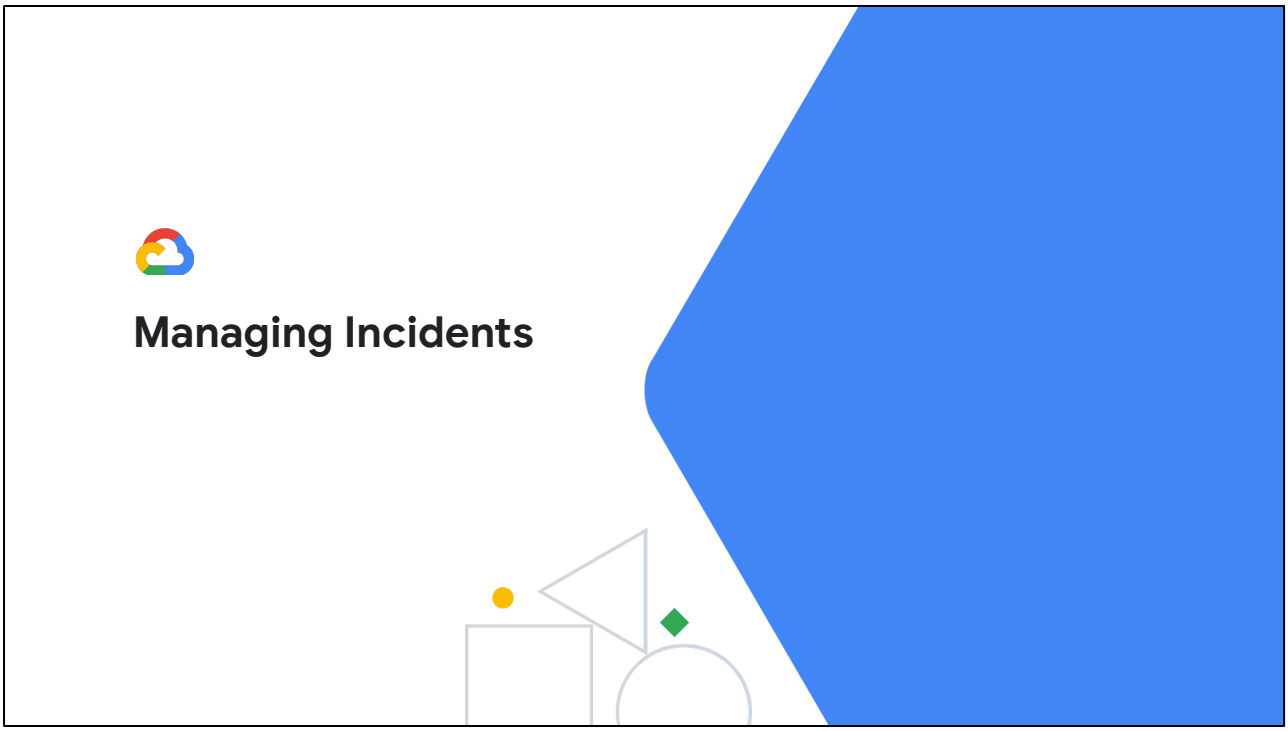


Up to this point in our course, we've mostly focused on ways to inspect and monitor the status of our systems running in Google Cloud.
But no matter how solid your planning, design, architecture, and preventive maintenance strategies are, things will go wrong.
When they do go wrong, how you manage those incidents will have a huge impact on user perception.

Agenda

Incident Management

Declaring an Incident

Restoring Service

Preventing Recurrence

Google Cloud

In this module, you will learn:

- How to handle incidents using a systematic process.
- How to define incident management roles and communication channels, so that
  - the team knows what their jobs are,
  - and the users are kept in the loop.
- How to see what, if anything, can be immediately done to mitigate the incident's impact.
- How to troubleshoot root causes of the incident so we can (hopefully) get a permanent fix in place.
- How to resolve the incident, or at least come up with a ready workaround.
- And finally how to document incidents as part of a standard post-mortem process.

Agenda

Incident Management

Declaring an Incident

Restoring Service

Preventing Recurrence

Google Cloud

Let’s get started.

What is Incident Response?

Alert

Incident

Google Cloud

Let's start with our terminology. What exactly is an incident? Is it just an Alert?

You're walking down the street, step off a curb, and your foot turns funny. Ouch! That's an alert. Pain is your body's way of saying, there might be a problem.

You stand a moment, on your good leg, and you tentatively move your ankle. Then you apply weight. Then, if that's all ok, you perhaps take a step. All the while you're thinking, "Is this bad enough to require me to do something to treat my ankle? Or can I safely continue on with normal operations." What you're doing right here, is making the decision: is it just an alert, or does an incident need to be declared.

An alert indicates that something bad is happening, or on track to happen. As we've discussed in this course, "something bad" is a generic way of saying that one of our Service Level Objectives is being violated, or might be on track to be violated if things are left as they are. An alert doesn't mean your client is yelling at you; in fact, in an ideal world, they may not have noticed anything, yet.

An incident is the declared start of an Incident response process, and that's all about what you do next. If you declare an ankle incident then you might have to go easy, perhaps ice it a little, or you might need to go to the emergency room. With software systems you need to answer, does this alert (pain) trigger a response, or is it a simple anomaly? In other words, can I safely ignore it or do I need to start some form of incident response (does it need a doctor)? Who is it that makes that decision? Does the client get informed? How about the boss? Does someone need to roll out of bed,

or can it wait? All of this should be part of your standard incident response methodology, which all kicks in when an incident is declared.

There's More to Incident Response than a Resolution

Resolving Incidents

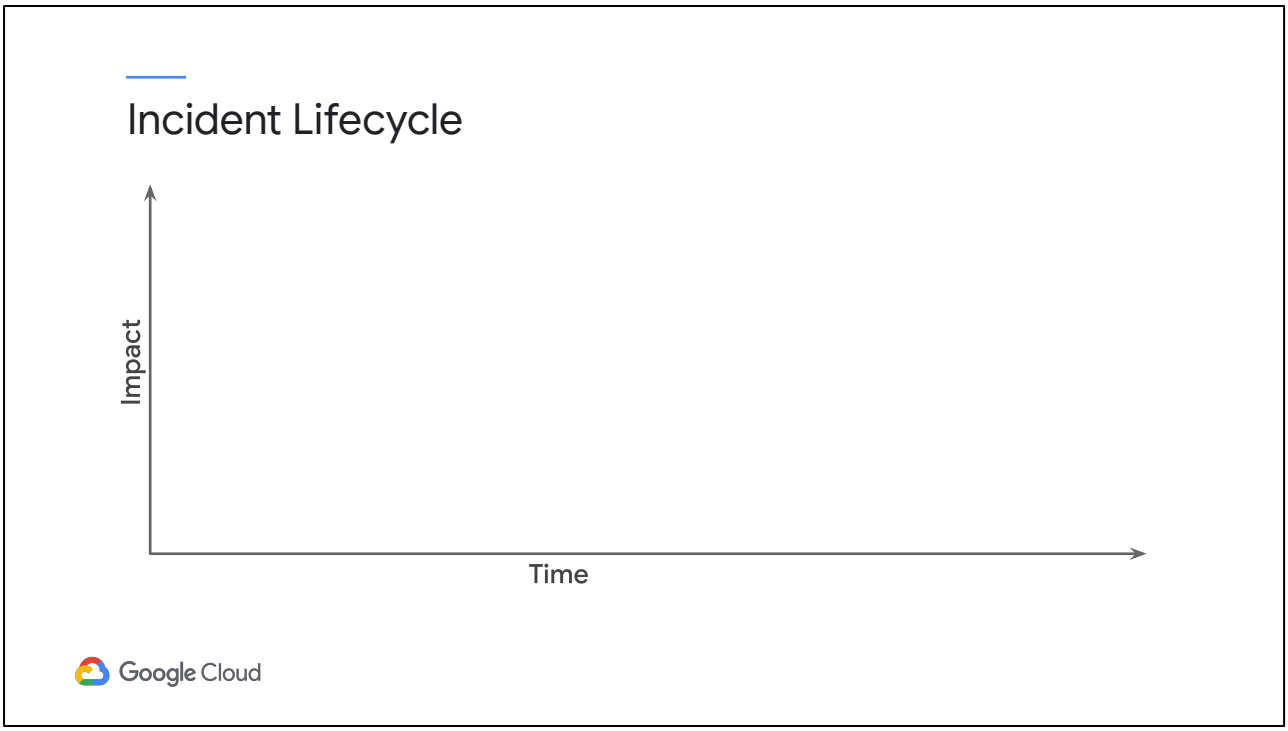
Incident Response

You need to do BOTH!

