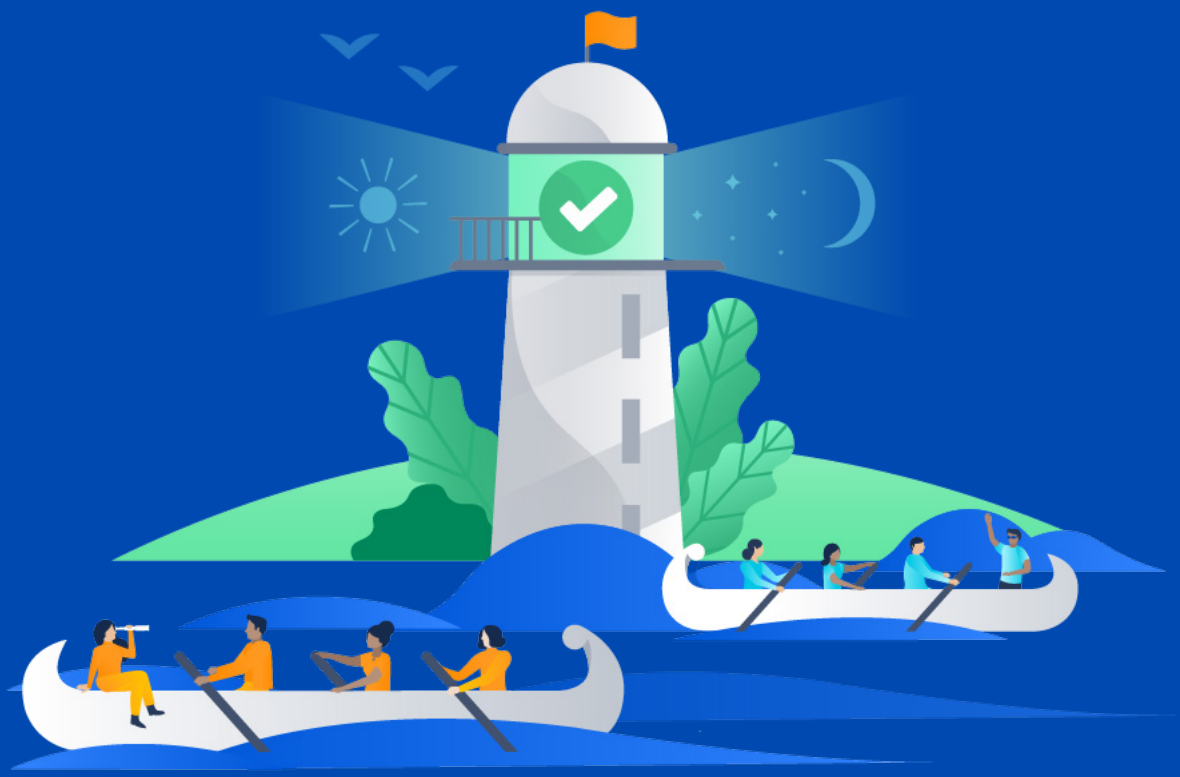
ATLASSIAN

# On Call

The definitive guide to running productive and happy on-call teams



Serhat Can

FOREWORD BY ANDREW CLAY SHAFER

# Foreword

Computers are not new, but many institutions and organizations were born before computers. Computing investments have been made along the way, but in many organizations, those computing capabilities have mostly been auxiliary support for business as usual. Over the last decade, the digital natives, organizations where computing is central to the business, have redefined the world. Across the planet, experiences have been reconfigured around new capabilities evolving from software-enabled by pervasive computing connected by high-speed networks. Shopping, eating, transportation, dating, politics; every activity, every need, transformed forever. We are starting to take these digital experiences for granted. We are increasingly shocked when they are not available. The most advanced and highest performing digital native organizations realize that these systems are not purely technical. Hidden behind every one of these digital capabilities stands a team of people whose actions keep those systems running. The reliability we take for granted is provided by a combination of technology, people, and processes.

As these socio-technical systems connected the world together, communities of practice emerged to share the lessons learned through those connections. The DevOps community has been sharing their principles, practices, and tools co-evolving to operate these systems for the last decade. From the beginning of DevOps, and even before that was a word, the topic of on call in theory and practice has been ever-present. On call is the front line. On call is where decisions and actions can make the difference between no service disruption and prolonged service failure. Every organization has different needs, skills, and resources. There exists no one true way but some ways are better than others. Many organizations have not planned nor prepared for the reality of 24-hour operations that their digital realities have necessarily thrust upon them. As a consequence, this work either becomes an unfunded mandate falling on the individuals willing to do whatever it takes or else neglect and hope for the best. The unfunded mandate can seem to work for a while but takes a toll on those individuals who, despite their best effort, will eventually falter or worse, burn out and leave. Hope-for-the-best inevitably ends in catastrophe.

Having people on call is a necessity for even the most reliable systems. Those systems are reliable because of those people. The most advanced technology organizations on the planet also have some of the most humane and humanizing call rotations. Why? Because these organizations know their people are a critical component and need to be ready and able to do their best work. They invest a lot of resources into developing their talent and do not want critical systems to be dependent on people who are unprepared and overburdened. The on call practices evolve over time with the tools and context but they are a first-class concern in all high performing digital organizations. The investment of making on call a deliberate practice, instead of an afterthought, often acts as an organizational catalyst to working through many other long-standing technology issues and ultimately enables more effective collaboration between developers and operations in other areas.

