Lightstep® Incident Response

How Preparation Makes All the Difference in Incident Management

Table of Contents

How Preparation Makes All the Difference in Incident Management

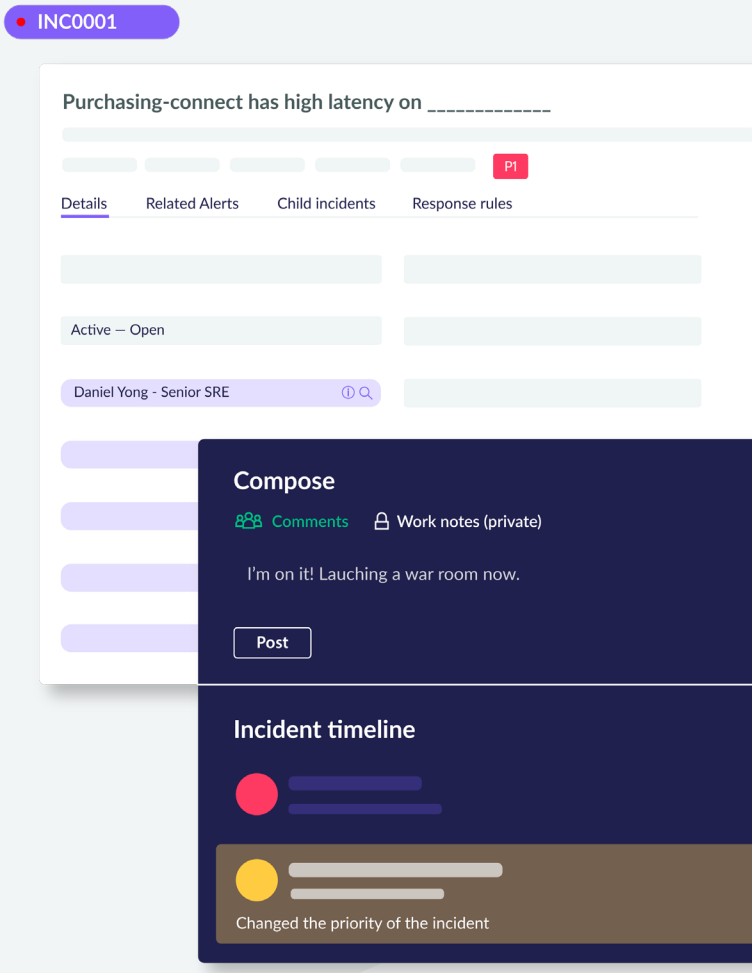
Modern enterprises depend on DevOps and site reliability engineers (SREs) to keep apps and services online. Our jobs are challenging but fun. Suppose an incident occurs while on-call— we must drop what we're doing and act immediately, even if it's 2 AM and we're fast asleep. Then, we must work furiously to fix the problem while knowing that our colleagues, managers, and customers are watching and waiting. This adds to the stress.

Even though by definition the idea of incident response is reactive, the role of SRE has matured and it has become clear that we can be proactive in many ways.

With proper preparation we can get ahead of the game.

Planning and preparing in advance might seem unusual, painful, or possibly a waste of time. But when done right, preparation of various forms *reduces* pain by helping us respond to and resolve incidents more quickly. And the more quickly we solve incidents, the less stressful they are.

In this eBook, we explore five activities that can take the pain out of incident response when combined with intelligent incident management tools.



Preparing Your Incident Response

Incident response shouldn't be a bespoke operation. While no two incidents are identical, we can do plenty of work up-front to ensure we're not starting from scratch every time an incident occurs. Preparation in several vital areas helps us provide quick incident resolution.

We can start by **coordinating the right incident response team**. Our incident response is only as strong as our incident responders.

Next, we can ensure our incident responders have access to **detailed documentation and runbooks**.