Google Cloud

Now, let's be clear. There's putting out a fire by resolving an incident, and there's formal Incident Response. A lot of organizations are in the ad hoc fix the problem business, but that's not where you want to be. You want to be one of those pilots, pulling out the checklist for some issue to make sure it gets fixed, and fixed right. Then you document it, so it doesn't happen again, and if it does, you have an even better checklist.

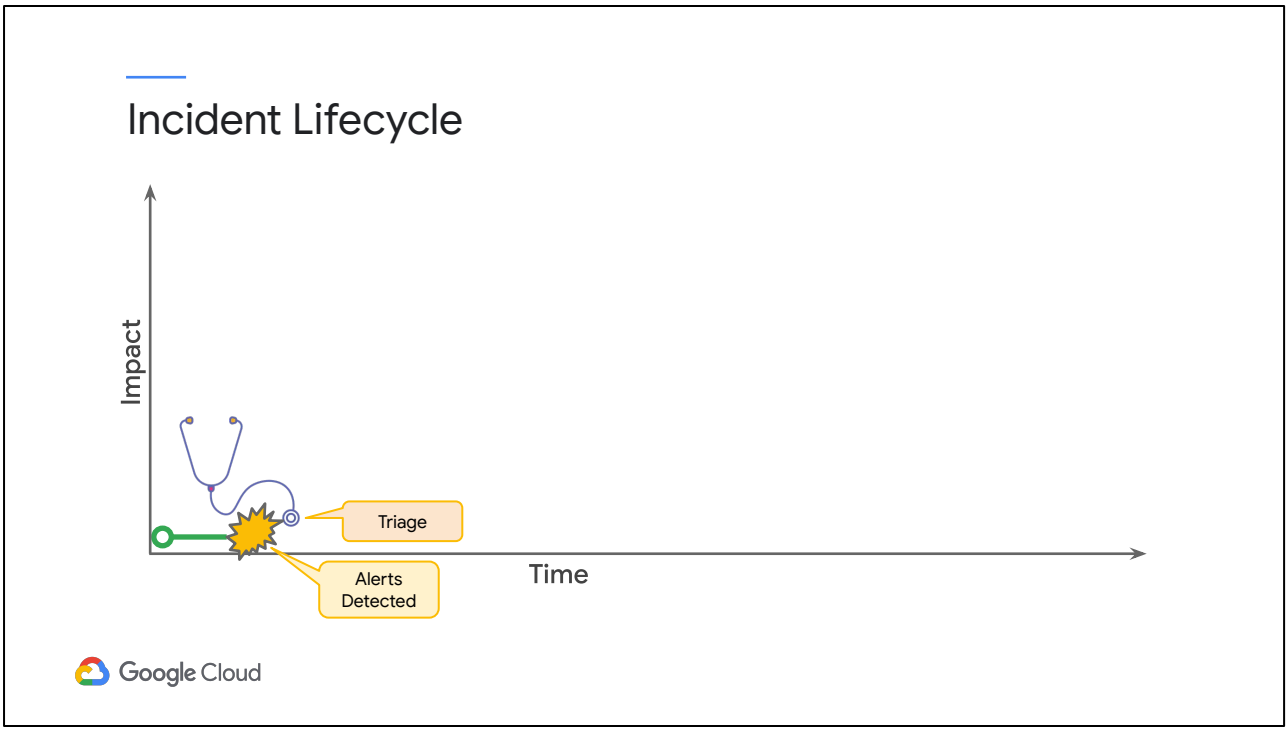
You want a formal incident response methodology that resolves the incident in a fast, organized way.



Have you ever watched a pilot start a plane? Or take off? Or deal with an issue in flight? They have checklists. Books of checklists. Checklists for starting the plane, for the take-off process, and for every alert that the aircraft might give them. Because every pilot knows, if they miss a step, or handle an incident improperly, someone could die.

Your system might not be life or death, but your incident response process can learn from how pilots handle things.

Every incident follows a pattern. On the slide, you see a graph with the impact moving up along the y-axis, while time moves right along the x.

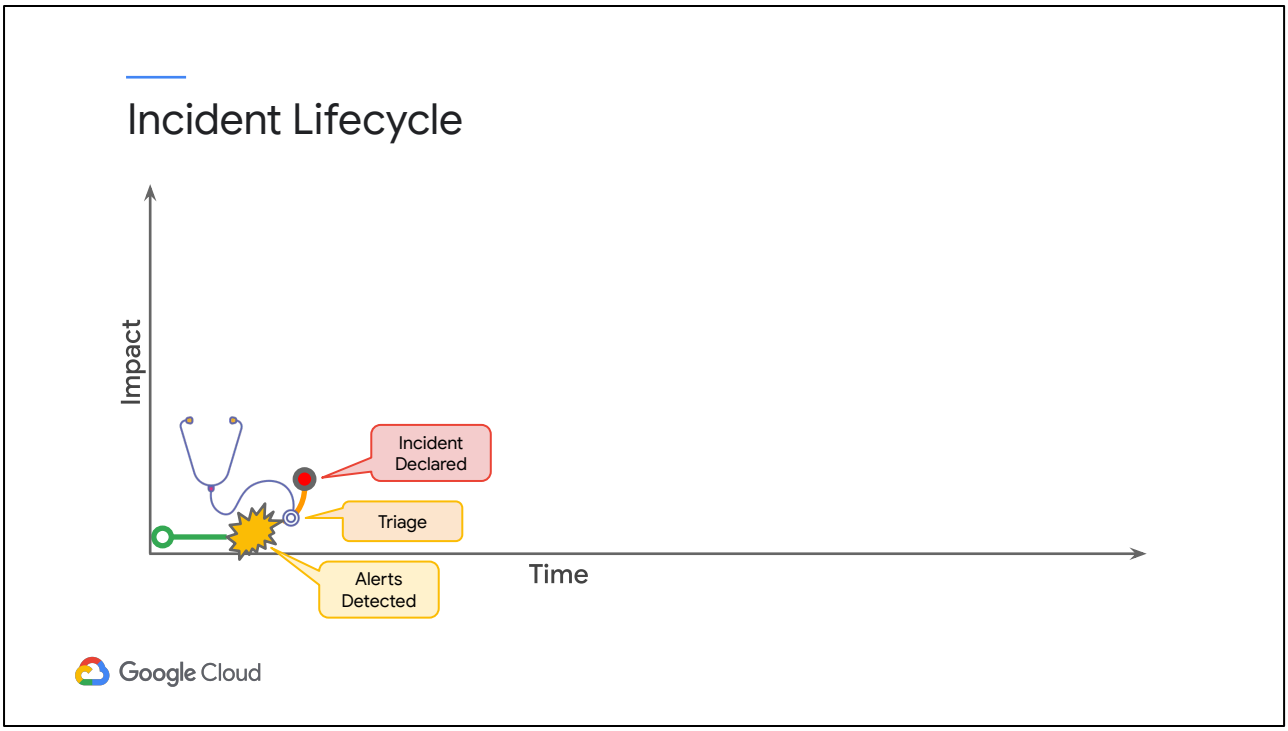


Start in the lower left. Things are going along fine when bang! Alerts start firing. This moves you into the first major incident state: **Detected**. Something untoward is happening, but that's all you know.

Ouch! You tweaked your ankle.

