Lightstep® Incident Response

Understanding Automated Incident Response for SREs

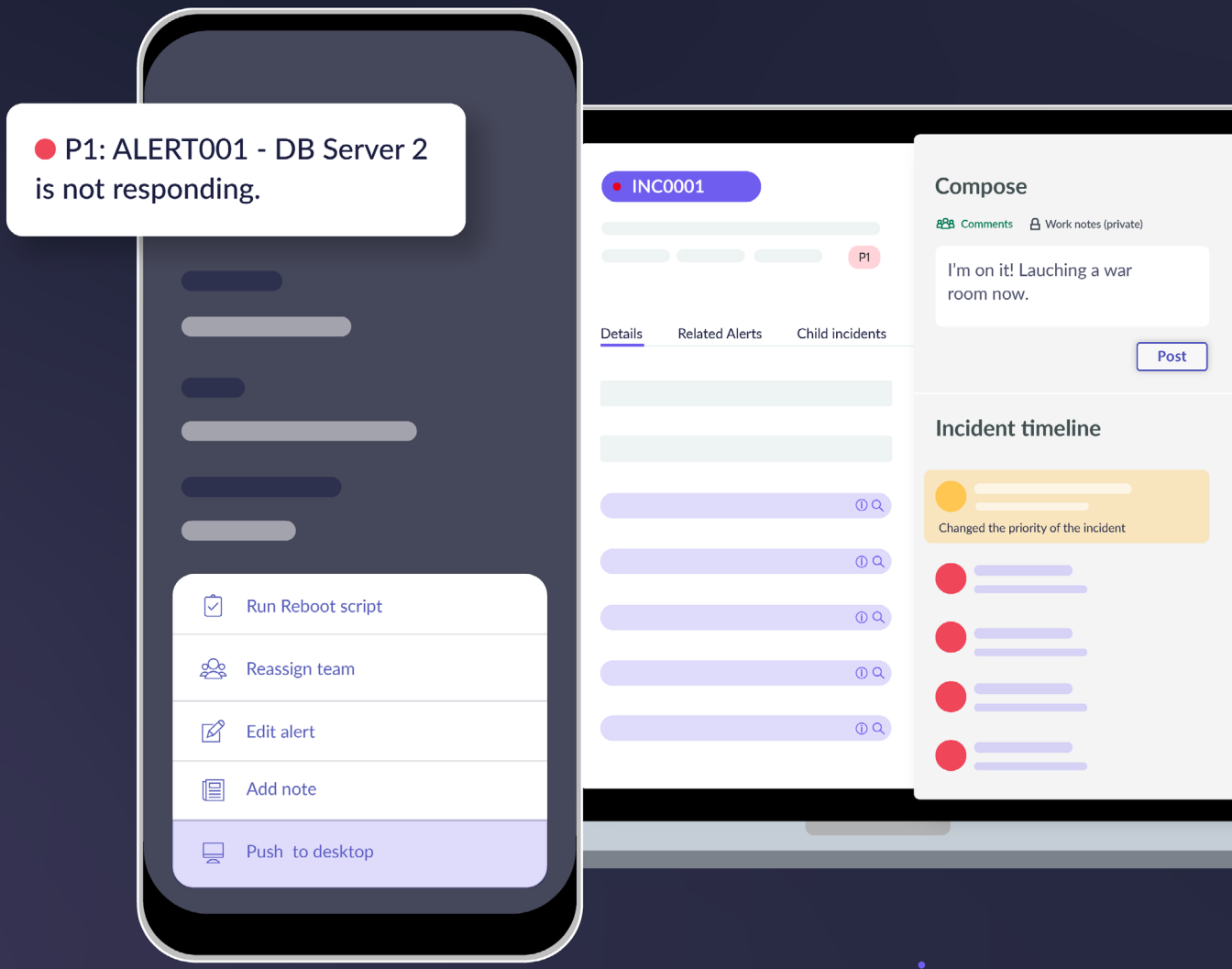


Table of Contents

Understanding Automated Incident Response for SREs

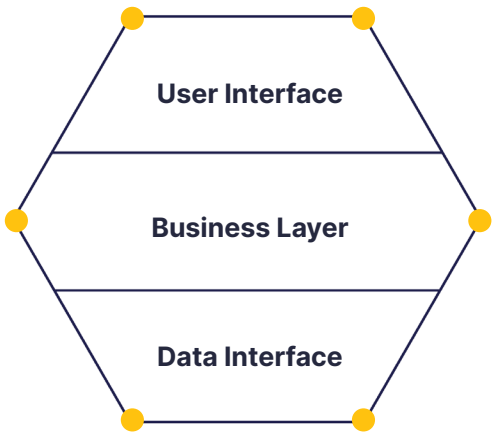
Site Reliability Engineers (SREs) have one of the most demanding jobs in the DevOps world. Modern enterprises depend on applications and software services to keep their businesses running. Errors, slowdowns, and outages are costly. So when incidents occur, we must resolve them immediately. Unfortunately, the incident resolution isn't always straightforward.