The DevOps community has been a central focus of my work for the last decade. This work has given me the opportunity to witness hundreds of organizations and meet with thousands of people. The processes and tools have evolved a lot in ten years but one constant is the truly excellent people who make progress possible. Serhat Can stands out as one of those excellent people. Through his involvement organizing in the DevOps community events, I found Serhat to be adaptable, resourceful, inquisitive, and genuine. He's capable and versatile with broad experience as an engineer and operator, but with a connection to communities that provides access and understanding beyond the confined focus of his work responsibilities. With this book, Serhat is sharing some of that understanding with us. If you are getting started or looking to improve your on call implementation, this will outline options and tradeoffs you should consider. Understanding what other organizations have done should accelerate and improve your results. My advice to you is not to try to create the perfect process but to start with where you are and be thoughtful about where you want to go. I also encourage you to get involved in the local and global communities of practice so you can both learn and share your progress with others. I hope your journey is as fulfilling and rewarding as mine has been.

**ANDREW CLAY SHAFER,**
**VICE-PRESIDENT TRANSFORMATION, RED HAT**

# Table of contents


**Chapter 01: On call – The necessary evil** 6
- Introduction 7
- Importance of on call 8
- Incident response vs. on call 8
- DevOps and on call 9
- Why you should care about on call 11
- Summary 15

**Chapter 02: Choosing your model** 16
- Introduction 17
- Establishing your on-call program 18
- Potential on-call groups 19
- Centralized vs. distributed on-call teams 20
- Summary 31

**Chapter 03: Establishing roles & responsibilities** 32
- Introduction 33
- Typical roles for on-call teams 34
- On-call Team Leader 34
- Primary On-call Engineer 35
- Secondary On-call Engineer 38
- Managers 38
- Customer Support 39
- Importance of transparency in on call 39
- Summary 42

**Chapter 04: Scheduling your staff** 43
- Introduction 44
- Scheduling considerations 45
- Implementing your on-call schedule 47
- Shift options 48
- On-call scheduling best practices 53
- Summary 55

**Chapter 05: Developing the alerting process** 56
- Introduction 57
- Alerting best practices 58
- Leveraging automation 66
- Runbooks 69
- Minimizing alert fatigue 70
- Summary 74

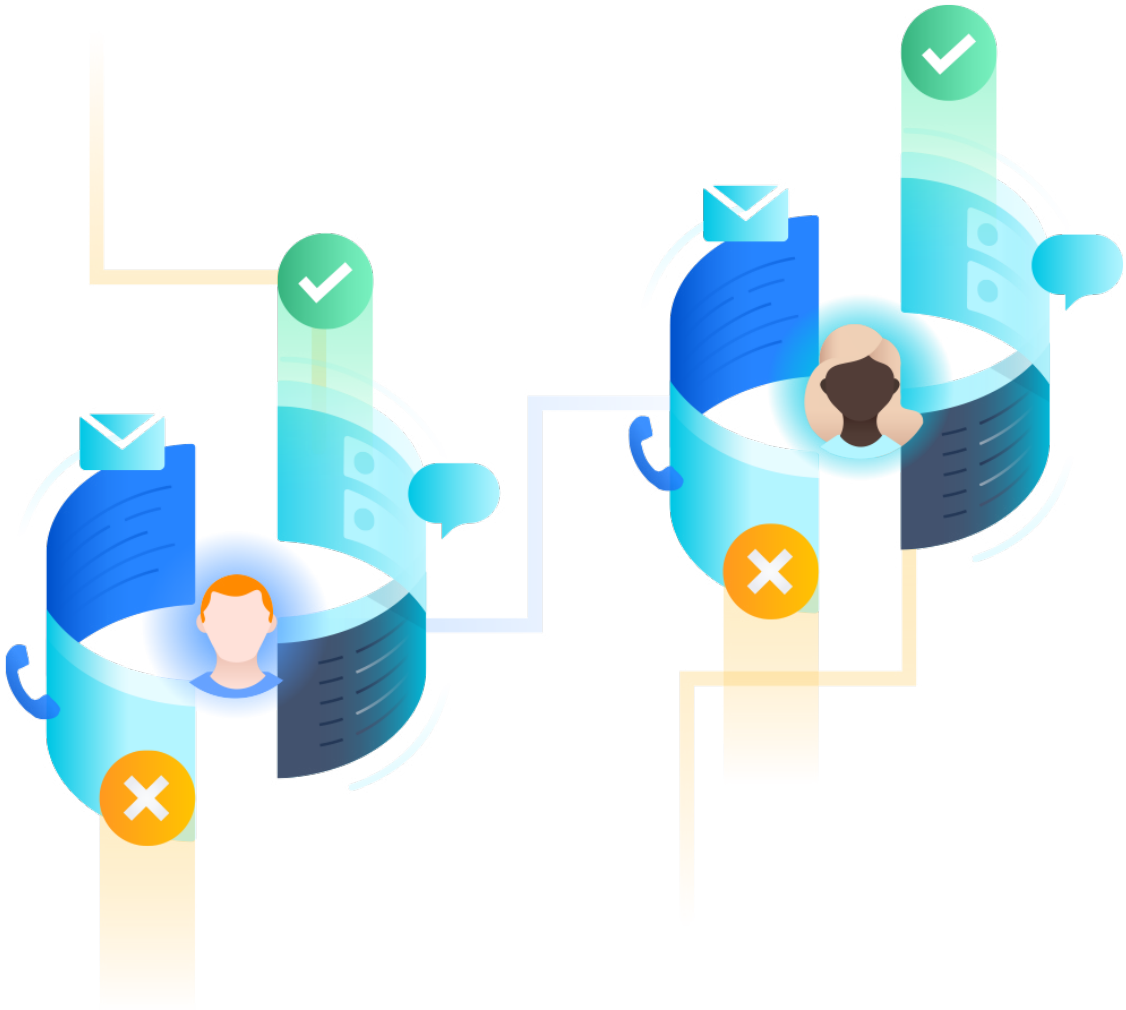
**Chapter 06: Setting the escalation path** 75
- Introduction 76
- Importance of escalations 77
- Reasons for an escalation 77
- Escalations policies & procedures 78
- Types of escalations 82
- Benefit of automated escalations 82
- Summary 83