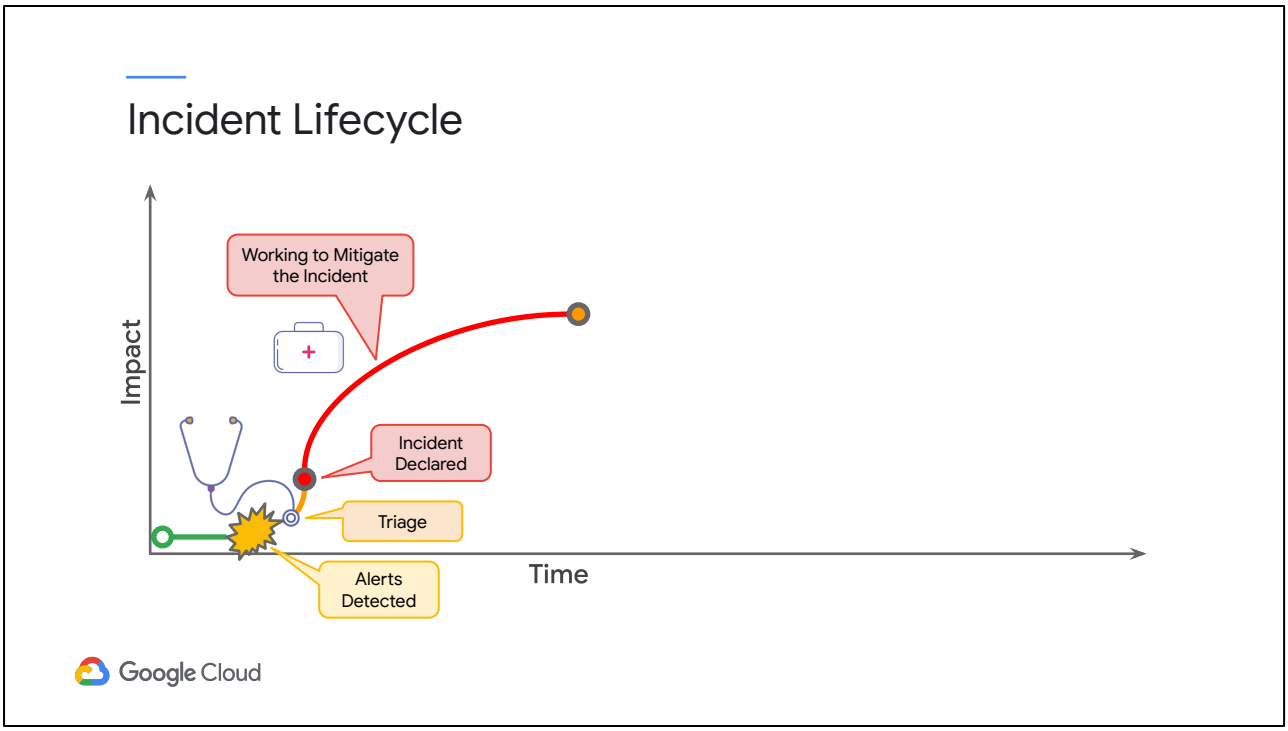
Triage is started. How bad is it? What's happening? And someone needs to make a decision: is this an incident or a simple anomaly.

Does the ankle need treatment?



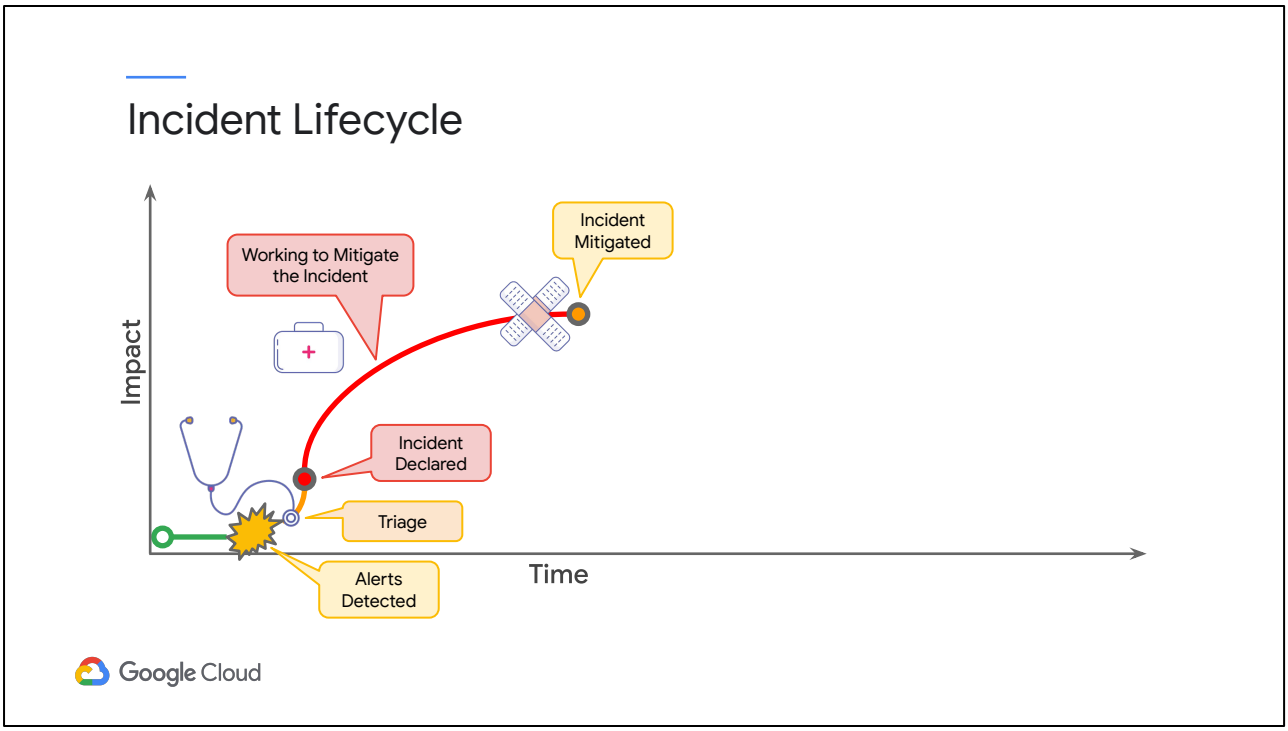
That takes you to the next incident stage: **Triaged**. Let's say that triage decides that this is something that has to be dealt with, so an incident is declared, and an initial severity is set.

Your ankle needs a trip to the doctor.

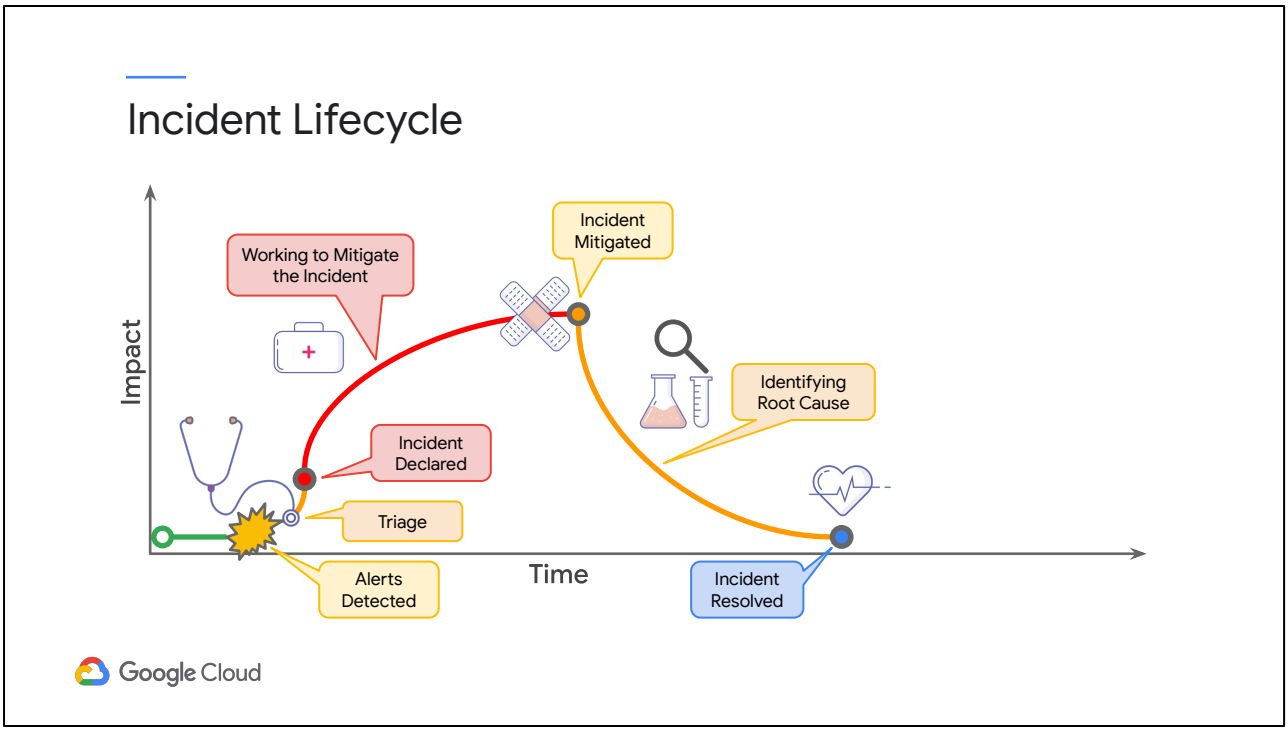


You and your team are working on the problem. The longer it goes on, the worse it gets. Your goal here is to make things better and, if possible, to fix the problem.

They put a temporary splint on your ankle (ouch), then get you to the doctor. Unfortunately the diagnosis is it's broken. Lets put it in a cast and get you some crutches, and pain killers might be nice.