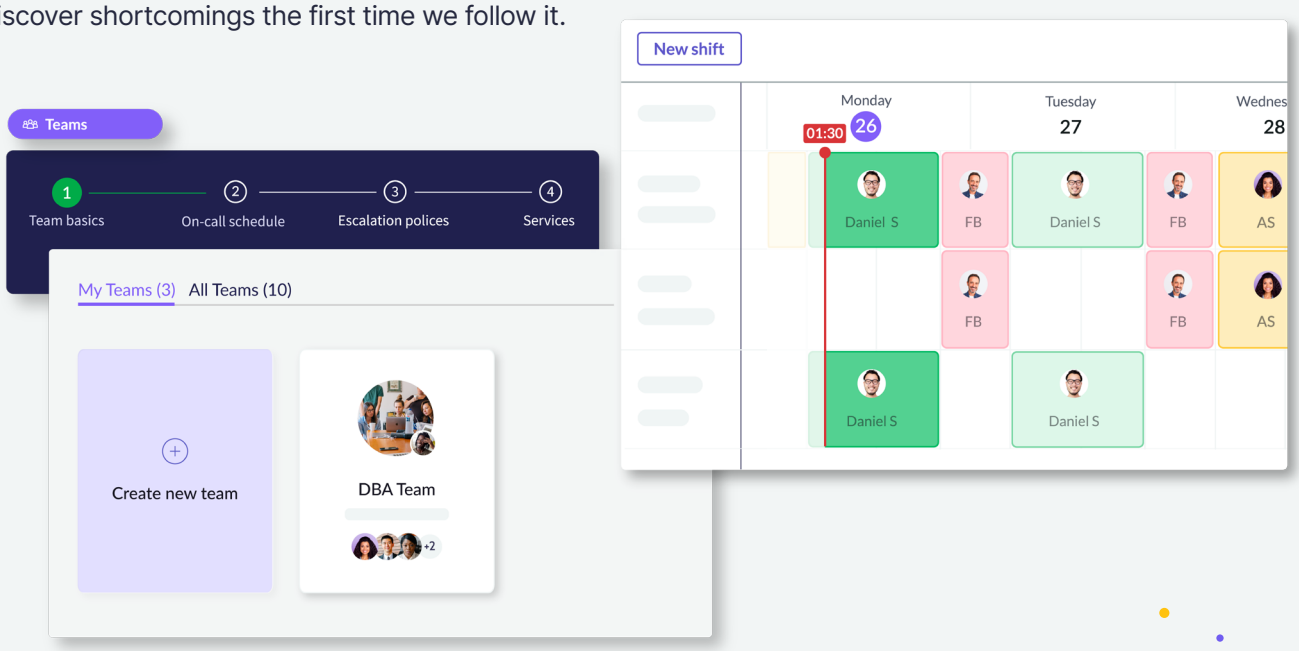
Then, we should **define a shared incident management vocabulary**. We don't want to waste time during an incident resolving misunderstandings. Discipline about naming conventions can be a crucial time saver and enabler of automation both during the incident and in postmortems.

Following these three steps will leave us with a solid incident response plan. But a plan does us no good if we discover shortcomings the first time we follow it.

That's why our **incident response preparation should include regular field tests of our response plan**.

As the infrastructure, apps, and services we monitor grow and change over time, so must our incident response preparation. **Conducting effective incident postmortems** can provide us the data needed to keep our incident response plans up to date.

Let's look at each of these five activities in greater depth. Along the way, we examine how incident response automation tools can enhance our incident response preparation.



Coordinate the Incident Response Team

To resolve incidents quickly, we must identify and notify the right people at each step of the response lifecycle — and ensure each incident responder has the data they need to complete their work.