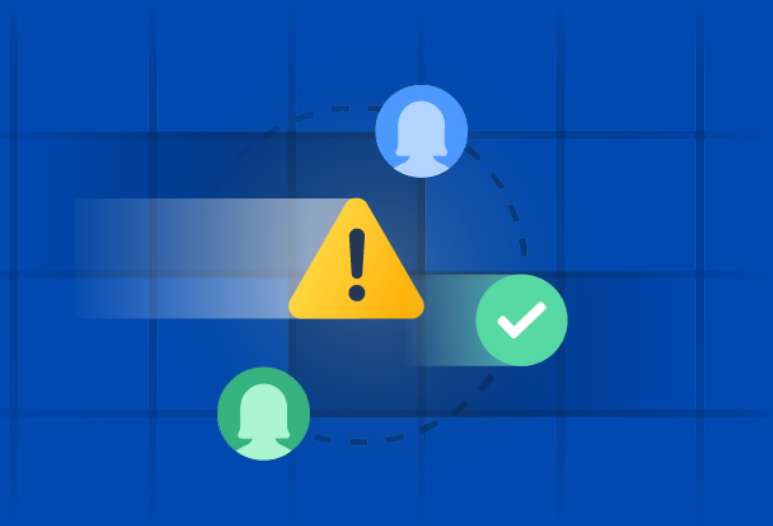
**Chapter 07: Selecting the right compensation model** 84
- Introduction 85
- Typical on-call compensation models 86
- Determining fair and competitive compensation 87
- Summary 88

**Chapter 08: Training your team** 89
- Introduction 90
- Set the tone for training 91
- On-call training process 92
- Training for major incidents: Game Days 95
- Other training resources 97
- Summary 99

**Chapter 09: Learning through experience** 100
- Introduction 101
- Integrate continuous learning 102
- Leverage data to ensure satisfaction 102
- Conduct blameless postmortems 104
- Evaluate successes 105
- Summary 106

**References** 107





# 01

# On call – The necessary evil

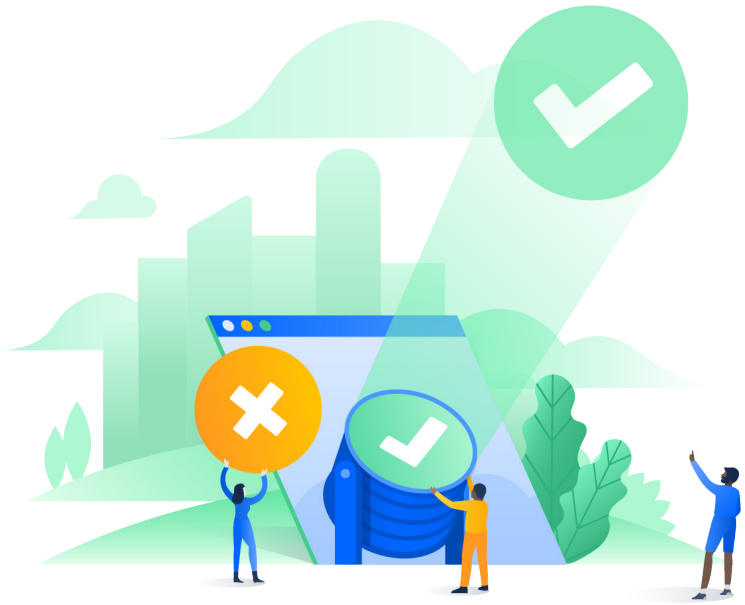
# On call - The necessary evil

## Introduction

Rapid increases in the adoption of your software products and services is likely the key goal for your IT and development team. Unfortunately as usage increases and software evolves, disruptions inevitably become more frequent. Effectively responding to these incidents is critical to ensuring ongoing acceptance of your products and services, and ultimately defining your company's business success.

The effectiveness with which you handle customer issues is often in direct proportion to how well you design and implement your on call response program. Many companies struggle with this challenge and fail to clearly articulate how their on-call program should be set up and staffed.

This book will help you successfully navigate the pitfalls of operating your on-call program by discussing the strategic philosophies and tactical practices that can help you not only face the on-call challenge but will help you transform your on-call program from an internal burden to a key driver of success for your company.



## Importance of on call

The need to implement on-call programs is not limited to software development companies. Healthcare, public service, retailers, and financial organizations all typically must have an on-call group. Regardless of the industry, the basic idea behind on call is the same: be ready to respond to customer issues when you are called.

Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy, editors of Google's Site Reliability Engineering book, define on call as "being available for calls during both working and nonworking hours." During the assigned set period of time, the on-call team must be ready to respond to production incidents with appropriate urgency.