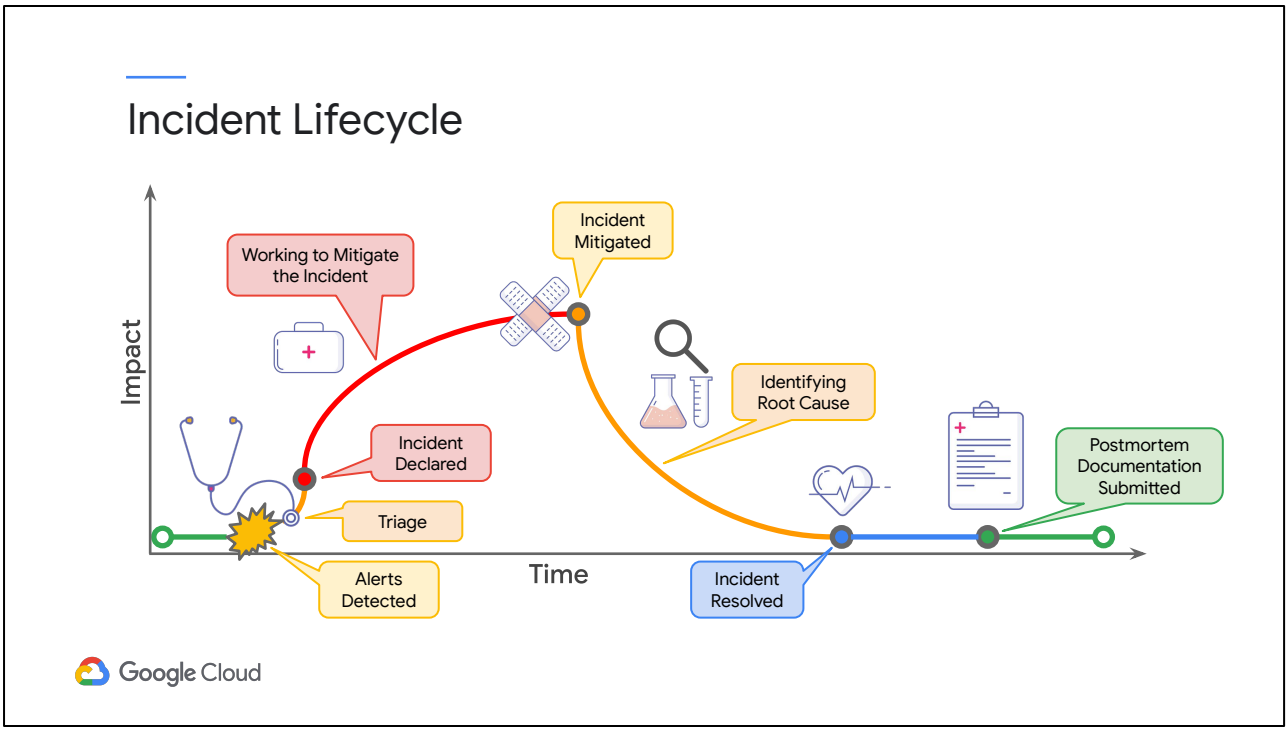


When the patch is in place, your incident moves into the next stage, **Mitigated**. At worst, you've lessened the problem. At best, the client is no longer being impacted.



Now you do your root cause analysis. Why did this happen? How can you make sure it never happens again. Once you can answer those questions, your incident moves into its final phase: **Resolved**.

Pay attention when stepping off curbs.



You write up your postmortem documentation, so everyone knows what happened, why, and what we've done to make sure it doesn't happen again. You submit it and move on.

User Trust **Depends** on Incident Response

- User trust is built upon consistency and reliability
- To develop and strengthen trust:
  - Minimize outages
  - Respond and learn quickly
  - Provide transparency

Google Cloud

Google learned that user trust **depends** on incident response. Users don't expect perfect service. We all work with technology, and we know stuff can happen.

Users trust systems that are not only reliable but are also consistent in the way they handle problems.

So to develop and strengthen user trust, your system must:

Minimize outages.

Respond quickly when things go wrong, and learn from the experience.

And be transparent with the user. Honest and open.

Develop a **Structured** Incident Response

Being prepared:

- Fast response
- Know what to do when

Consistency:

- Reduce duplication of effort
- Know who is doing what

SRE Culture

- Collaboration, communication
- Learning and sharing
- Organized flow of information
- Organized customer notifications

Google Cloud

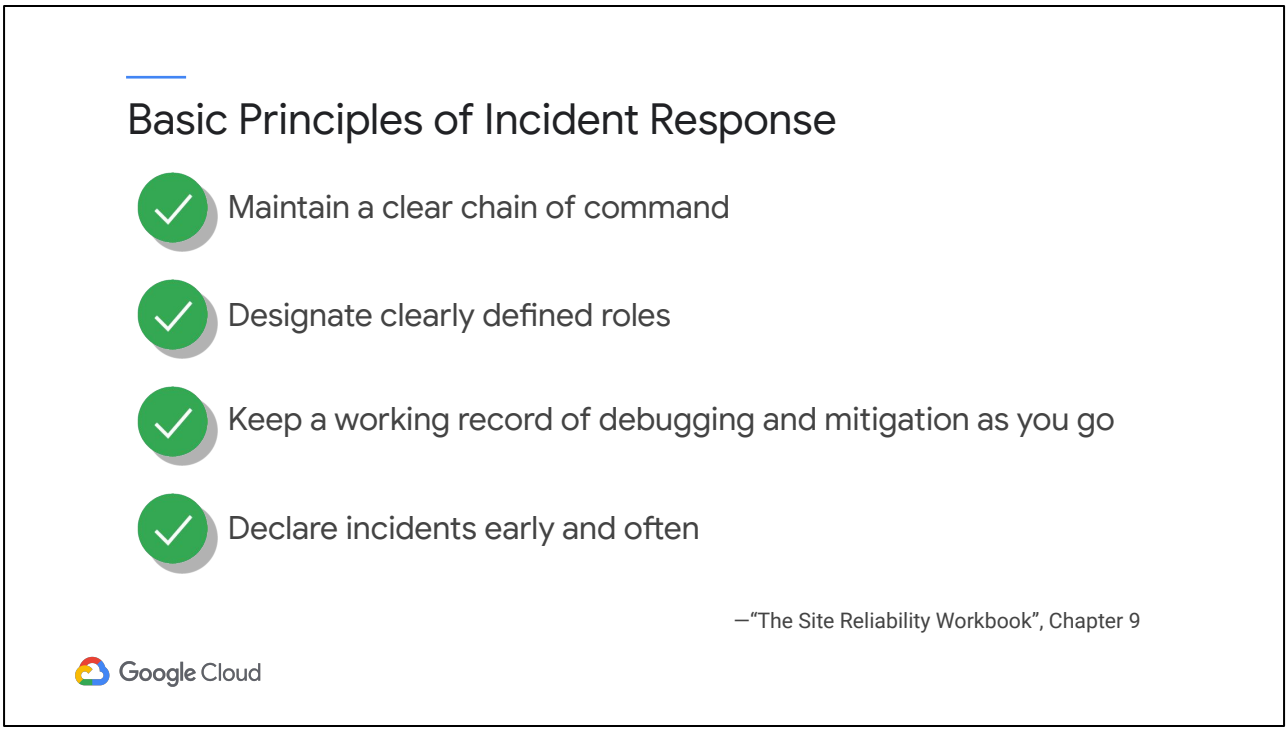
The time to decide how to handle an incident is not when it's happening. You, and your organization, need to develop a structured incident response methodology.

You want to build an SRE culture:

- On collaboration and clear communication.
- Which learns and shares that knowledge.
- With a predefined, and organized information flow
- And, with a clearly organized approach to client notification.

Once your SRE foundations are in place, then it's time to practice. That way you can be prepared and respond quickly, because you'll have the checklists, and you know what to do when.

As you and your team train, you'll also learn to be consistent. Like the pilots, you want to reduce duplication of effort and know who's doing what.



Now that we understand the importance of a standard incident response methodology, let's cover some basic principles.

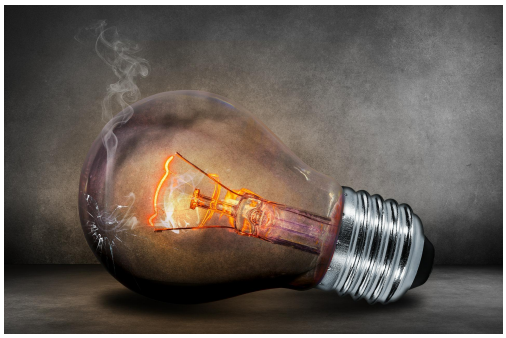
When responding to an incident, maintain a clear chain of command, so team members aren't trying to balance the demands of multiple bosses.