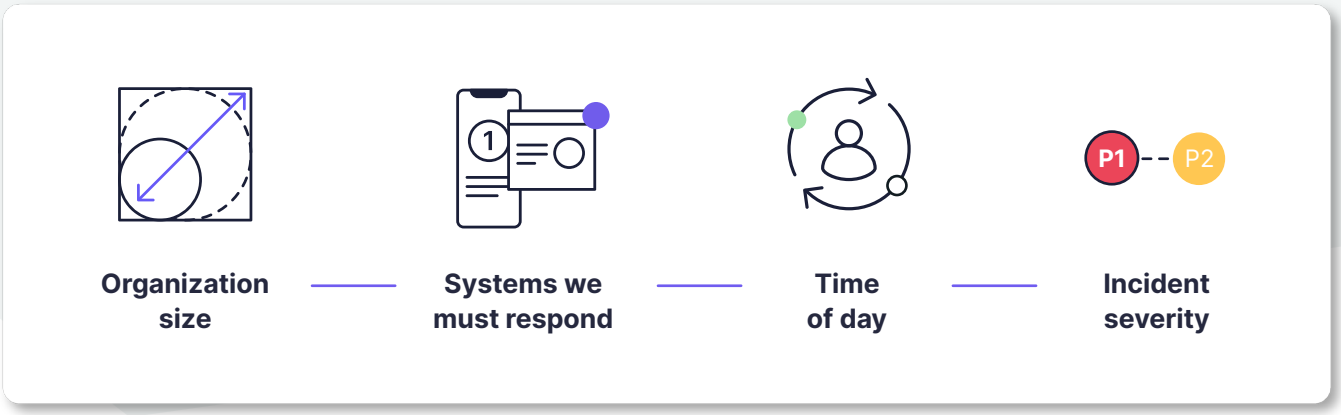
The incident response typically starts when our monitoring tools detect an anomaly and alert the SRE team. In a large organization, we likely have multiple levels of monitoring and observability; perhaps an application performance monitoring (APM) tool to monitor application performance, Prometheus monitoring our Kubernetes clusters, and OpenTelemetry recording distributed traces.

A simple error or outage might trigger alerts in one of our monitoring tools. More severe errors and outages might trigger alerts in *all* our monitoring tools. It's challenging to coordinate incident responders with fragmented data about the incident in dozens of alerts stored in different places by different monitoring tools.

So, the first step toward coordinating our team should be coordinating and correlating our data. That's where an incident management automation platform like Lightstep Incident Response can help. A good incident management platform connects to all our monitoring and observability tools. The platform will automatically correlate alerts from all sources and present them as a unified incident dashboard when an incident occurs.

A unified incident dashboard does us no good, however, unless the right people see it. Choosing the correct first responder to an incident can depend on many things, such as:

- **Organization size —** large organizations likely have many DevOps and SRE teams and only a subset of personnel on-call at a given time. A startup may have a single harrowed team responsible for all DevOps and SRE tasks.
- **To which systems we must respond —** We can't expect every SRE to understand all our apps and infrastructure other than with startups.
- **Time of day —** Keeping DevOps and SRE teams on-call 7/24 leads to burnout, so most organizations set on-call rotations. This ensures an even distribution of the workload. International enterprises may even employ SRE teams in different time zones, so there's always someone awake and working to deal with incidents as they occur.
- **Incident severity —** Mild to moderate incidents might require the attention of a single SRE. A complete outage of all services might require an all-hands-on-deck response, even if it means waking up your entire DevOps and development teams at two in the morning.



