Blameless Postmortem

A template/write-up that emphasizes learning from mistakes rather than blaming people. This format is widely used in Site Reliability Engineering (SRE) teams. The focus is on learning and improvement—not assigning blame.

Here’s a structure you can use:

Blameless Postmortem: Learning from Mistakes

1. Summary

Incident Date/Time:

Impact: [Describe what was affected, e.g., service downtime, customer experience, data inconsistency]

Duration:

Detected by: [Monitoring, customer, internal team]

2. What Happened (Timeline)

\[HH\:MM] Event started

* \[HH\:MM] First alert triggered

* \[HH\:MM] Investigation began

* \[HH\:MM] Mitigation applied

* \[HH\:MM] Service restored

3. Root Cause (Technical Factors)

[Explain the technical issue without personal attribution]

Example: "The configuration file was missing validation checks, which allowed invalid values to propagate into production."

4. Contributing Factors (Systemic / Process)

Gaps in monitoring, unclear runbooks, insufficient testing, unclear ownership, etc.

Example: "The deployment checklist did not include a validation step for configuration values."

5. What Went Well

Fast detection by monitoring tools

Clear communication within the team

Quick mitigation strategy

6. What Didn’t Go Well

Alert didn’t clearly indicate the root problem

Documentation gaps caused delays

Knowledge silo (only one person knew about this component)

7. Action Items (Learning & Improvement)

Add validation checks for config files

Improve alert descriptions with actionable next steps

Update runbook for future incidents

Knowledge-sharing session on component X

8. Takeaways

Mistakes are an opportunity to learn and strengthen the system.

Focus on *system improvements*, not assigning fault to individuals.

Continuous learning ensures resilience.

Here’s a filled-in example of a Blameless Postmortem using Google’s standard format.

Blameless Postmortem Example

Title:

High Latency and Partial Outage in Image Serving API (Aug 28, 2025)

Date of Incident: August 28, 2025

Authors: Samantha Y. (SRE), Leo P. (Backend Engineer), Arjun M. (On-call)

Status: Completed

Summary

Between 10:42 Universal Time Coordinator (UTC) and 11:17 UTC, users experienced elevated latency and a 30–60% failure rate in the `image-serving.googleapis.com` API across North America and Europe. Root cause was traced to an unintended configuration change to the internal load balancer which routed traffic inefficiently between zones.

The incident was mitigated by rolling back the configuration change, restoring normal performance by 11:17 UTC.

Timeline (all times in UTC)

Time	Event
10:42	Alert triggered: 500 error rate exceeds threshold
10:45	On-call acknowledges page; starts investigation
10:49	Internal metrics show latency > 1.5s and partial timeouts
10:52	Issue localized to `us-central1` zone routing imbalance
11:02	Load balancer config change from earlier in the day identified
11:05	Change rolled back
11:17	Error rates and latency return to baseline
11:30	Post-incident review initiated

Root Cause

A new automated traffic shaping policy was deployed at 06:00 UTC as part of a routine efficiency improvement. Due to a misconfigured fallback route in the load balancer, traffic was pinned to a single unhealthy backend in the `us-central1-c` zone. This caused high latency and failures as the backend became overloaded.

Impact

Customer-facing impact:

30–60% of requests to `image-serving.googleapis.com` failed or timed out for \~35 minutes

Affected 23% of global requests, mostly in North America and Europe

Internal impact:

Alerting worked as expected

On-call response within 3 minutes

Some confusion about recent config changes delayed mitigation by \~10 minutes

Detection

Resolution

Load balancer configuration was rolled back to the last known good state.

A traffic dry-run checker was used to simulate traffic post-rollback before pushing live.

What Went Well

Alerting was fast and precise

Rollback mechanism was available and effective

Clear collaboration between SRE and Engineering teams

What Went Wrong

No automated validation for fallback routes in load balancer configs

Recent changes were not documented in shift logs

Lack of tooling to visualize zone-level traffic skew in real-time

Where We Got Lucky

Only one region was heavily impacted due to traffic shape

The misconfigured policy was pushed during a low-traffic window in Asia

| Docs Team | Improve change management logging for infra configs