On call is particularly challenging for companies that run always-on services. Meeting this need has spread the responsibility for on call from just the software development or operations teams and is now covered by the all-encompassing umbrella term "IT." An IT on-call engineer is responsible for responding to any unexpected issues related to production services that arise during the shift.

## Incident response vs. on call

Incident response, also known as incident management, is different than on call and can mean slightly different things depending on the context. In IT operations, incident response refers to the methods and protocols deployed following an unplanned failure event, service interruption, or a reduction of expected quality of service. The stages of incident response are outlined in the following box:

| | |
|---|---|
| Planning | Processes are put in place to remediate anticipated incidents. |
| Detection & Alerting | Incidents are made known in whatever service they affect. |
| Containment | Efforts are made to ensure that detected incidents do not affect other services or functions. |
| Remediation | Corrective actions are taken to resolve an incident. |
| Analysis | Resolved incidents are analyzed to implement improvements in resiliency or the remediation process. |

In simple terms, incident response outlines the process while on call refers to the people carrying out those processes. On-call staff have a role in every stage of the incident response process.


## DevOps and on call

DevOps is an approach to software deployment that involves the engineering and operations teams to manage the quality, release, and production bug and incident issue resolution of the software.

DevOps has become a “rising star” in the software industry responsible for best practices like continuous delivery, automation, and infrastructure as code has become mainstream and considered a key element for fast and reliable delivery of IT services.

There is much discussion around the importance of tooling and practices as key enablers of DevOps, but experienced practitioners know that effective DevOps requires a culture of collaboration, sharing, and learning.

Culture often feels like an abstract term, but it is a very real concept encompassing a complex ecosystem driven by people. Watching for outward behavior and analyzing metrics is a key part of culture and can help determine if things are being handled properly. In their book Accelerate[1], Dr. Nicole Forsgren, Jez Humble, and Gene Kim compare data from several organizations and highlight the importance of culture, instituting technical practices, making work sustainable, and driving employee satisfaction as some of the key drivers for success in DevOps.

These elements are indeed relevant to on call since a successful on-call program relies on a culture of sharing and collaboration, good technical practices, more sustainable work, and employee satisfaction. Lack of these critical ingredients can lead to a dysfunctional operation where end users suffer. If organizations do not spend the necessary time and resources to build an effective on-call program, the result can be a waste of time, effort, and money as well as delayed incident response time and employee burnout.

Popular books on software development like The Phoenix Project[2] and The DevOps Handbook[3] discuss the foundational principles underpinning DevOps known as "The Three Ways." This philosophy illustrates how on call is a natural extension of DevOps and acts as a bridge between operations and development. High performing organizations who embrace DevOps principles and practices also devote significant time and resources to their on-call program.

**The First Way: Systems Thinking.** Each team should focus on eliminating blockers by owning the code they write. The robustness of the system is more important than the individual so everyone involved should participate in on call to ensure its strength.

**The Second Way: Amplify Feedback Loops:** Ensuring open and rapid feedback loops is critical. When a defect is detected, predefined escalation policies should notify the engineers about the problem. Having this process established prior to an event is crucial to shortening the loop by ensuring someone knowledgeable is aware of the problem and is immediately taking the necessary actions.

Smart analytics involving on-call and incident response data is another way of establishing effective feedback loops, gaining insights, and improving operations. Comprehensive reporting and analytics on metrics such as a team's MTTA (mean time to acknowledge) or the on-call times of people are critical to providing feedback to developers and managers.

**The Third Way: Culture of Continuous Experimentation and Learning:** Creating a blameless culture with constant experimentation and learning is the best way to learn while solving critical problems. Analyzing incidents and then questioning and improving the related processes is a critical part of DevOps. Conducting postmortems without regard to assigning fault is extremely important for continuous experimentation and learning. This process helps teams better understand what caused an incident, how teams responded, and what actions can be taken in the future to avoid the same incident recurring.

## Why you should care about on call

On call used to be the responsibility of system administrators and operations engineers. These individuals often worked in a Network Operation Center (NOC) and were the first responders for production alerts. Having dedicated staff responding to incidents is changing rapidly because it does not work for today's demanding business requirements and cloud-based systems. Businesses now have to meet customers' demands for always-on services and IT teams have to deliver innovation to meet these uptime commitments or face the risk of losing business to the competition. There are three main reasons you should be focusing on your on-call program:

### On call is a competitive necessity

Building reliable distributed systems is hard and gets even more difficult when you are committed to 99.999% uptime. Yet meeting those "9s" is a huge competitive advantage and missing them can be a threat to the company's viability.

> **"The cost of downtime to an organization can be devastating. Gartner frequently cites a cost of $5,600 for every minute of downtime which is well over $300K per hour.[4]**

Amazon's Wener Vogels has famously said, "Everything fails all the time." Even though monitoring and automation has improved system reliability dramatically, incidents are inevitable.

Assuming that systems are, more or less, in constant failure-mode can help us build more resilient services. The Google SRE book notes that "accidents are normal" so organizations should be focusing on rapid recovery rather than the daunting task of trying to prevent unforeseen incidents. On call is a critical component to obtaining the necessary feedback to conduct diagnoses and implement recovery options.

### Failure is inevitable

Three of the leading cloud providers – AWS, Microsoft Azure, and Google Cloud - control nearly 77 percent of the cloud computing market. When they have problems, the impact is felt around the world. These tech giants spend a tremendous amount of effort and bring years of experience to achieve an unprecedented level of reliability but a look at their status pages shows that even they fail all the time. They do an effective job in reducing the blast radius of the incident, but there are always a few exceptions:

Amazon's S3 service, one of the backbones of the internet, went down on February 28, 2017 and it took more than four hours to fully recover the service. Amazon shared that the issue that brought down the rest of the internet was triggered by a mistyped command.
Learn more here.