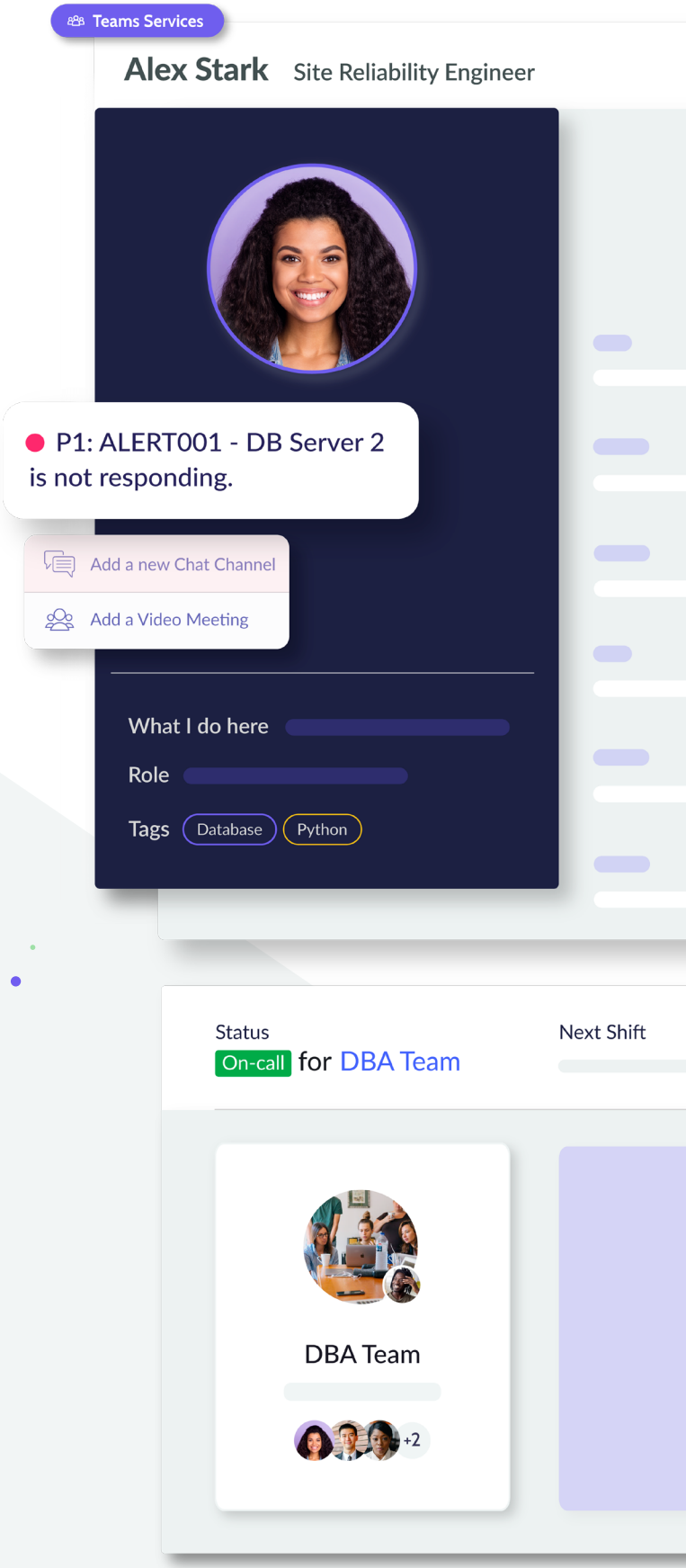
It isn't easy to decide who to notify with multiple factors impacting the decision. Fortunately, our incident automation platform can help us here, too. We can set up an on-call rotation of first responders based on time of day and area of responsibility. So, for example, if our company's billing system begins experiencing increased database read latency at 02:00 UTC on a Sunday, the platform can instantly decide who to notify.

The first responder SRE can sign in to the platform, view the unified incident dashboard, and perform triage. From there, they can drill down into the data sources and alerts that triggered the alert. If it's a false alarm, the SRE can close out the incident.

Otherwise, the SRE can escalate the incident and bring other responders online. Once again, our incident platform can help us coordinate the team. For example, if we use Lightstep, we can pull in additional on-call responders and stakeholders with a single click. We may need technical experts like developers and network administrators to help resolve the incident.

If the incident is severe enough, we may need additional responders to manage public relations and communications, and we must notify business stakeholders. We might even need an escalation path to the CEO and general counsel if the incident represents an existential threat to the company.

We can't expect to coordinate that many people manually. We don't want SREs to waste time looking up who's on call and then personally phoning each response team member to get their attention. Instead, we should lean on our incident management platform to help coordinate our team and manage escalation channels. It should automatically notify the right people at the right time and give them the means to communicate and coordinate in real time by creating a shared Slack or Teams channel.



Create Runbooks and Documentation

Coordinating our response team is a great start — but we should also be proactive in preparing documentation and runbooks that help incident responders act quickly and decisively.

