerting on SLOs
- Google SRE Workbook — Incident Response
- PagerDuty Incident Response documentation
- PagerDuty Postmortem documentation