On June 2, 2019, a major incident with Google Cloud caused slow performance and elevated error rates in many Google services as well as Google Cloud Platform customers. The issue was caused by a configuration change made for a small number of servers in one region, but it quickly escalated and took more than four hours to fully remediate.
Learn more here and here

On May 2, 2019, Microsoft's Azure suffered a global outage that took nearly three hours to resolve. Many services including Microsoft Teams, SharePoint Online, and OneDrive for Business were affected. The issue was reportedly due to a configuration issue that resulted in DNS resolution problems affecting network connectivity.
Learn more here.

## On call is often a source of stress and burnout

Implementing a modern, effective on-call program is not only important to retain business, but also to avoid the negative impact it can have on your staff. Constantly dealing with problem incidents that are common for on-call staff can be a significant source of stress.

According to Robert Sapolsky, a professor of biological sciences and neurology at Stanford University who studied stress in animals, the duration of time that one experiences stress is important. Short-term stress helps us survive, increases motivation, and can boost memory. Long-term stress, on the other hand, suppresses the immune system and can lead to burnout - a state of chronic stress that leads to physical and emotional exhaustion.[5]

**The burnout plague?**

Many studies show that burnout is becoming one of the biggest reasons hampering employee retention. Kronos Inc. and Future Workplace® conducted a study with human resources leaders and 46 percent of respondents said employee burnout is responsible for up 20-50 percent of their annual workforce turnover.[6]

One of the top three causes of burnout is excessive overtime / after-hours work (32 percent) – two situations common to on-call staff. Another study by Blind, a popular anonymous community app, focused on tech workers and revealed that 57 percent of 11,487 tech workers believe they are burned out.[7]

According to the Mayo Clinic, companies spend $300 billion annually for healthcare and missed days as a result of workplace stress. The problem is so severe, that the World Health Organization announced that its next version of its handbook on diseases will recognize burnout as an “occupational phenomenon” that could drive people to seek medical care.[8]

A more in-depth discussion of burnout can be found on the Atlassian blog: https://www.atlassian.com/blog/productivity/work-burnout-symptoms-and-prevention

Many tech companies look at being on call as part of the job. However, on-call work is considered an inherently stressful task because of the uncertainty it adds to an on-call worker's life. A recent study focused on measuring the cortisol level to determine whether extended work availability has an effect on stress. The authors concluded that forced availability has a significant effect on the daily start-of-day mood and cortisol awakening response, even if they are not called to work.[9]

An effective on-call program built on best practices can significantly reduce staff stress and go a long way towards making on-call work sustainable for the health of the employees and the success of the company.

## Good on-call practices lead to hign preforming developers and teams

There is an obvious need for 24/7 on-call coverage for most software companies. Applying best practices can not only make on call tolerable but can also lead to other benefits.

On call improves the work product of developers by providing the opportunity for them to experience and learn from new challenges. On-call engineers must care about how to operate the software and should have the operations skill set to triage, diagnose, and fix problems. They need to understand not just the code itself but also the overall system, architecture, monitoring and deployment of the software. Many experienced engineers fully promote this idea as a way toward career development.

Software developer and avid blogger Julia Evans has discussed the benefits of developers handling on-call duties such as understanding different parts of the system that engineers don't normally experience, gaining confidence in judgment, and learning to better design reliable systems.[10]

Engineers also benefit from on call by promoting the idea of ownership. Companies want their people to take responsibility for the work they do. On call is an effective catalyst for creating a culture of ownership among the staff. Developers who participate in on call care more about nonfunctional requirements and often produce high-quality code that is reliable, secure, and observable since they are ultimately the ones who will deal with the problems when things go wrong – sometimes in the middle of the night.

The New York Times technology editor Susan Fowler made an effective argument for developers being on call saying that software engineering is about making mistakes, taking responsibility for those mistakes, and learning from them. On call involves those things as well.[11]

Good on-call practices also have a tremendous impact on team relationships. Carefully crafted policies emphasize the importance of on call and reinforce that everyone must take part in carrying it out. This can relieve the pressure from a select few engineers and operations staff who have been assigned the tasks. This joint effort increases collaboration and empowerment across teams and increases the effort to make on call a natural part of DevOps.

Handling the on-call function is daunting, regardless of who is ultimately assigned to the task. Building a solid on-call program by focusing on established principles can go a long way to alleviating stress and delivering effective services to customers.

## Summary

As more companies become reliant on applications for critical elements of the business, reliability and uptime become crucial. When incidents arise, customers want to know that the issue is being handled and will be resolved in a timely fashion. Establishing an efficient, responsive, transparent on-call program is critical to providing the trust that will ensure you remain competitive in the market.

# 02

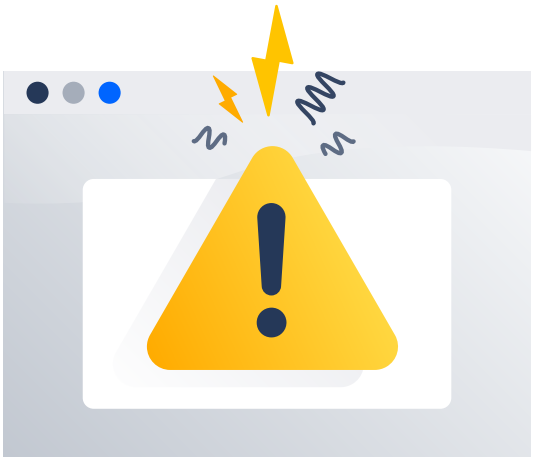
# Choosing your model

# Choosing your model

## Introduction

Bugs, outages, and incidents can stress your entire organization, especially the customer service and engineering teams. Setting up an appropriate on-call program to handle issues as they arise can yield significant benefits – both for your internal teams and more importantly, for your customers.