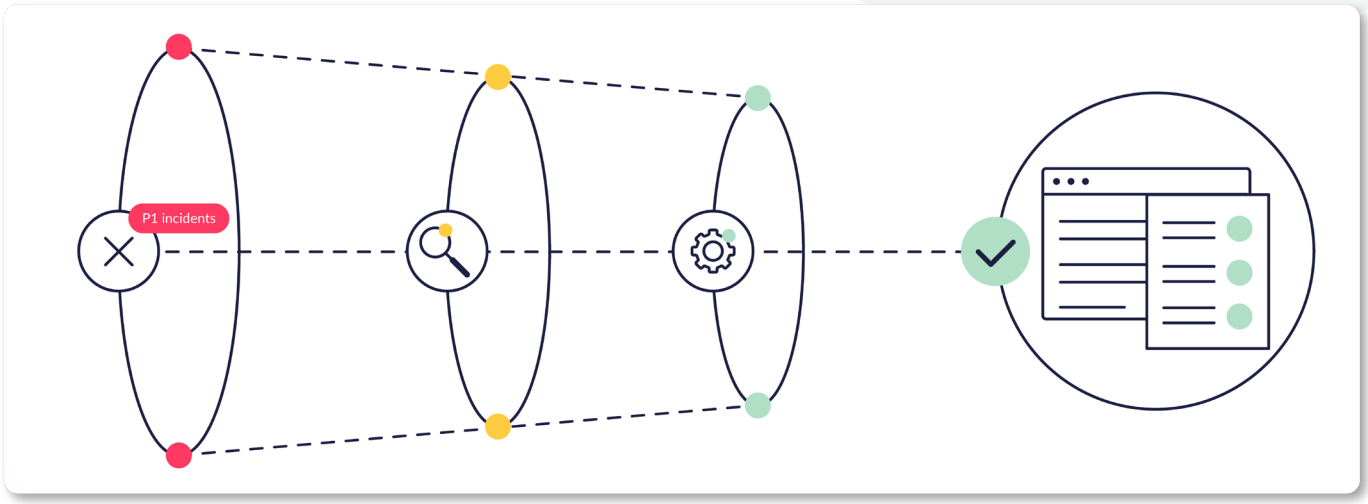
Runbooks for responders are like checklists for pilots on a plane. With a detailed list, the pilot is ready for critical problems during the flight. The pilot doesn't need to know the details about the function of each component to be prepared to handle a malfunction.

An incident response runbook is a collection of essential instructions composing a specific procedure for handling an issue. A runbook contains a series of steps an SRE can follow to diagnose and fix the problem, and each step might link out to *another* runbook. For example, the runbook for dealing with an outage of our company's authentication service might include instructions to restart the service. To complete that step, we have to complete all the steps in our authentication service restart runbook. Preparing runbooks in advance can help incident responders fix a problem quickly without waiting for the system owner to jump in and assist.

We can go a step further with runbook automation. Consider the authentication service restart runbook we just discussed. Most cloud platforms let us restart a VM or container using an API call, so we should automate the runbook and let incident responders trigger it by clicking a button. It's also good practice to create an automated "roll back to previous deploy" runbook. Poorly-tested changes cause many incidents that we push into production, so rollback should always be a one-click operation. Less time spent on manual runbook steps leads to quicker incident resolution.

If we trust our automated runbooks, we can even use them to make parts of our infrastructure self-healing. For example, our "Service A has stopped responding to requests" playbook might tell us to restart all service instances, send multiple HTTP requests to Service A's endpoint on the load balancer, and verify that it responds as expected.

We don't need an SRE in the loop to run that playbook. Our monitoring tools fire an alert if the service stops responding, and our incident management platform can run the automated playbook. If it works, the crisis is averted! We'll want to investigate the error, but it's no longer an incident requiring immediate resolution. If the automated playbook fails, *then* our automation can alert incident responders.



A comprehensive set of runbooks can help us resolve many incidents — but not *all* incidents. If we run through all our runbooks and haven't fixed the incident, we must dig deeper into the apps, services, and infrastructure involved. Fortunately, we can prepare for this, too. Development and DevOps teams should thoroughly document their work so incident responders can access all the information they might need.

From an SRE's perspective, good documentation tells us how to troubleshoot, reconfigure, and redeploy apps and services — or even reprovision infrastructure, if necessary. It should give us everything we need to develop an ad-hoc runbook on the fly because, effectively, that's what we're doing! During the incident postmortem, we can look back at the resolution steps we followed and use them as the basis for a *new* playbook incident responders can use if this type of incident occurs again.