Over the past decade, most enterprises have undertaken digital transformation initiatives to help them respond more quickly to changes in the marketplace. These initiatives frequently replaced monolithic apps running on-premises with containerized apps and microservices deployed in the cloud.

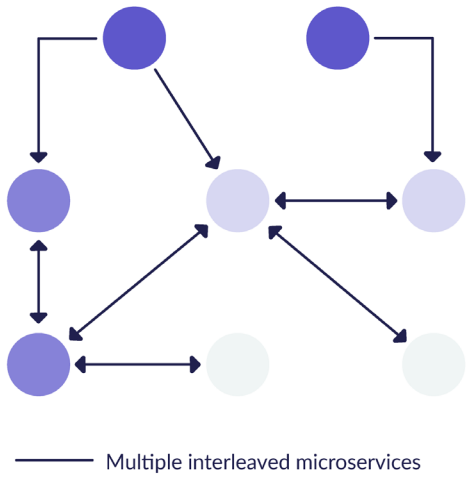
Distributed microservice architectures help enterprises adapt quickly to changing business needs. However, microservices also can make incidents more difficult to resolve. Debugging applications based on microservices is a complex process that involves understanding alerts, assembling data to understand the situation, analyzing that data, and taking manual or automated actions to remedy the situation.

Often, essential parts of the incident resolution are far too manual and repetitive. At this point, however, the day to day work of SREs has become better understood and a new generation of intelligent incident response tooling is rapidly increasing automation across the board. This paper reviews how this increased level of automation is transforming the way that SREs do their jobs.

Monolithic Architecture



Microservices Architecture



Barriers to Site Reliability Engineering Incident Response

Let's start by examining the challenges we face that can be improved with automation. Then, for each challenge, we'll explore how an automated incident response platform can help us respond to incidents more quickly and reduce mean time to resolution (MTTR).

Too much data

SREs need as much meaningful data as possible about application and service performance to diagnose an incident's cause and begin resolution. The challenge facing most SREs is that an overwhelming level of data is available when an alert is raised. For example, a typical microservice-centric app might store:

- Logs in Elasticsearch
- Distributed tracing data in Jaeger or OpenTelemetry
- Kubernetes performance data in Prometheus
- Application Performance Monitoring (APM) information in Dynatrace or Datadog

And that information is just a start. Not only do enterprise applications generate many types of data, but they also generate a staggering amount of data types that could help SREs determine an incident's cause in the real world — if we had time to sift through all of it.

The catch is that it's difficult to separate signal from noise. An SRE trying to resolve an incident doesn't always have time to sift through multiple data sources to determine what went wrong. Even worse, each monitoring system might send a separate alert when an incident occurs, forcing us to spend valuable time ensuring that all warnings we see are related to the same incident.

We need a way to ingest all our monitoring, tracing, and performance data and automatically correlate it when an incident occurs. This automated correlation ensures we have a complete view of every incident's root causes and helps us reduce MTTR.

Application complexity

While microservices typically make it easier to debug a problem with an individual service, they often add complexity when it comes to a collection of applications that share a portfolio of reusable microservices. Tracking down exactly where an incident-triggering glitch happened in a monolith was hard enough. But, if developers split a monolith into twelve microservices, each with a separate database, our job as SREs becomes more difficult. Instead of one application that can trigger alerts to wake us up at 2 AM, we now have a dozen that may depend on each other in complex ways.

Microservices themselves aren't our only worry. Every point of communication between two microservices represents a potential point of failure. What was once a failsafe and speedy method call inside a monolith can now fail or slow down due to a network outage or Kubernetes misconfiguration.

If that weren't enough, when SREs get an alert about a problem with a microservice, we can't be sure there's anything wrong with the service triggering the alert. For example, a microservice-based application might call on several microservices to serve a single web request — and each of those services might call several other services, and so on.

