Chapter 14 - Managing Incidents

Managing Incidents

Written by Andrew Stribblehill[78]
Edited by Kavita Guliani

Effective incident management is key to limiting the disruption caused by an incident and restoring normal business operations as quickly as possible. If you haven't gamed out your response to potential incidents in advance, principled incident management can go out the window in real-life situations.

This chapter walks through a portrait of an incident that spirals out of control due to ad hoc incident management practices, outlines a well-managed approach to the incident, and reviews how the same incident might have played out if handled with well-functioning incident management.

Unmanaged Incidents

Put yourself in the shoes of Mary, the on-call engineer for The Firm. It's 2 p.m. on a Thursday afternoon and your pager has just exploded. Black-box monitoring tells you that your service has stopped serving *any* traffic in an entire datacenter. With a sigh, you put down your coffee and set about the job of fixing it. A few minutes into the task, another alert tells you that a second datacenter has stopped serving. Then the third out of your five datacenters fails. To exacerbate the situation, there is more traffic than the remaining datacenters can handle, so they start to overload. Before you know it, the service is overloaded and unable to serve any requests.

You stare at the logs for what seems like an eternity. Thousands of lines of logging suggest there's an error in one of the recently updated modules, so you decide to revert the servers to the previous release. When you see that the rollback hasn't helped, you call Josephine, who wrote most of the code for the now-hemorrhaging service. Reminding you that it's 3:30 a.m. in her time zone, she blearily agrees to log in and take a look. Your colleagues Sabrina and Robin start poking around from their own terminals. "Just looking," they tell you.

Now one of the suits has phoned your boss and is angrily demanding to know why he wasn't informed about the "total meltdown of this business-critical service." Independently, the vice presidents are nagging you for an ETA, repeatedly asking you, "How could this possibly have happened?" You would sympathize, but doing so would require cognitive effort that you are holding in reserve for your job. The VPs call on their prior engineering experience and make irrelevant but hard-to-refute comments like, "Increase the page size!"

Time passes; the two remaining datacenters fail completely. Unbeknown to you, sleep-addled Josephine called Malcolm. He had a brainwave: something about CPU affinity. He felt certain that he could optimize the remaining server processes if he could just deploy this one simple change to the production environment, so he did so. Within seconds, the servers restarted, picking up the change. And then died.

The Anatomy of an Unmanaged Incident

Note that everybody in the preceding scenario was doing their job, as they saw it. How could things go so wrong? A few common hazards caused this incident to spiral out of control.

Sharp Focus on the Technical Problem

We tend to hire people like Mary for their technical prowess. So it's not surprising that she was busy

making operational changes to the system, trying valiantly to solve the problem. She wasn't in a position to think about the bigger picture of how to mitigate the problem because the technical task at hand was overwhelming.

Poor Communication

For the same reason, Mary was far too busy to communicate clearly. Nobody knew what actions their coworkers were taking. Business leaders were angry, customers were frustrated, and other engineers who could have lent a hand in debugging or fixing the issue weren't used effectively.

Freelancing

Malcolm was making changes to the system with the best of intentions. However, he didn't coordinate with his coworkers—not even Mary, who was technically in charge of troubleshooting. His changes made a bad situation far worse.

Elements of Incident Management Process

Incident management skills and practices exist to channel the energies of enthusiastic individuals. Google's incident management system is based on the Incident Command System,[79] which is known for its clarity and scalability.

A well-designed incident management process has the following features.

Recursive Separation of Responsibilities

It's important to make sure that everybody involved in the incident knows their role and doesn't stray onto someone else's turf. Somewhat counterintuitively, a clear separation of responsibilities allows individuals more autonomy than they might otherwise have, since they need not second-guess their colleagues.

If the load on a given member becomes excessive, that person needs to ask the planning lead for more staff. They should then delegate work to others, a task that might entail creating subincidents. Alternatively, a role leader might delegate system components to colleagues, who report high-level information back up to the leaders.

Several distinct roles should be delegated to particular individuals:

Incident Command

> The incident commander holds the high-level state about the incident. They structure the incident response task force, assigning responsibilities according to need and priority. *De facto*, the commander holds all positions that they have not delegated. If appropriate, they can remove roadblocks that prevent Ops from working most effectively.

Operational Work

> The Ops lead works with the incident commander to respond to the incident by applying operational tools to the task at hand. The operations team should be the only group modifying the system during an incident.

Communication

> This person is the public face of the incident response task force. Their duties most definitely

This person is the public face of the incident response task force. Their duties most definitely include issuing periodic updates to the incident response team and stakeholders (usually via email), and may extend to tasks such as keeping the incident document accurate and up to date.

Planning

> The planning role supports Ops by dealing with longer-term issues, such as filing bugs, ordering dinner, arranging handoffs, and tracking how the system has diverged from the norm so it can be reverted once the incident is resolved.

A Recognized Command Post

Interested parties need to understand where they can interact with the incident commander. In many situations, locating the incident task force members into a central designated "War Room" is appropriate. Others teams may prefer to work at their desks, keeping alert to incident updates via email and IRC.

Google has found IRC to be a huge boon in incident response. IRC is very reliable and can be used as a log of communications about this event, and such a record is invaluable in keeping detailed state changes in mind. We've also written bots that log incident-related traffic (which is helpful for postmortem analysis), and other bots that log events such as alerts to the channel. IRC is also a convenient medium over which geographically distributed teams can coordinate.