Define Your Incident Management Vocabulary

To make runbooks and documentation effective, we should avoid excessive technical jargon. That doesn't mean we should dumb things down. We *should* include a level of technical detail appropriate to each type of incident responder, but we should do so as simply and clearly as possible. Remember that incident responders are working under pressure. Unnecessary jargon slows comprehension and can make incidents tougher to resolve.

Wherever possible, use a clearly defined vocabulary and naming conventions for all teams. This might mean you come up with clear definitions for terms like "application," "service," "error," and "outage." These might seem simple, but teams that confuse apps and services or errors and outages in the heat of an incident might take the wrong steps to solve the problem. When an incident is named, it should be as self-explanatory and information rich as possible.

To avoid confusion, organizations can define a glossary of terms used in incident management. The clearly defined vocabulary is helpful for several reasons. First, teams can save time if they can easily find a definition with a linked explanation. This increases clarity and helps team members to avoid any misunderstanding. Second, a shared vocabulary allows all team members to use the documentation and runbooks easily as their expertise and experience levels can vary.

Finally, the vocabulary can also include acronyms and terms used by the engineers involved in incident management. Although we want to avoid jargon if we can, there might be specific terms and acronyms that are meaningful in the context of our apps. Including these terms in our glossary lets us use them in our runbooks if we must.

Field Test Your Incident Response Plan

As the old saying goes, no plan survives first contact with the enemy. For SREs, incidents are the enemy — and regularly field testing our incident response plan gives us confidence the plan will work when we're actually under fire.

We can start by conducting incident fire drills. These work in much the same way as real fire drills. We schedule them in advance, and participants know about them, so they don't come as a surprise. There's no *actual* fire. We pick a specific incident scenario, and everyone involved *pretends* there's a fire and acts accordingly.

Incident fire drills aren't perfect simulations of actual incidents but help confirm that responders follow procedures and runbooks. We can simulate different types of incidents to ensure that our personnel and automated systems move through the stages of our incident response plan correctly in a variety of scenarios.

Fire drills are helpful but inherently limited because they are fake emergencies. No systems are down, heading toward failure, or even under stress.

We can go a step further by triggering real failures in a non-production test environment. The apps and services running in the test environment should be identical to those in production, with the only difference being they aren't serving customer workloads.

Real failures let SREs and other incident responders verify that the steps in both manual and automated runbooks work as expected. After all, there's not much point in incident responders following runbooks diligently if the runbooks themselves are wrong. And if we find we lack the documentation needed to resolve the incident, we can make sure it gets written so it is there when responders need it.

Going further, we might consider chaos engineering as the ultimate field test of our incident response plans. As SREs, we strive to build resilient, fault-tolerant systems. Chaos engineering puts that resilience and fault tolerance to the test by deliberately and unexpectedly inducing failure into our systems *in production*.

Failure might come in the form of increased network latency or even a network outage. It might be app and service instances shutting down or performing poorly. Or it might involve exhausting a resource like memory or disk space.

Chaos engineering isn't a field test of our incident plan per se. It's a test of system resilience. But unless we're too timid in testing our system's stability, chaos engineering will inevitably test our incident response plans for us. We shouldn't *start* with chaos engineering but should consider it as a goal to aim for once we're reasonably confident in the resilience of our systems and the effectiveness of our incident response plans.

Conduct Effective Postmortems

There's rarely a single root cause when complex issues arise. As a result, situations occur that may never occur again. But that doesn't make it any less important to track, diagnose, and understand problems.

