BLAMELESS

The Blameless Complete Guide to Incident Management

PART 2

Preparing for incidents

Welcome or welcome back to **The Complete Guide to Incident Management**. In Part 1 of the Incident Management eBook, we looked at how to be in charge of an incident when you don't have a lot of resources prepared. Hopefully you don't have to be in that situation too often: building resources for incidents is one of the best investments you can make.

In Part 2, let's work through how to build all the pillars of an incident response plan. We'll look at each aspect in the order of how it comes into play. However, this shouldn't be a sequential guide; rather, you should evaluate what tools will help you most effectively.

Classifying incidents

Incident classification is a standardized way of organizing incidents with established categories. This is what happens first after an incident is detected, and determines the rest of the incident response policy.

Incidents can include outages caused by errors in code, hardware failures, resource deficits — anything that disrupts normal operations. Each new incident should fit into a category dependent on the areas of the service affected, and in a ranking of the severity of the incident. Each of these classifications should have an established response procedure associated with it.

Having a robust classification system is beneficial for many reasons:

- → Improves triage by ensuring you respond to the most critical incidents first
- → Determines who should be alerted and what roles they should play in resolution
- → Helps with consistent responses, saving time and toil, and reducing confusion about how other people will proceed
- → Measures the expected impact of incidents by type for longer term planning
- → Identifies patterns in incident occurrences to prioritize preemptive fixes

Classifying incidents is important to building useful on-call policies. If you're finding that engineers are being alerted too often or subject matter experts aren't brought in when appropriate, you should probably prioritize building a classification system

1. Determine what types of responses are required

Ultimately, you want each classification of incident to map to a particular response, such that knowing the classification is enough to know what process will need to be implemented. Working backwards, a good way to come up with classifications is to think about what different responses could be necessary.

Think about who would need to be contacted for incidents in different service areas. These lines of ownership can help establish categories of incidents. These can be broken down into subsections by considering when different resources and procedures, like runbooks, should be utilized.

After you have categories of service areas, create tiers of how severely those areas could be impacted. When would you need to escalate a response, contacting more people and deploying more resources?

This impact should be based on how much customers are negatively affected. For example, if you're looking at a category of incidents around pages loading, you could end up with these tiers of severity:

Severity	Situation	Customer Impact	Response
Severity 1	Pages totally failing to load	Service unusable to customers, SLA violations	All hands on deck
Severity 2	Pages loading 200% slower	Service extremely tedious to use, customer retention threatened	Senior engineering teams and management alerted
Severity 3	Pages loading 50% slower	Service annoying to use, customers complaining	Senior engineering teams alerted
Severity 4	Pages loading 10% slower	Service usage not impacted to the extent customers complain, but could indicate further issues	Relevant engineering teams alerted
Severity 5	Pages loading 1% slower	Unnoticeable to customers	Incident logged into ticketing systems, but no immediate escalation or alerting necessary

Of course, the number of severity levels and where you delineate them will depend on your unique needs.

The Agero team has been able to simplify the way they classify incidents. Where before they distinguished 5 incident severity levels, now they only leverage 4.

Ethan Cohen, Sr. Manager, SRE

2. Determine metrics to classify incidents

Once you have a matrix of categories of impact and tiers of severities, it's important to have clearly defined metrics for reliably classifying new incidents. These should be objective so any team member will classify the incident in the same way.

Where possible, use metrics that can be automatically monitored to trigger alerts. As classification of the incident may require more analysis, you shouldn't aspire for fully automated classification. Instead, these metrics help alert people to the existence of the incident so they can further classify it.

3. Integrate your classifications into your incident response system

Now that you have your classifications and metrics to place new incidents within them, it's time to integrate them into a larger incident response system. Set your alerting tools to notify team members based on the incident's classification. Codify your runbooks and playbooks based on which classes of incidents they apply to. You may want to have automated runbooks start to execute in response to a new incident's classification, allowing for a fast and consistent response.

Many steps of your incident response process can be determined by the incident classification, reducing the toil of making these choices in the heat of the moment. Response systems like checklists, assigned roles for responders, and communication channels can be created based on the classification.

4. Review, learn, and revise

The golden rule of reliability is that failure is inevitable, so plan for your classifications to need ongoing revision. Review incident retrospectives in the context of the classification system. Look for areas of ambiguity where incidents could fall into multiple categories—could there be clearer rules to sort them? Look for incidents where a response differed from what was dictated by the classification—was it misclassified? Or should the response runbook for that classification change?

There will be incidents that don't fall cleanly into any established category. Trying to determine how to classify these novel incidents can slow you down during responses. Review these incidents and see how existing classifications can be expanded to include them or if new categories need to be made. Determine a set procedure for unknown incidents based on the categories they fall closest to so a response can begin without unnecessary hesitation.

Response roles and checklists

When responding to an incident, you'll likely be in a team with other people. To respond effectively, each person should have specific roles and responsibilities, expressed in a checklist. This will reduce redundant work and help each person start contributing right away.