Make sure the incident response team members have clearly defined roles that work in your chain of command.

Keep a working record of debugging and mitigations as you go. Please don't wait until everything's over, or it will never get recorded.

Declare incidents early and often. It's better to roll on an incident and discover it was a false alarm, than to let something and let it get exponentially worse.

Plan for Major Outage



Appoint one person to be in charge of the response

- They delegate necessary tasks to others and coordinate
- Handoff carefully if necessary until incident over
- Maintain a log of the incident state and response

Communicate

- Let those affected know you're responding
- Make it clear who to contact if you want to help

Clean up any loose ends left during incident response

Prepare a postmortem report

- Determine root causes
- Update playbooks, consider adding DR drills

Google Cloud

For your chain of command, start with the incident commander. The commander is the one person in charge of the response.

They delegate necessary tasks to others and coordinate between the team members.

If the incident runs longer than the commander's work cycle, handoff carefully as necessary until the incident is over. But commander changes are inherently disruptive, so don't hand off unnecessarily.

Maintain a contiguous, easy to read log of the incident state and response state.

Be clear and frequent with your communications.

Let those affected know you're responding,

and make it clear to them who they should contact

Make sure to keep a record of loose ends as you handle an incident so you won't leave them hanging.

And don't just fix and move on. Prepare a postmortem report with clear information on root causes.

Last but not least, update your playbook (yes, keep a playbook), so your checklists are even better next time you respond to a similar incident.

Efficient response is built on process, but it's also built on drilling. Drilling makes handling incidents part of a normal process, not a panic activity. Should this incident, or something you learned in this incident response, become part of a drill?

Incident Management Roles

Incident Commander (IC)

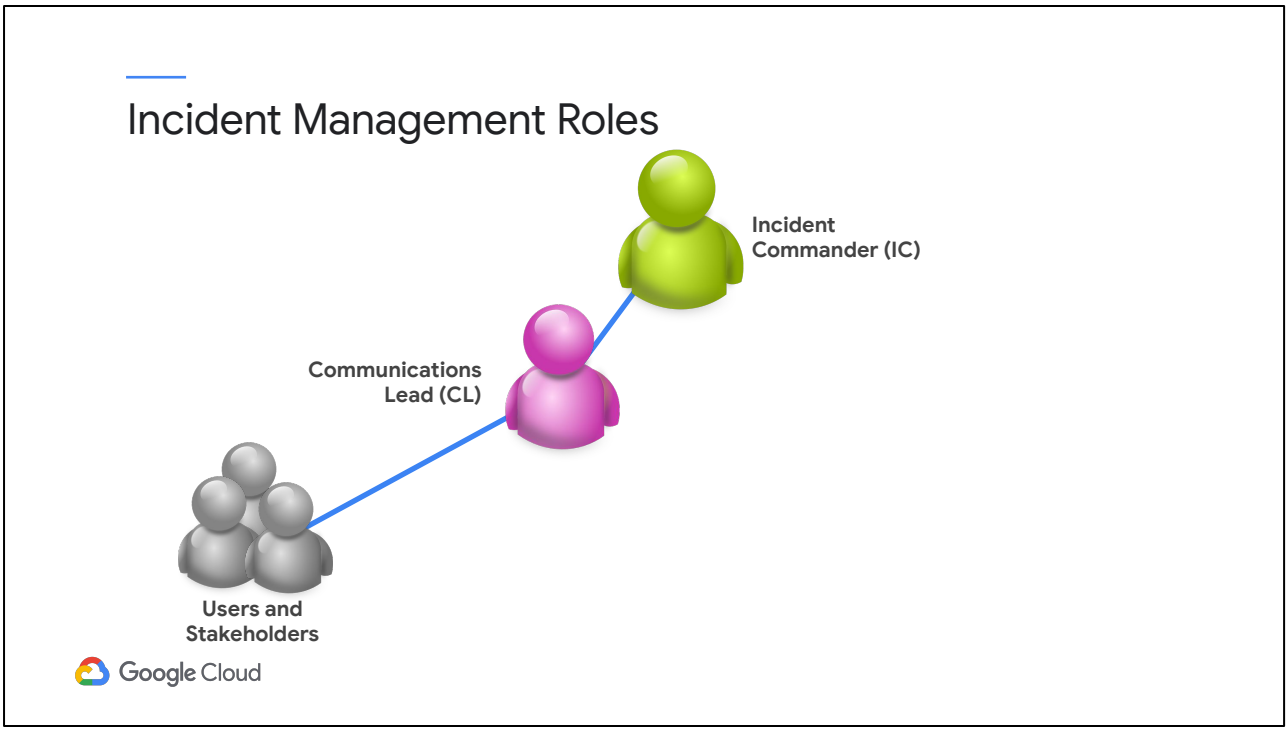
Google Cloud

Here we have some crucial incident management roles. These roles may represent different people, or different roles of an individual, depending on team size and incident needs.

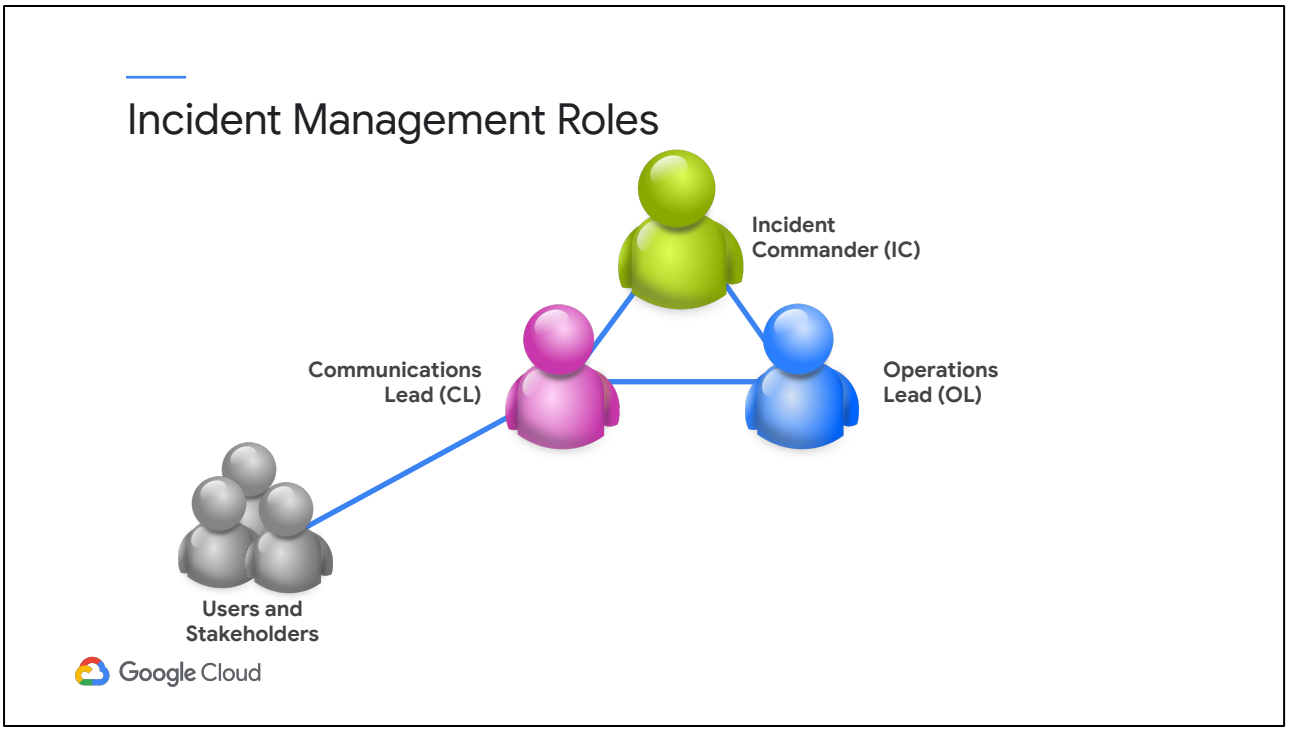
Starting at the top.

The Incident Commander (IC) is at the top of the team chain of command, and during an incident, every member of the response team reports to the Incident Commander. The IC's role isn't to personally fix the problem, but to make sure the problem gets fixed. Think management and organization, not technical wizardry.

IC responsibilities include building and maintaining the team, coordinating the parts of the response, deciding priorities, and delegating activities. It's the IC that always knows the current status of significant events during the incident, and the IC will also handle communication if a communication lead is not assigned.