The design and development of your on-call program will have a huge impact on your organization's ability to respond quickly and efficiently, and ultimately on your ability to meet customer expectations.



## Establishing your on-call program

Many startups and small companies put their most senior technical people, often the CTO, on call in the early days of the company. Companies that have only a few developers who have intimate knowledge of the system and the product code are able to make quick fixes without the need for documented processes. One- or two-person on-call teams can usually handle any issues that arise. This solution works quite well–until it doesn't. This type of "on-call hero" approach can't scale as companies grow. Priorities change, complexity grows, and alerts increase over time.

As product adoption increases, it becomes necessary to develop a structured on-call program to handle issues. There are many ways to structure an on-call program and it is important that you choose the one that best fits your company's resources and culture. Things to consider as you start the process of designing your on-call teams include:

**Resources:**

- How many people are available for on call?
- Do you have enough people for sufficient schedule coverage without inducing burnout from too much after-hours work?
- Can you have dedicated on-call resources or will people be handling alerts in addition to their primary job?
- Does your team have adequate skills, or will you have to design a training program to fill gaps?

**Product:**

- How mature is the product and what is the rate of change?
- How stable is the system of services?
- Do you have a clear idea of the types of incidents you'll likely encounter or is the product still new enough that there are many unknowns?
- Is the product mature but you still lack enough data to know the types of issues your on-call team may encounter?
- How is your product deployed: cloud, on-prem, or hybrid?

Once you have the answers to these questions you can start to consider the most appropriate on-call team model for your organization. Your model needs to be flexible enough to evolve over time since change in the software industry is constant and your on-call team structure and responsibilities will likely evolve as well

In this chapter, we will look at two types of on-call team structures: centralized and distributed as well as the groups within your organization that might be best suited to handle the on-call function. We will explore the advantages, disadvantages, and practical considerations that will help you decide which structure is right for your organization.

## Potential on-call groups

Before discussing the different types of on-call teams, it's important to examine which groups within an organization are typically selected to be part of the on-call team.

### Operations

Traditionally, many organizations have dedicated systems admins or Ops teams responsible for running IT operations. Although this group traditionally didn't work on the software product in the past, that is changing somewhat today.

### Developer/Engineering

This is the team that works on the software code and is sometimes responsible for testing and quality assurance. In some cases, Testing and QA is handled by a separate team.

### Site Reliability Engineering (SRE) team

A relatively new role popularized by Google, site reliability engineers are software engineers who design, code, and maintain an operations function.


Centralized Ops team is on call while Dev teams are not.

## Centralized vs. distributed on-call teams

On-call teams can be designed as centralized or distributed, regardless of which group you ultimately choose to handle the function. A centralized on-call team is a dedicated group of people with skills and experience in the operations of software products and environments that responds to alerts regarding bugs and incidents. They're often called the Ops Team.

In some cases, organizations outsource this operation to a third party that specializes in their software space. Whether in-house or outsourced, the key differentiator for this type of team is that they're solely responsible for being on call with no responsibility for coding or software development.

### Advantages of a centralized on-call team include:

- Fewer people to train and manage compared to a rotating team of people who need to be cross-trained and scheduled around their other responsibilities
- Fewer special on-call compensation plans to design and manage
- Fewer management issues than with a distributed team
- Development, over time, of more experience with the types of incidents your organization is likely to encounter
- More acquired learning of the “ins and outs” of the automation scripts and the likely causes of common incidents.
- More likelihood of meeting SLAs due to more experience dealing with incidents.

### The disadvantages of a centralized on-call team include:

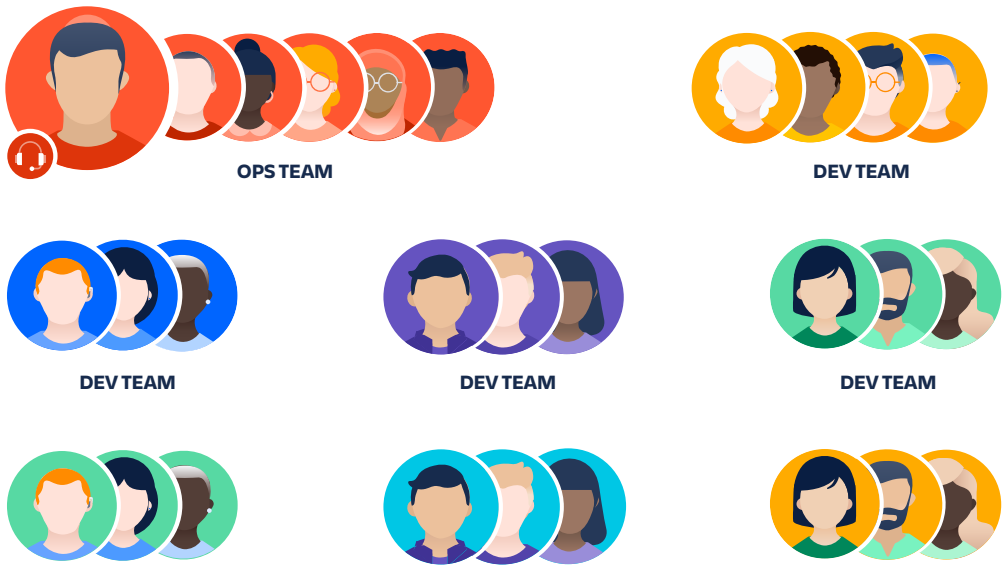
With limited exposure to the software code, teams may struggle to quickly identify the contributing factors of an incident.