Have roles with accompanying checklists set up ahead of time, along with policy for how they're assigned. These roles can be dynamic based on the size of the response needed, based on how the incident is classified. For small incidents someone might be handling multiple roles, whereas big "all-hands-on-deck" incidents could have multiple people working on the same types of tasks.

If you're finding that responders have a hard time getting started, or that certain tasks aren't getting done, implementing roles and checklists can help.

Here are some examples of roles and their responsibilities:

Incident commander

The incident commander runs the response process. They delegate other roles to the rest of the response team. As explained in this panel webinar on incident command, the incident commander role is something like a first aid responder. You shouldn't be in charge of absolutely everything, but you should be able to get the team started and make sure nothing is left unaccounted for.

Here's an example checklist for an incident commander:

1. Ensure that communication channels for the incident are available
2. Ensure the team is checked in
3. Assign roles to the rest of the team
4. Check in to make sure people are on track
   → But also make sure people are taking breaks and pacing themselves!
5. Help escalate when necessary
   → Handle the requests of management and other stakeholders

Learn more about incident command in our eBook!

Technical lead

The technical lead coordinates the engineering process of diagnosing and solving the technical issue itself. They make sure that redundant things aren't being tried by multiple people, and make sure resources like runbooks are available to people for people who need them. Depending on the number of people involved in the response, the technical lead could be a separate person, or one of the technical responders. This could also be referred to as the subject matter expert, the person most knowledgeable about the project area involved in the incident. However, the subject matter expert may take on other roles depending on what is needed of them.

Here's an example checklist for a technical lead:

1. Ensure each technical responder is working on something different
2. Provide each technical responder with any relevant resources
3. Escalate to subject matter experts if responders aren't sure of something
4. Make sure responders are in communication and sharing findings

Technical responder

Technical responders handle the main goal of the response process – diagnosing the problem and implementing a solution. Most people on the incident response team will be in this role, and other people will step into some of the responsibilities of this role when they're able to.

The primary goal of this role is to get the service back to working condition. This might not be the same as implementing a long-term solution that fixes the cause of the incident. However, the incident response process should ultimately encompass making systemic changes based on the incident, once the system is in working health.

Here's an example checklist for a technical responder:

1. Diagnose the incident by controlling different variables and looking for the limits of the incident
2. Work through appropriate runbooks
3. Where appropriate runbooks aren't available, debug code bases and diagnose infrastructure to look for more possible causes
4. Share results with other technical responders and brainstorm more ideas
5. Escalate results to the comms lead and incident commander to keep them updated on the status

Communications lead

As an incident progresses, various stakeholders need to stay in the loop. Stepping away from technical diagnosis to update people can break engineers' concentration and flow. Having someone assigned to manage communications can help. Tooling can help automate this process, making it easier and more consistent.

Here's an example checklist for a communications lead:

1. Monitoring developments from the rest of the team to react to updates that require communication
2. Updating customers affected by the incident with assurances and expected timelines
3. Updating upper management on how the situation is progressing and what resources are necessary
4. Relaying messages from stakeholders to the incident commander

Scribe

The scribe is responsible for documenting the incident response process. This will let you create retrospective documents, valuable tools for learning from incidents. Like communications, tooling can help automate this process, as all roles should be able to easily contribute without breaking focus.

Here's an example checklist for a scribe:

1. Make sure all communications are captured and logged in a permanently available area
2. Add screencaps of relevant error messages, monitoring data, and anything else that can provide context
3. Note timestamps of significant events, like escalations and status changes
4. Ask questions to clarify when there's ambiguity in communication

You won't get all of your roles and checklist perfectly the first time. There will be some ambiguity in who ought to handle what, and people might feel over- or under-worked. When you review past incidents, look at how people's roles functioned in the process.

Runbooks and other guiding tools

Runbooks serve as a main structure for technical incident response. They guide a responder through a series of diagnostic checks and steps to take based on their results. They can be imagined as a flowchart, where you navigate through each check and step based on the results of previous ones.

Runbooks are similar to playbooks, which generally refers to a guide for a more general process. Runbooks handle diagnosing a single hypothesis and implement specific solutions, whereas playbooks could cover the entire incident response process for a given role.

No matter what scale you're working with, proactively building runbooks, playbooks, and other guides is a valuable investment in being ready for incidents. Consider building guides for processes that have qualities like these:

- Processes that are often helpful for solving a variety of incidents
- Processes that require a lot of knowledge – e.g. they could return a wide variety of error messages, each of which needs to be dealt with differently
- Processes that are very simple and can be easily codified into consistent diagnoses and responses
- Processes that are essential when severe incidents occur

You want to make sure your most essential and common processes are covered with some sort of guide. When things go wrong, you can't expect people to respond with perfect sensibility and memory. People will naturally be panicking – guides can compensate for these challenges.