We could get an alert about a service that's down due to a problem in a downstream service. But that's not always obvious, so we may waste time digging through logs and APM data before we realize the problem is in another service that's a different SRE's responsibility.

Wouldn't it be nice if we had an automated system to show us all services that an incident impacts to help us pinpoint the problem's cause?

Collaboration challenges

SREs don't operate in a vacuum. We usually need to bring in other responders to help resolve incidents. But when an incident is ongoing, we typically don't want to waste time browsing the company directory or frantically CTRL+F-ing through an incident response plan, trying to figure out who to contact.

It doesn't get any easier after notifying the response team. Once all responders have acknowledged the incident, the team needs a way to communicate. Without an up-front plan, incident response teams must improvise.

Some coordinate via email. Others fire up 1:1 Slack chats. Or, one incident responder might set up a Slack channel and invite the other responders. While these ad-hoc communication approaches do work, they all require manual intervention and waste precious time.

Time spent figuring out how to communicate is time not spent resolving the incident.

If that weren't enough, incident response teams must also inform business stakeholders if an incident is severe enough. These stakeholders then need to field questions from customers and executives who want to know what's happening.

As SREs, we'd prefer to spend our time solving an incident's technical causes. We're not hermits. We enjoy interacting with our colleagues. That said, during an incident, we best spend our time attacking the incident's technical cause.

We'd rather not spend our time pinging incident responders via email or Slack. And we'd do anything to avoid drive-by requests for resolution ETA from frustrated executives. Ideally, we'd like automated systems to notify incident responders and ensure they're taking action.

Painful postmortems

SREs are busy. After we've solved a problem, we'd like to forget about it and focus on the rest of our work — or go back to enjoying our weekend if an incident happened while we were on-call.

But the incident response doesn't stop when the incident resolves. As SREs, we have a professional obligation to conduct an incident postmortem. Incidents that impact site availability are never good, but they can still provide lasting value. We can take what we've learned while diagnosing and resolving every incident and use that information to develop a plan to prevent similar incidents from happening again.

Unfortunately, postmortems can be more painful than we'd like. A proper postmortem requires specific, detailed information, including:

- What triggered the incident
- When the incident trigger(s) occurred
- A list of impacted services
- When responders were notified
- What resolution actions responders performed
- When responders performed each resolution action
- When the incident resolved

While we can reconstruct all of this after the fact, it's time-consuming and tedious. It's the very definition of toil. Time spent manually dredging up incident data doesn't add value to a postmortem. If we could automate these manual steps, we'd be able to spend more time on the most valuable parts of every postmortem:

- Incident impact assessment
- Analysis of what went right, what went wrong, where we got lucky, and lessons learned
- A list of action items to prevent incident recurrence

Overcoming Barriers with Automation

Now that we've clarified how manual toil slows us down, let's talk and end-to-end walk through the workflow automation that tools like Lightstep Incident Response offer.

Diving into data

It all starts with data. We discussed the problem — too much data from disparate services writing performance metrics and logs to multiple monitoring systems, with each system triggering its own alerts. The trouble is that SREs don't want numerous overlapping warnings — they just lead to confusion and alert fatigue.

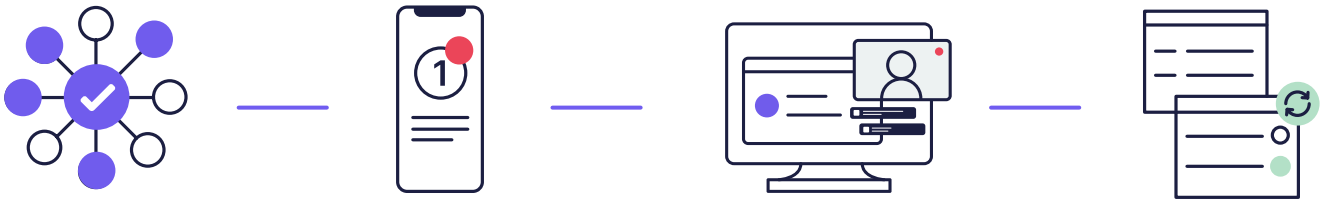
Incident response automation platforms like Lightstep help us solve data overload by talking to all the observability tools monitoring our services, such as Datadog, New Relic and Prometheus. As a result, we can centralize alerts in one place. When errors occur, we'll hear about it once — in a single notification from Lightstep — instead of hearing about it ad nauseam from every monitoring tool we use.