Live Incident State Document

The incident commander's most important responsibility is to keep a living incident document. This can live in a wiki, but should ideally be editable by several people concurrently. Most of our teams use Google Docs, though Google Docs SRE use Google Sites: after all, depending on the software you are trying to fix as part of your incident management system is unlikely to end well.

See Example Incident State Document for a sample incident document. This living doc can be messy, but must be functional. Using a template makes generating this documentation easier, and keeping the most important information at the top makes it more usable. Retain this documentation for postmortem analysis and, if necessary, meta analysis.

Clear, Live Handoff

It's essential that the post of incident commander be clearly handed off at the end of the working day. If you're handing off command to someone at another location, you can simply and safely update the new incident commander over the phone or a video call. Once the new incident commander is fully apprised, the outgoing commander should be explicit in their handoff, specifically stating, "You're now the incident commander, okay?", and should not leave the call until receiving firm acknowledgment of handoff. The handoff should be communicated to others working on the incident so that it's clear who is leading the incident management efforts at all times.

A Managed Incident

Now let's examine how this incident might have played out if it were handled using principles of incident management.

It's 2 p.m., and Mary is into her third coffee of the day. The pager's harsh tone surprises her, and she gulps the drink down. Problem: a datacenter has stopped serving traffic. She starts to investigate. Shortly another alert fires, and the second datacenter out of five is out of order. Because this is a rapidly growing

issue, she knows that she'll benefit from the structure of her incident management framework.

Mary snags Sabrina. "Can you take command?" Nodding her agreement, Sabrina quickly gets a rundown of what's occurred thus far from Mary. She captures these details in an email that she sends to a prearranged mailing list. Sabrina recognizes that she can't yet scope the impact of the incident, so she asks for Mary's assessment. Mary responds, "Users have yet to be impacted; let's just hope we don't lose a third datacenter." Sabrina records Mary's response in a live incident document.

When the third alert fires, Sabrina sees the alert among the debugging chatter on IRC and quickly follows up to the email thread with an update. The thread keeps VPs abreast of the high-level status without bogging them down in minutiae. Sabrina asks an external communications representative to start drafting user messaging. She then follows up with Mary to see if they should contact the developer on-call (currently Josephine). Receiving Mary's approval, Sabrina loops in Josephine.

By the time Josephine logs in, Robin has already volunteered to help out. Sabrina reminds both Robin and Josephine that they are to prioritize any tasks delegated to them by Mary, and that they must keep Mary informed of any additional actions they take. Robin and Josephine quickly familiarize themselves with the current situation by reading the incident document.

By now, Mary has tried the old binary release and found it wanting: she mutters this to Robin, who updates IRC to say that this attempted fix didn't work. Sabrina pastes this update into the live incident management document.

At 5 p.m., Sabrina starts finding replacement staff to take on the incident, as she and her colleagues are about to go home. She updates the incident document. A brief phone conference takes place at 5:45 so everyone is aware of the current situation. At 6 p.m., they hand off their responsibilities to their colleagues in the sister office.

Mary returns to work the following morning to find that her transatlantic colleagues have assumed responsibility for the bug, mitigated the problem, closed the incident, and started work on the postmortem. Problem solved, she brews some fresh coffee and settles down to plan structural improvements so problems of this category don't afflict the team again.

When to Declare an Incident

It is better to declare an incident early and then find a simple fix and close out the incident than to have to spin up the incident management framework hours into a burgeoning problem. Set clear conditions for declaring an incident. My team follows these broad guidelines—if any of the following is true, the event is an incident:

- Do you need to involve a second team in fixing the problem?
- Is the outage visible to customers?
- Is the issue unsolved even after an hour's concentrated analysis?

Incident management proficiency atrophies quickly when it's not in constant use. So how can engineers keep their incident management skills up to date—handle more incidents? Fortunately, the incident management framework can apply to other operational changes that need to span time zones and/or teams. If you use the framework frequently as a regular part of your change management procedures, you can easily follow this framework when an actual incident occurs. If your organization performs disaster-recovery testing (you should, it's *fun*: see [Kri12]), incident management should be part of that testing process. We often role-play the response to an on-call issue that has already been solved, perhaps by colleagues in another location, to further familiarize ourselves with incident management.

In Summary

We've found that by formulating an incident management strategy in advance, structuring this plan to scale smoothly, and regularly putting the plan to use, we were able to reduce our mean time to recovery and provide staff a less stressful way to work on emergent problems. Any organization concerned with reliability would benefit from pursuing a similar strategy.

Best Practices for Incident Management

Prioritize. Stop the bleeding, restore service, and preserve the evidence for root-causing.

Prepare. Develop and document your incident management procedures in advance, in consultation with incident participants.

Trust. Give full autonomy within the assigned role to all incident participants.

Introspect. Pay attention to your emotional state while responding to an incident. If you start to feel panicky or overwhelmed, solicit more support.

Consider alternatives. Periodically consider your options and re-evaluate whether it still makes sense to continue what you're doing or whether you should be taking another tack in incident response.

Practice. Use the process routinely so it becomes second nature.

Change it around. Were you incident commander last time? Take on a different role this time. Encourage every team member to acquire familiarity with each role.

[78]An earlier version of this chapter appeared as an article in *;login:* (April 2015, vol. 40, no. 2).

[79]See *http://www.fema.gov/national-incident-management-system* for further details.

previous

Chapter 13 - Emergency Response

next

Chapter 15 - Postmortem Culture: Learning from Failure