Look to past incidents to come up with runbooks. See what processes worked and then break them down into individual steps. This will make them modular, so you can build up other runbooks from them.

For example, here's a very simplified version of a runbook for diagnosing your login service going down:

1. Check to see if your servers or cloud provider is down.

   → If so, move to runbook for restarting servers or dealing with cloud outage.

2. Run the login code on a local testing environment.

   → Does it work? If not, move to debugging the code base.

   → If yes, move to debugging deployment issues.

3. Did you find an error in deployment or the code base? Try circumventing it (even just commenting it out temporarily) and see if you can log in.

4. Could you log in? Great! Test that fix to make sure it doesn't cause other issues and deploy it to get the login service online ASAP.

   → Then start investigating why it happened to make a more long-term solution

   → If not, loop back to debugging to see where other errors crop up.

This gives you the general idea of how a runbook moves you through a problem with an originally ambiguous issue, pinning down where the problem occurred and setting up a fast fix.

Automating runbooks

The next step to make runbooks even faster, easier, and more consistent is to automate them. Runbook automation eliminates toil by having steps run through software triggered by certain situations (such as detecting an outage of a specific service), minimizing the amount of input you need to provide. This requires tools to execute each step, as well as a tool to orchestrate the overall runbook and determine which steps are necessary. We'll cover investing in automation and tooling later in the eBook.

Tooling

So far in this eBook, and in Part 1, we've talked about how tooling can help your incident response process. You want the work around incident response to be as effortless and automatic as possible. That way, each incident creates a lot of data to study. Tooling can make this happen. Let's take a look at some specific types of tools and why you may consider investing in them.

Alerting tools

These tools, like PagerDuty, OpsGenie, or Prometheus, handle the actual alerting of on-call engineers when something goes wrong. You'll want to make sure your escalation policies can be represented in the framework of your alerting tool.

Observability tools

Observability or monitoring tools, like NewRelic, Sumo Logic, or Honeycomb, track metrics about your system, letting you know when things are abnormal. These tools are essential, as they allow you to detect incidents as quickly as possible.

Testing tools

An important way of staying ahead of incidents is to thoroughly test code before it reaches production, and to continue testing as further changes are made. These tools can make this process automatic. Automatic tooling can take the form of static analysis (looking at the code without running it), or running tests on code in production. Tools include Sentry, Code Climate, and Selenium.

Ticketing tools

Ticketing systems, like Jira, ServiceNow, or Asana, can help you track the larger tasks that emerge from incidents. You can use your incident response platform to automatically create tickets from incidents, which then can be integrated into larger sprints or other planning systems.

Integrating new tools into your response process is an investment – there's a learning curve to adopting them for engineers. To see if a tool would be a good fit, try our reliability tool checklist!

Incident response platforms

Perhaps the most important tool is a platform that can serve as a foundation for the rest of your processes. Our Blameless platform is a great example. It can handle the following processes to prepare for, resolve, and learn from incidents:

→ Automatically creating channels and video meetings for each incident
→ Automatically assigning roles and associated checklists for each responder
→ Automatically issuing preset communications to stakeholders at incident changes
→ Documenting runbooks to act as more elaborate guides for responders
→ Automatically creating a retrospective for each incident, logging activity in channels
→ Creating and tracking follow up tasks from retrospectives
→ Automatically tracking incident statistics and creating dashboards

Each of these features can make your incident response process smoother, with less toil and more consistent results. You'll have the info you need to be better prepared for incidents without needing a lot of manual overhead.

Having your response platform integrate with other incident response tools is important, too. You don't want to have to manually transfer data from one tool to another in the heat of an incident.

Your journey to incident excellence is underway!

No matter how good you get at preparing for them, incidents are still inevitable. With the information in this guide, you can start equipping yourself to deal with them more efficiently and learn more from each one. As you invest your time in implementing each process and tool, you'll find incidents resolving quicker, with less stress for engineers. Your perspective on your system health will become clearer, and you'll respond more appropriately to the true impact of each incident.

To learn more about the incident response process, explore our blog! Here are a few examples to check out:

→ What's difficult about incident command?

→ Incident Response Team

→ How to Become a Master at Incident Command

To see how Blameless can form the foundation for your incident management plan, check out a demo!

Blameless drives reliability across the entire software lifecycle by operationalizing Site Reliability Engineering (SRE) practices. Teams share a unified context during incident response, efficiently communicate, and resolve faster. Detailed Retrospectives give teams a data-driven approach to learn and Service Level Objectives (SLOs) inform teams where to prioritize work and innovate at velocity. Customers include brands such as Procore, Under Armour, Citrix, Mercari, Fox, and Home Depot who embrace a blameless culture, team resilience, and greater reliability to protect their customers.

Blameless is a 2021 Gartner Cool Vendor recipient and is backed by Accel, Lightspeed, Decibel and Third Point Ventures.

BLAMELESS