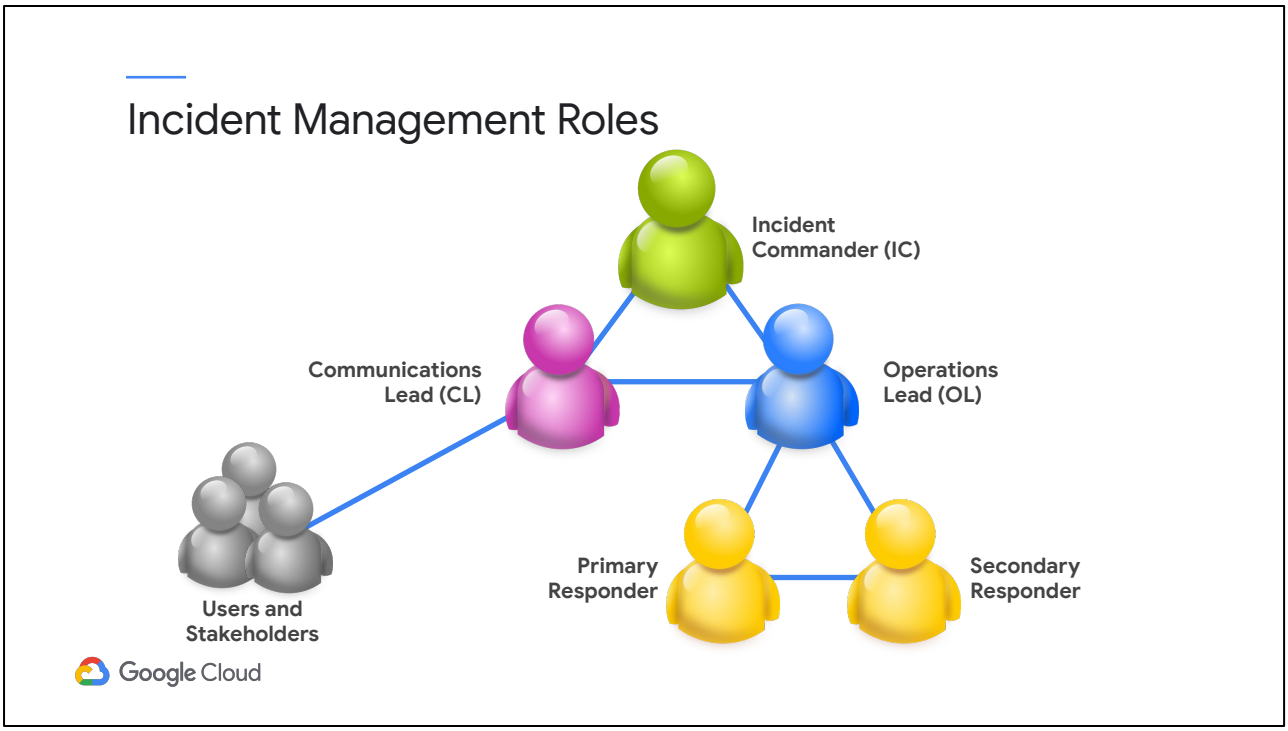


Next, we have the Communications Lead (CL). The Communications Lead is responsible for clear, timely communication. The CL keeps everyone outside the response team informed, and it's the CL that fields questions, and who decides what should be released and through which communication channels. So the CL faces the outside world and lets the team work in peace. Direct communication between the client and other team members should be actively discouraged.



The Operations Lead (OL) manages the immediate, detailed, technical, and tactical work of the incident response. The OL develops and executes the incident action plan, requests additional resources as needed, allocates resources among the various operational response, and maintains close contact with the Incident Commander, Communications Lead, Primary and Secondary Responders, and others involved with handling the incident.

The Operations Lead leads, and works closely with, the responders.

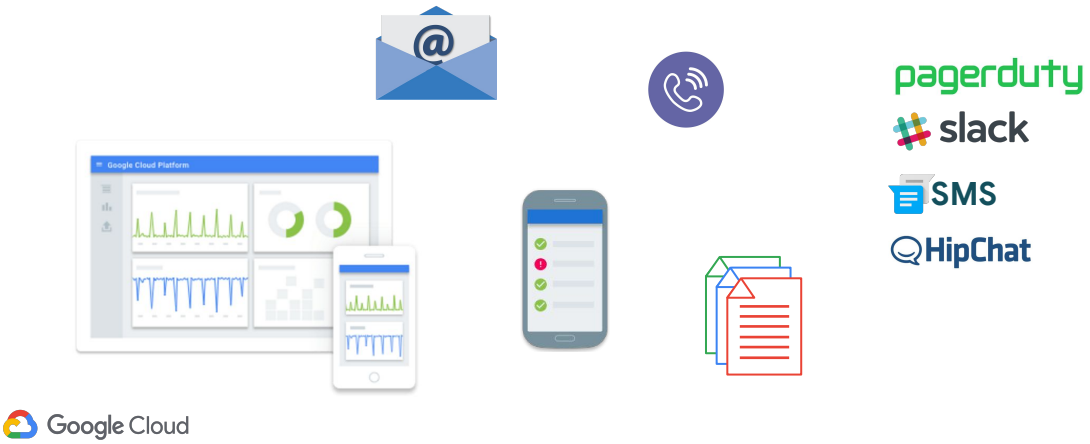


The Primary Responder executes the operation lead's technical response plan, and maintains close contact with the Operations Lead and, if there is one assigned, uses the Secondary Responder for additional technical support.

The Secondary Responder role assists the Primary Responder if they need help on the particular incident.

Communication Channels

- How will you keep your team informed?



How you choose to keep your team and the client informed will depend on your organization, the team, and the client.

Dashboards or simple shared documents can be radiators of useful business intelligence, and can relay a lot of information very concisely. A good example would be the Google Cloud Status page.

More direct and interactive options may include email, SMS, conference calls, chat rooms, Slack, or any of a number of third-party services like PagerDuty.

Best Practices Before the Incident

- Establish clear criteria for declaring an incident
- Choose a communications channel in advance
- Create templates for communicating
- Train the team how to respond
- Conduct drills and practices

Google Cloud

We said earlier in this module that starting your SRE practices in the middle of an outage is a bad idea, so before the first incident:

Establish clear criteria for declaring an incident and remember, err on the side of caution. Better false incidents, than missed.

Choose a communications channel in advance and make sure SRE members and clients know the plan.

Create templates for communications. Templates are easy to fill in, save time, and help avoid missing information.

Train. Train. Train. Train the team on how to respond in theory, and then drill them to make sure they know how to practice what they've learned. Drilling also helps avoid panic.

Agenda

Incident Management

Declaring an Incident

Restoring Service

Preventing Recurrence

Google Cloud

Here’s where it starts, with the declaration of an incident.

When do you Declare an Incident?

- The threshold is defined by the organization
- Example incident triggers:
  - A specific active alert
  - A number of active alerts
  - The duration of an active alert
  - A combination of different alerts

Google Cloud

When do you declare an incident? Well, it depends.

Each organization must define for themselves what constitutes an incident, and what's just an anomaly. The secret is not to wait, but to have clear guidelines that help operators determine when to declare an incident. Ambiguity is your enemy here. If operators don't know when to declare an incident, they may do so too late. A late declaration of an incident may impair the response team's ability to mitigate the incident and restore service. A corollary to this is that you have to be very careful how you deal with false positives. If you make people afraid of reprisals, things will be missed.

Example incident triggers include specific alerts, alert counts, alert durations, or combinations of the above.

Assess the Impact

- What exactly was detected?
- What does that mean?
- Who is impacted?
- How are they impacted?

- Does this qualify as an incident?

Google Cloud

Another incident qualifier is impact assessment. When an alert fires answer the questions:

What was detected?

What does that mean?

Is anyone being impacted? If so, who and how?

Now that you have these answers, is this an incident?

Declaring the Incident

- When you declare an incident you become the Incident Commander

Assemble your team

Triage the Incident

Start Communicating

Google Cloud

When you declare an incident, you become the Incident Commander. On a side note, this should tell you something about the type of person who has the power to declare incidents.

Now the process starts.

Assemble your team.

Triage the incident.

Start communicating.

Incident Declared - Comms Report!

- Report the service outage
- List the impacted services
- Estimated time to recovery

"We are currently investigating an issue with instances that utilize Persistent Disk (PD) experiencing increased disk latency following a migration event."

Google Cloud

An incident has been declared. The communication lead needs to step up and issue a notice. It doesn't have to be complex, but ideally, it should contain: a statement that there's an issue, a list of the impacted services, and if possible, an estimated time to recovery.