- While dedicated teams may be able to execute some automated actions, they are more likely to require escalating to someone else in the organization to solve the incident resulting in response time delays.
- Outsourced dedicated on-call teams may not have the means or inclination to share data other than response time and SLA metrics. This prevents you from gathering valuable information about types of incidents, specific responses, and other data points that inform training, software improvements, and help prioritize other work.
- Chances of burnout increase as fewer people are dealing with many problems.

A more subtle risk involves potentially creating a conflict between the Ops Team and the Development Team. Developers who are not exposed to alerts and incidents involving the software may not be as interested or incented to make sure they're deploying only working, tested software. When something goes down the Ops team must take the brunt of the front line response which can cause friction between the Ops Team and the Software Developers.

This kind of conflict across the organization demoralizes all employees, bogs things down, and creates cracks in fragile systems that can impact the company's ability to release new and better products.

As companies grow and need to scale their response efforts, a dedicated operations team becomes less effective. Ongoing staff turnover makes it harder to have effective knowledge transfer, especially if there are issues with the engineering group. As companies expand, the product improvement feedback loop between engineering and the Ops Team can break down preventing the product managers and developers from critical information that can help them set priorities to develop a more stable system.




Large Ops team is on call while more Dev teams are not.

With a dedicated, centralized response model, as the company scales its product teams, it must also grow its Ops Team. As the Ops Team expands, reaching a high level of system reliability with a large group becomes a challenge. It's also difficult to keep up with hiring, training, and retaining talent in an ever-expanding group.

In addition, as companies scale, there is a significant increase in operational complexity. Ops Teams will have difficulty handling the exponential growth as they become buried in alerts that they may not even understand.

Today, leading tech companies realize that scaling the Ops team relative to product growth isn't feasible unless they focus on reducing manual repetitive work and distribute some on-call responsibilities to product teams to relieve the workload.

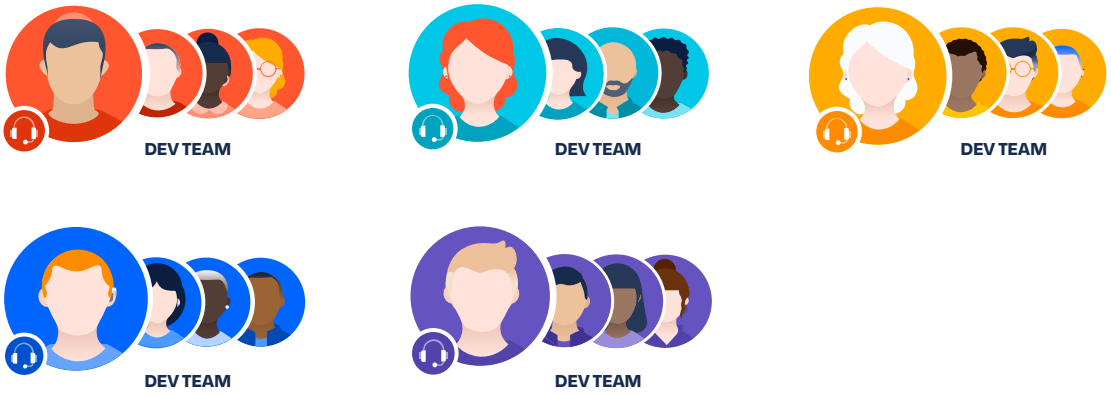
## Distributed on-call teams

With a centralized Ops team, many critical concerns that are important to the success of the company are placed into one group in an isolated one silo. This can lead to breakdowns and scaling problems.

To take advantage of the extensive knowledge base in the company, you may instead opt for a distributed on-call team. This approach can play an important role in achieving Ops understanding throughout the company. When all teams participate in on-call work, everyone is better able to assess the importance of investing in practices that will help maintain the product.

Companies may follow different approaches to a distributed team depending on their needs and where they are in their growth trajectory. There are currently three popular types of distributed on-call teams being used in the industry today.

> **"Everybody from tech support to product people to CEO participates in your operational outcomes, even though some roles are obviously more specialized than others.**
>
> CHARITY MAJORS, FOUNDER AND CTO OF HONEYCOMB.IO



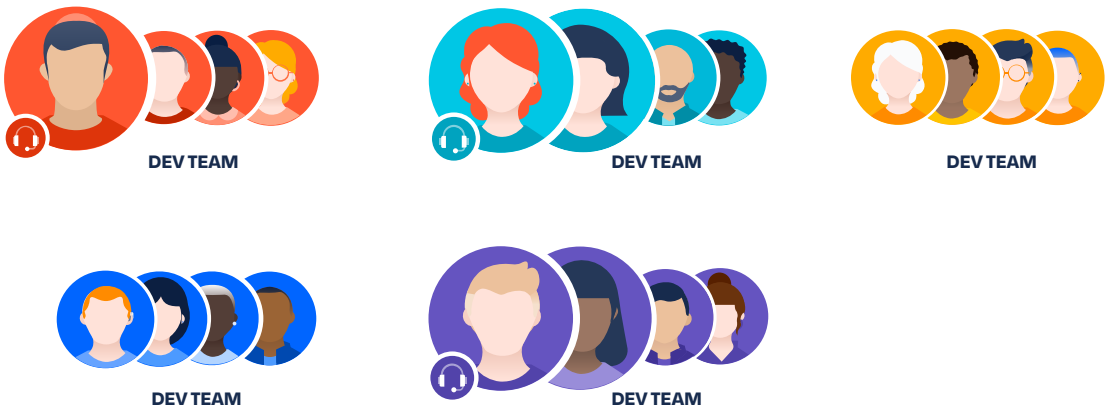

Embedded Ops engineer is on call within each team.