But that's just the start. When we receive a mobile alert from Lightstep, we can acknowledge it right from our mobile device then switch over to desktop to investigate the incident in depth.

We'll first see a summary Lightstep built from the data all of our monitoring tools supplied. We'll see which data source triggered the alert, which service it applies to, and what kind of problem is occurring.

From there, we can drill down into each data source to find out more about what's going wrong with the service that triggered the alert. The first order of business when we receive a notification is to ensure it represents an actual problem. Automation can help us here too. Lightstep, for example, will automatically correlate information from all data sources relevant to the alert — so we don't need to search for it. Instead, we can quickly gather data on the alert trigger, whether an elevated error rate, increased latency, or a complete outage.

Once we're confident the alert is not a false alarm, we can promote it to an incident and get the ball rolling on incident resolution.



Centralize alerts in one place

Mobile alert from Lightstep

Investigate the incident in depth

Incident resolution

Taming complexity

Once we've triggered an incident, the SRE managing the incident must determine how widespread the problem is. Earlier, we covered the complexity microservices can introduce. Any service involved in an incident likely has upstream services that depend on it and downstream services that it depends on.

Resolving the incident that we're working on depends on understanding its full scope. Tools like **Lightstep Incident Response** automatically show us a high-level view of how many other services our incident is impacting. If the problem is limited to the service triggering the alert responsible for our incident, we can notify our response team and begin working on remediation immediately.

If, however, we see that the problem has started to cascade to other services, we may want to increase our incident's severity level, sound the alarm, and start bringing in other SREs and response teams to help manage the situation.

On the other hand, we might find that a problem in an upstream service (that our service depends on) causes our incident. In that case, after we've verified that the team responsible for the upstream service is working on a solution, we may have to "hurry up and wait" until the upstream incident team fixes the problem. Once the upstream service is working correctly, we can monitor our service to ensure it recovers gracefully.

Whatever the incident's ultimate cause, automated access to a high-level view of the incident's scope ensures we minimize incident resolution time because we don't over- or under-react.

Collaborating effectively

Although we usually serve as incident response quarterbacks, we rarely solve incidents on our own. Anything small enough for an SRE to fix in a minute or two without assistance probably isn't large enough to qualify as an incident.

Incidents tend to be significant, business-impacting events, meaning we must pull in a team of incident responders immediately. We'd rather not do this manually because, at a minimum, we'll need to find a developer who's familiar with the service. We also want someone to handle communications from business stakeholders.

Automated on-call management helps us quickly spin up our incident response team by telling us exactly who is on-call to respond to an incident with the impacted service. We can then alert every team member by any means necessary — email, SMS message, voice calls — whatever it takes.

Effective incident response automation tools like Lightstep can handle these notifications for us so we can devote our time to reducing incident resolution time. We can even set up escalation policies, so if a member of the incident team fails to respond within a certain timeframe, Lightstep automatically notifies that responder's backup.

Once all responders are online, they need a way to communicate. Enabling inter-team comms shouldn't require any extra work: Lightstep can automatically create a Slack or Teams channel for the incident and invite all members of the response team. As with everything else we automate, communications automation leads to faster incident resolution because we're not wasting time on administrative busywork.

After we've pulled in an incident responder to handle communications with business stakeholders like product managers, VPs, and other executives, an ideal incident response automation tool should notify these stakeholders automatically, so they'll know we're aware of the incident and are working on a resolution. This notification is precisely what Lightstep and other incident management automation tooling do, ensuring that SREs are free to spend more time on incident resolution and less time taking flak from stakeholders.

After we've finished analyzing data and taken steps to resolve the incident, we can use automated tools to let all stakeholders know we resolved the incident and everything is back to normal.