While writing this slide, I just checked the Google Cloud Status Dashboard, and up top in yellow, it reads: "We are currently investigating an issue with instances that utilize Persistent Disk (PD) experiencing increased disk latency following a migration event."

Simple and to the point.

Agenda

Incident Management

Declaring an Incident

Restoring Service

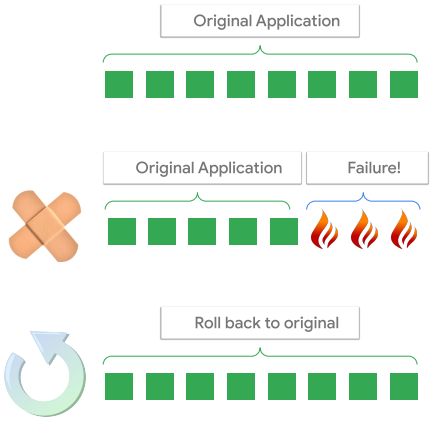
Preventing Recurrence

Google Cloud

When an incident is declared, getting service restored is the priority.

Stop the Bleeding

- Has someone checked the symptoms in the playbook?
  - Suggested fix?
- Has something changed?
  - Revert
- Is connectivity still up?
  - If no, how can we work around?
- Do we have a replacement system?
  - Hot swap, cold start, IaC



Google Cloud

You've declared an incident. That usually implies that your system was working, but now it's not.

Has someone checked the symptoms in the playbook? Is this a known issue with a suggested fix? Easy button!

One common question you can ask is, what's changed? Is it possible the issue is related to the change? If so, can we revert?

Connectivity is a common source of issues, is the network up? Everything communicating as it should? If not, what can we do?

If something is down, do we have a hot or cold replacement? Could we leverage infrastructure as code (IaC) and spin up a replacement environment?

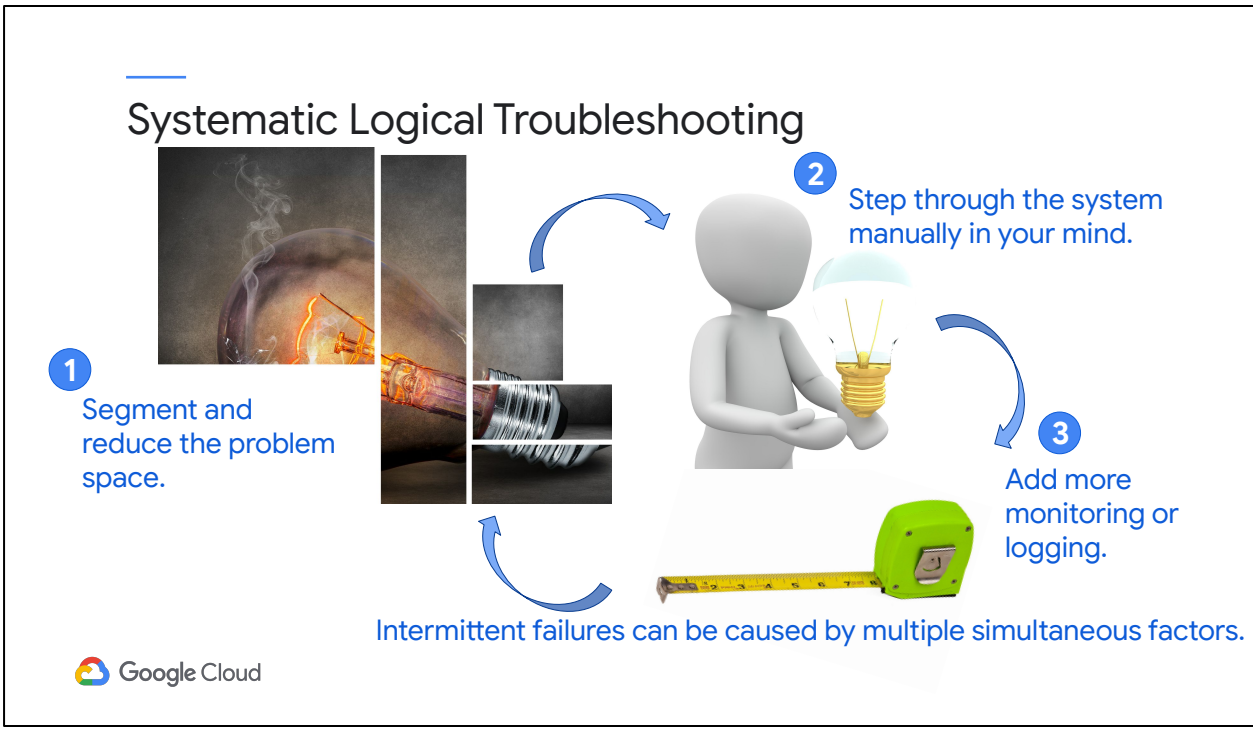
Status Update - Comms Report!

- Here's what's happening
- Here's what we know / what's changed
- Issues that are still ongoing

Google Cloud

Don't merely let users know there's an issue, and not update them until things are fixed. If there are significant knowledge or situational changes, let the users know.

Here's what's happening. Here's what we know or what's changed. But these are the issues that are still ongoing.



It there isn't an easy button for the problem, then troubleshoot systematically and divide and conquer your way to success.

Segment the problem space, so you're looking at smaller pieces, and see if those pieces are working or if one or more could be creating the issue.

Step through the system in your mind, think about what the parts are doing, and could they be contributing to the bad behavior you are seeing.

If something doesn't smell right about one of the parts, but there isn't definitive evidence one way or the other, add monitoring and logging to prove one way or another. If you find the problem, fix it. If not, move on to the next logical piece.

Don't get too myopic with your suspect list. Failures, especially intermittent failures, are frequently linked to multiple simultaneous factors.

Bespoke Mitigations

- You may need to build a custom mitigation
- Examples:
  - Failed external dependencies
  - Unable to roll back update
- Do what it takes to restore service

Google Cloud

There will be times where there is no easy button, and you will have to create bespoke mitigations.

Examples may include failed external dependencies you have no direct control over or multi-system changes where you can't easily rollback.

Ultimately, it doesn't matter. You have to do what it takes to restore service.

Incident Mitigated - Comms Report!

- Incident has been mitigated
- Service is restored
- Users are no longer impacted

Google Cloud

Once you have stopped the bleeding and mitigated the impact of the incident, you should inform the stakeholders of the status.
The Communications Lead should notify the stakeholders clearly and in no uncertain terms:

The incident has been mitigated,

service has been restored,

And users are no longer being negatively impacted

Troubleshoot Root Cause

- Now you must look for the root cause
  - Investigate multiple possibilities in parallel
  - Document your progress as you go
  - Keep the team informed

Google Cloud

But just because the incident has been mitigated, that doesn't mean that the work of the incident response team is complete. No, far from it.

Now your team needs to look for the root cause. Troubleshooting and mitigating the incident may have given you a partial understanding of the reason for the outage, but we need the specifics.

Since the fire is out, you have a little breathing room and should be able to investigate multiple possibilities in parallel.

Continue to document your progress as you go and keep the team and the stakeholders informed.

Find and Fix the Real Problems

- The root cause is always:
  - Systems
  - Processes
  - Behaviors
- System stability depends on finding and fixing the core problems
- Learn together to avoid repeating mistakes
  - Fix what can be fixed
  - Prevent what can't be fixed
  - Handle what can't be prevented

Fix problems, not people

Google Cloud

Consider this example:

The service was out.
1. Why was the service out? Because the filesystem was full.
2. Why was the filesystem full? Because the person responsible for archiving old files failed to do so.
3. Why didn't they archive the old files, was the archive tool broken? No, the tool wasn't broken.
4. Why didn't they archive the old files? The procedure documentation didn't tell them to archive the files.

So the root cause was a problem with the process documentation.

The root cause is always systems, processes, or behaviors.

Blame makes people afraid to bring real issues to light and is detrimental to a learning culture. If the analysis in our example had stopped with, "because the person responsible for archiving old files, failed to do so," then the person might have been punished without solving the core system problem, which was absent procedures.

System stability, uptime, and user trust all depend on finding and fixing the core problems.

Outages happen, and you can never stop them completely. What you can do is learn

as a team and avoid repeating mistakes. And when I said "as a team," remember that what's clear to you, might not be clear to everyone on the team.

Fix what can be fixed, prevent what can't be fixed, and handle what you can't prevent.

Resolve the Incident

- Once the root cause has been identified:
  - Implement the permanent fix
  - Document the resolution
  - Comm's report

Google Cloud

Once the incident has been fully resolved and the root cause identified. Implement the permanent fix and remove any temporary mitigations, document the resolution, and release a comm.