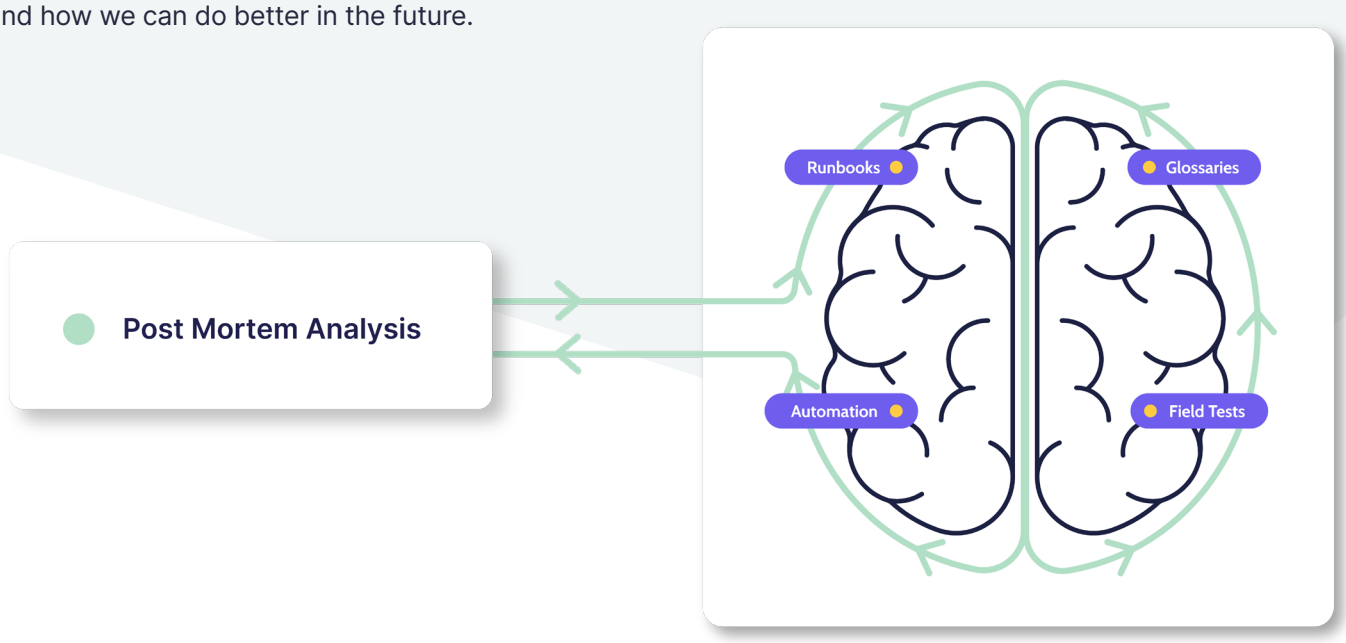
After identifying and fixing each problem, we should prepare detailed postmortem reports as SREs and incident responders. Good postmortem reports help us better understand the root causes of the incident so we can, if possible, prevent it from happening again. At the very least, we can use what we learn to improve our incident response plan.

Detailed records describing the issue and the steps taken to fix the problem can help us respond more effectively in the future. An automated incident management platform like Lightstep can help us by gathering all the data we need in one place, including information about when the incident began, how quickly responders received notice, and how quickly they responded. It should also attach the correlated logs and monitoring data from the incident so we can re-analyze them as we draw our postmortem conclusions. Using this data, we can ensure our postmortem reports contain a thorough analysis of what went right, what went wrong, and how we can do better in the future.

Postmortem reports should also be relevant to non-SRE stakeholders. We should write stakeholders' reports at their level of technical understanding, and the reports should contain the information necessary to understand the incident and the significant steps to the solution. This enables architects and business decision-makers to prepare for the future.

For instance, an architect can revisit the system infrastructure assumptions and consider shifting on-premises services to the cloud for greater resilience. Similarly, business stakeholders should be able to use the postmortem report to better understand what happened so that they can manage the expectations of executives and customers should a similar incident occur in the future.

We should ultimately look at postmortems as a critical part of incident preparation because they prepare us to effectively solve future incidents.