Painless postmortems

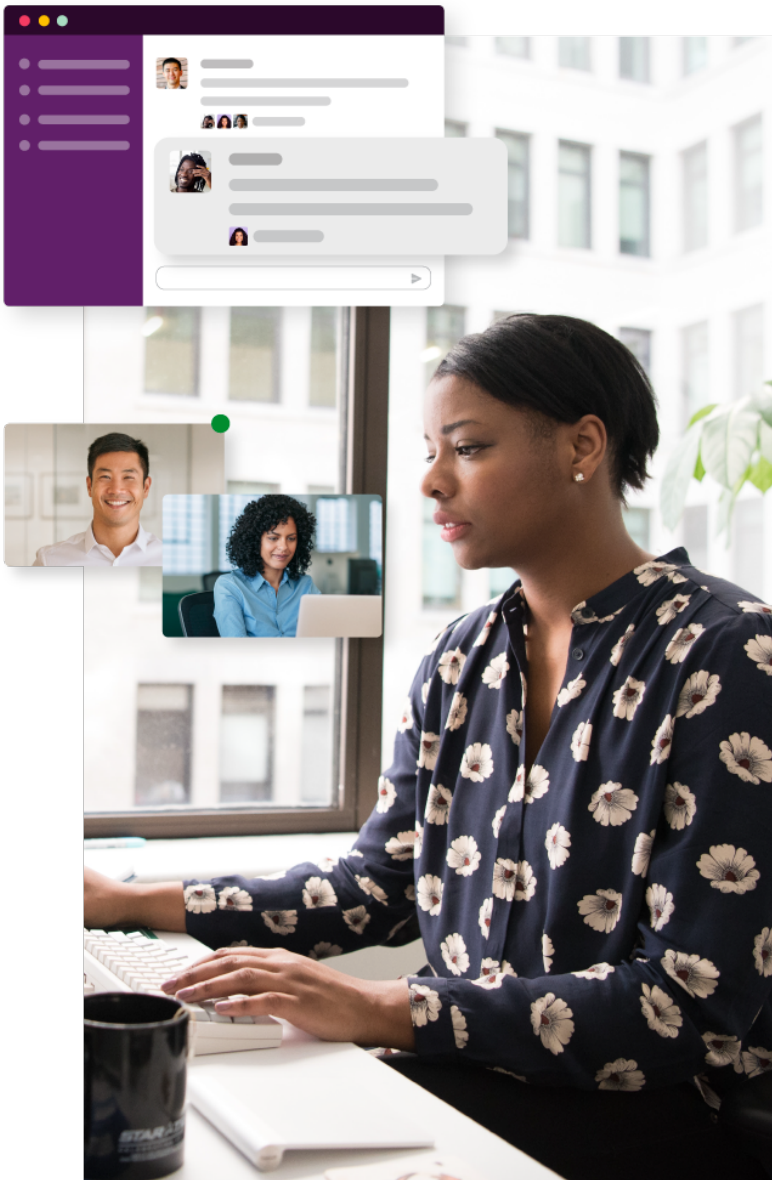
Although postmortems can be painful, good SRE teams don't shy away from them. And with a bit of help from automation, postmortems don't need to be painful at all.

Let's consider all the data we'll need to construct an incident timeline for our postmortem. An ideal incident response platform should automatically compile timeline data. Fortunately, that's precisely what Lightstep does. This automatic compilation frees up our time so we can devote it to the most valuable aspects of the postmortem.

With data gathering and compilation off our plate, we can apply our SRE knowledge and experience to determine the incident's overall impact. Then, we can document what we learned as a team and develop action items the organization should follow to prevent a recurrence of the incident.

Automation also helps us fulfill another fundamental tenet of good SRE practice: blameless postmortems. It's easy to avoid blatantly biased postmortems that say things like, "the outage is Joe's fault because he deleted all data from the production database." It's more challenging, though, to avoid subtleties like confirmation bias — a bias that makes us more likely to notice data that supports what we already thought the incident's root causes were and less likely to notice data that invalidates our hypothesis.

By using Lightstep to provide a clear, unambiguous record of events and timelines throughout an incident's lifecycle, we don't need to rely on memory or worry about unconscious bias. In relying on our incident response platform to attach all relevant data to the postmortem, we can focus on understanding what went wrong to ensure we produce clear, blame-free postmortems that educate our SRE peers.



Next Steps

We know from experience that incident management is no walk in the park — and we're okay with that. As SREs, we're proud of the work we do to keep apps and services running smoothly.

Without a doubt, incident management can be complex. We've seen many parts of the incident resolution process where SREs can spin their wheels on toil instead of staying laser-focused on problem-solving.

But there's no need to make incidents more complicated than necessary! Automation — particularly the sophisticated automation we get from Lightstep Incident Response— lets us skip low-value activities to focus our time and talent on solving incidents as quickly as possible.

If you're ready to see what automation can do for your SRE workflow, why not give Lightstep Incident Response a try?

Lightstep® Incident Response

See it for yourself

lightstep.com/incident-response/

Contact us | Follow us