Incident Resolved - Comms Report!

- Root Cause Analysis completed
- The incident has been resolved
- Incident is being documented

Google Cloud

The Communications Lead should notify the stakeholders and users. The root cause analysis has been completed, the incident has been resolved and we are in the process of documenting it.

Agenda

Incident Management

Declaring an Incident

Restoring Service

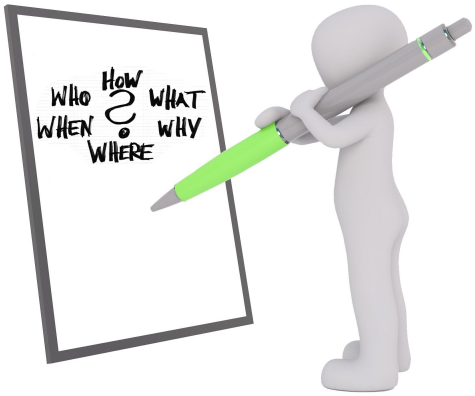
Preventing Recurrence

Google Cloud

The incident has been fixed, and that's important, but perhaps even more important is preventing it from happening again.

Postmortem Report

- A detailed and organized document outlining the events of the incident
- A blameless report of:
  - Impact
  - Root cause
  - Triggering event
  - The resolution
  - Quantifiable metrics
  - Action items
  - Lessons learned
  - Timeline



Google Cloud

The first step towards making sure problems don't reoccur is the postmortem. Remember my pilot's with the books on how to handle incidents example? This is where those books originated.

A postmortem report is a detailed and organized document outlining the key events of the incident. It's a blameless report containing the impact, root cause, triggering event, the resolution, any quantifiable metrics, action items, lessons learned, and a timeline.

Why Create Postmortems?

- Letters to your future self and team
- Helps with future troubleshooting
- Provides clarity on future mitigation

Google Cloud

Why bother creating postmortems, especially if the incident has been resolved?

Postmortems are letters to your future self and team. They help with future troubleshooting and provide starting points and clarity on future mitigations.

This is the origin of those books that airplane pilots can reference. Whenever an incident occurs related to a particular type of aircraft, it gets documented and possibly included in the next generation of those books.

Your organization should have the same type of methodology.

Policy for Writing Postmortem Reports

- Always write a report:
  - Anytime an SLO is breached
  - When an incident required an emergency (on-call) response from another team
  - When an impacted team requests a follow-up communication
- You should have a policy regarding these reports:
  - A draft report should be published within X hours of the incident resolution
  - The report should be completed within Y business days

Google Cloud

Your organization should have policies and procedures for writing postmortem reports. This is an important part of the learning process.

As a best practice, always write a postmortem whenever an SLO is breached, when an incident required an emergency response, and when an impacted team requested a follow-up.

Your policies should cover how long from incident resolution to draft report, and to the final report.

Standardize your Documentation

- Use public postmortem templates
- Document the entire troubleshooting process
- Include any policy changes that should be made
- Include a review of the Incident Response itself
- Never stop learning

Google Cloud

Just like the incident response itself, standardize your documentation. You can find postmortem templates from Google, PagerDuty, Atlassian, and others.

Document your troubleshooting process, so you learn what worked and what didn't.

Include recommendations for policy changes that should be made.

And, include a review of the incident response itself.

Never stop learning

Use Simulation Sessions

- Experienced member presents a failure scenario to Team member(s)
  - Investigate using system's monitoring and diagnostics
  - Determine the cause
    - Loss failure
    - Overload failure
  - Present strategies for fixing the issue

Google Cloud

As stated earlier in this module, practice drills are an excellent way for teams to build experience, refine incident response methodologies, and get used to handling incidents.

Simulation sessions work like this. An experienced team member presents a scenario based on system knowledge. A great way to do this is to have the incident commander pull a real-life, historical incident.

Team members investigate using the system's monitoring and diagnostics, determine the cause, and present strategies for fixing the issue.

Use your whole incident response process and practice until it's all second nature.

Quiz

Which of the following are incident response team roles?

A. Incident Commander, Operations Lead, Communications Lead

B. Helpdesk, DevOps, Engineering

C. Executive, Operations, Lead

D. On-call Responder, Helpdesk, DevOps

Quiz

Which of the following are incident response team roles?

A. Incident Commander, Operations Lead, Communications Lead

B. Helpdesk, DevOps, Engineering

C. Executive, Operations, Lead

D. On-call Responder, Helpdesk, DevOps

Quiz

Which of the following is the most important task after an incident has been declared?

A. Assign blame

B. Postmortem

C. Root Cause Analysis

D. Mitigation

Quiz

Which of the following is the most important task after an incident has been declared?

A. Assign blame

B. Postmortem

C. Root Cause Analysis

D. Mitigation

Quiz

Which of the following documents summarize the events during the incident and evaluate the effectiveness of the incident response?

A. Support tickets

B. Alerts

C. Postmortems

D. Communication Channels

Quiz

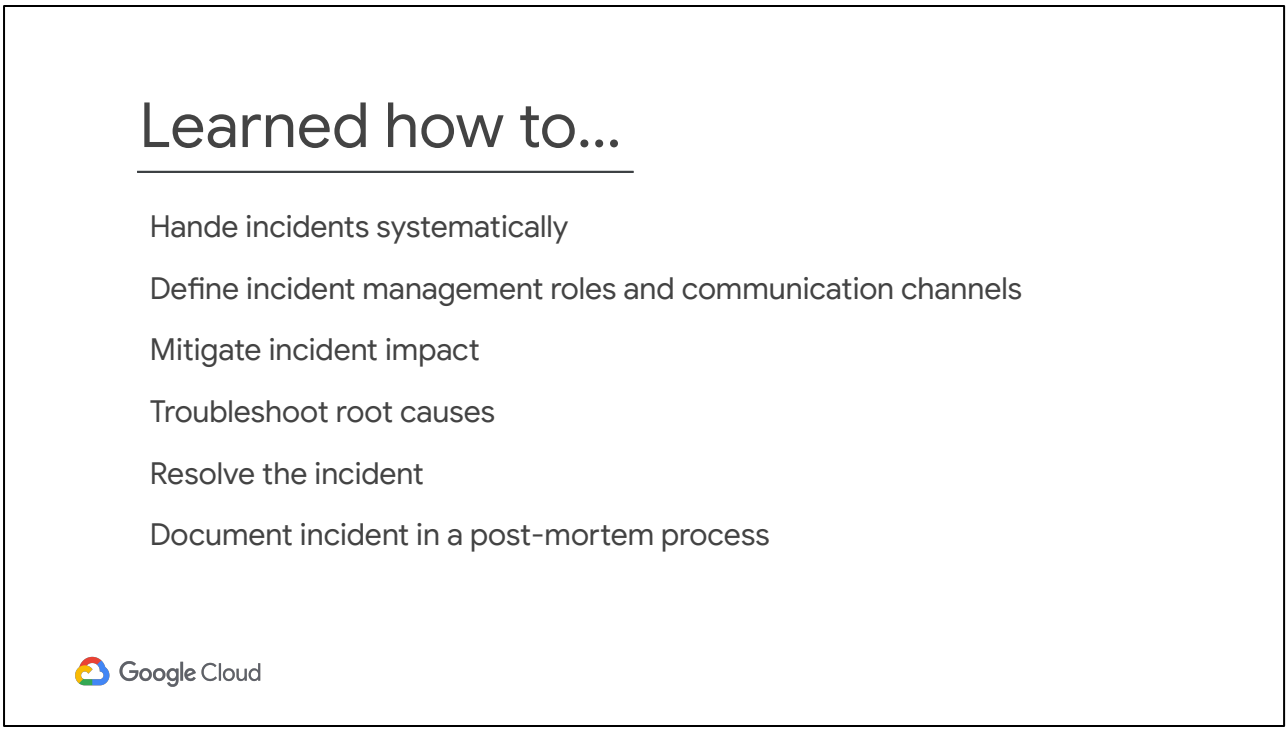
Which of the following documents summarize the events during the incident and evaluate the effectiveness of the incident response?

A. Support tickets

B. Alerts

C. Postmortems

D. Communication Channels



Well done.

In this module you have learned how to:

Handle incidents using a systematic process.

Define incident management roles and communication channels, so the team knows what their jobs are, and the users are kept in the loop.

See what, if anything, we can do immediately to mitigate the incident's impact

Troubleshoot root causes of the incident so we can (hopefully) get a permanent fix in place.

Resolve the incident, or at least come up with a ready workaround

And document incident as part of a standard post-mortem process

Way to go!