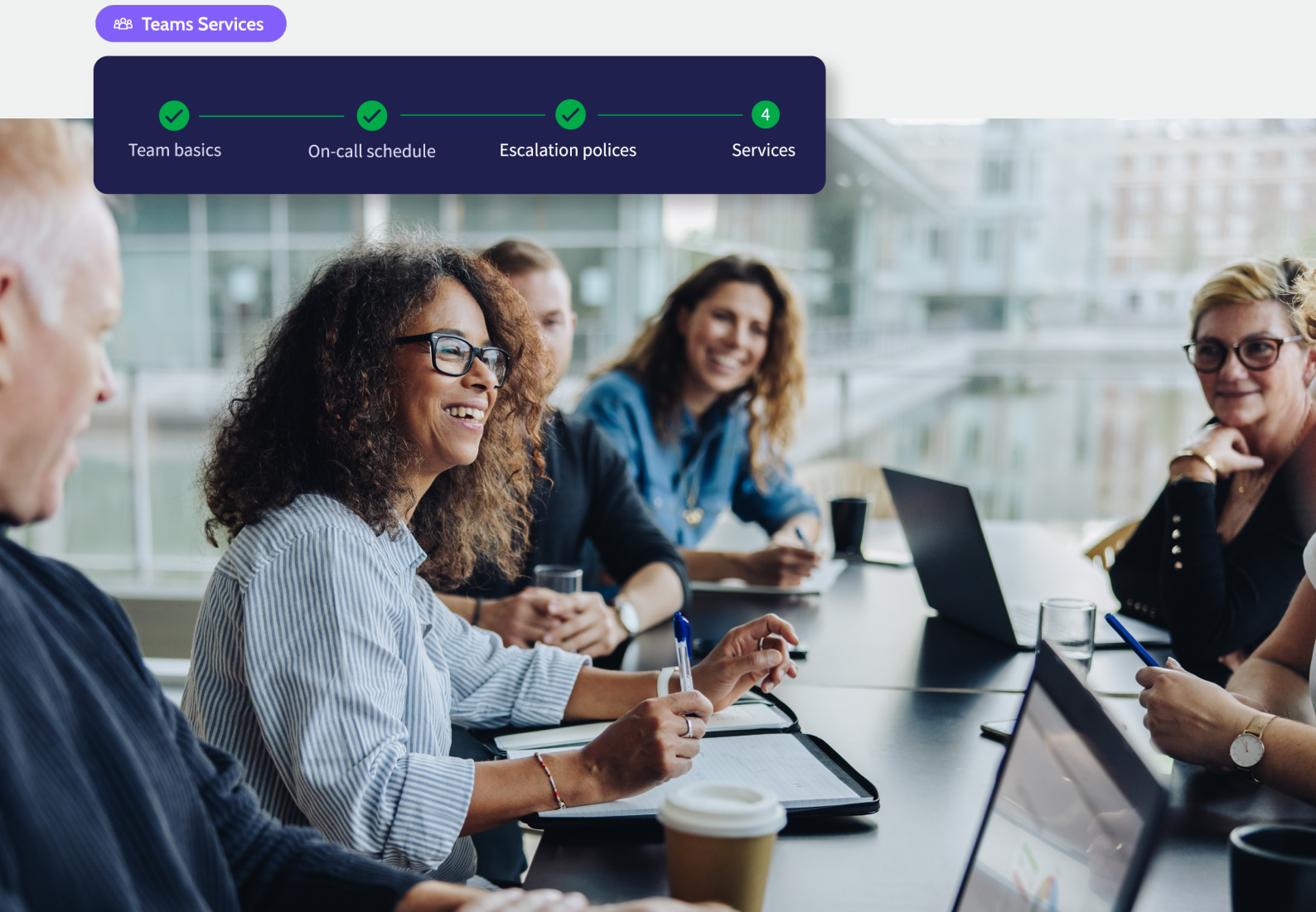
Summary

Modern incident management is no walk in the park. If most of our systems are traditional monolithic apps, incidents are tough enough for SREs and other incident responders to manage.

Incident management is even more challenging in modern cloud-native systems made up of dozens, hundreds, or thousands of interconnected apps and microservices. There are more ways for things to go wrong and a lot more data to sift through when an incident occurs.

We can manage this complexity through good preparation. Preparing for incidents in advance means we're not making things up on the fly when incidents occur. And good practice includes automation, using an incident management platform like Lightstep. Automation ensures we solve incidents quickly by letting responders focus their talent on solving the problem instead of wasting time on mechanical busywork.

In the end, incident management preparation is about empowering talented people — whether SREs, engineers, or other incident responders — by giving them the data, the confidence, and the tools they need to resolve incidents quickly.



Lightstep® Incident Response

See it for yourself

lightstep.com/incident-response/

Contact us | Follow us