## Embedded on-call teams

In this model, the team divides the responsibility for operations and development and assigns the on-call duties to one or more embedded members of the team. This subgroup, often called DevOps engineers, takes the operations role and becomes responsible for automation, deployment, and alerts.

Ops Engineers are an integral part of the team, attending daily meetings and sharing incident data gathered from their on-call work. This can positively impact the product roadmap and help the development team build a more resilient, reliable product with fewer issues.

Combined operations and development teams often have up to 10 people with the majority being developers who routinely participate in incident response and other on-call work. Senior developers get involved in on-call work as secondary responders when escalation is required. When the team grows beyond 10, the embedded Ops group may get too far from the code. This results in the team facing the same challenges as centralized on-call teams.




Engineers volunteer for on call.

### Voluntary on-call teams

In a volunteer on-call approach, some members of the combined operations and development team volunteer – with pay - to handle on-call work. They are typically developers who sign up for a shift, with operations people available to prioritize the work as it comes up. Volunteers do this work outside of normal business hours.

**Advantages include:**

- The on-call work is distributed across a number of people from different teams without expecting everyone to participate. Members are not mandated to take extra hours
- Fewer people on-call simplifies company policies regarding extra hours work.
- Allows individuals to enhance their skills.
- Builds camaraderie among people who wouldn't otherwise get a chance to work together.

**Disadvantages include:**

- Difficulty managing on-call work in a diverse environment with different projects and totally different tech stacks, even within the same products
- As the company grows, complications arise trying to schedule potentially hundreds or thousands of engineers who want voluntary shifts.
- Requires a strong commitment from each team member to do all of the work that comes to them in their shift, including required follow-up.
- Lack of scalability of the model since as a company grows, assigning on-call work to make sure all shifts are covered becomes unwieldy.

**Volunteer on-call teams in action**

Intercom, a SaaS customer messaging platform, formed a "virtual" on-call team that consisted of 5 - 7 volunteers per shift.[12] Engineering was still on call for their products during business hours, and members of the virtual team were on call after hours. The company rotated out the team every six months after a week or so of actual on-call time. The key to the success of this plan was gaining a commitment from individual engineers to take ownership of each alert and see it through to resolution. As a result of implementing this plan, Intercom saw a reduction in the amount of alarms and the number of people required to schedule for on-call shifts.

In another example, Monzo, the fast-growing digital bank, leveraged their nationwide engineering team to solicit volunteers for on-call work. The company pointed out that the drawbacks are a model that uses all developers. They noted that when developers all serve on on-call teams in fast-growing companies, it is harder to fairly compensate each person and care about each engineer's well-being while still maintaining high quality levels of incident response. The model Monzo selected is to pay on-call engineers £500 per week and allow them to take any time off to compensate for the impact of being on call. This compensation model makes sure people feel valued and encourages engineers to show more interest in solving incidents.[13]


Everyone participates in on call.

### All-Hands (including developers) on call

In Companies of all sizes, including leading tech companies like Atlassian, Amazon, Google, and Netflix, expect all engineers to take on-call responsibilities to varying degrees. For example, while Amazon focuses on full ownership and expects developers to take on-call responsibilities, Google follows the principles of Site Reliability Engineering (SRE) and expects a healthy relationship between SRE teams and service teams (more on this later in the “Site reliability engineering approach to on call” section, below).

The concept of “all-hands” ownership of on-call work has become popular in the software industry since Amazon CTO Werner Vogels shared his philosophy of running service on production: “You build it, you run it.” Amazon creates autonomous teams and makes them fully responsible to manage everything they create, including on-call work. They even include teams that develop tools for other developers to deploy, monitor, and maintain their software such as the Ops Team.

Developers working on call is becoming more popular as companies of all sizes – including developers themselves – realize the significant benefits of this approach, including:

### Faster incident resolution

Developers don't just write code, they build things. All code that is deployed to production requires maintenance and will create problems so who better to understand and diagnose issues with the product than the team that created it? Putting developers on call often means accelerated time to resolution since they are intimately familiar with the code.

### Higher quality software

Developers who take the responsibility of supporting their creations see the problems firsthand. This increases their concern with things like debugging, testing, monitoring, logging, and incident response. This helps developers sharpen their engineering skills and better understand the business and results in higher quality and more operable software. It is important to note that putting developers on call is not a punishment, but instead an opportunity to provide them with insight that will help them focus on new features and more effective testing before deployment.

### Encourages cooperation between developers and managers

The "All-Hands" approach helps develop collaboration between managers who are under pressure to meet demanding release schedules and developers who want to build robust and maintainable systems. As expectations rise, developers might tend to start pushing more untested, undocumented, and often unmonitored code leading to more problems on the operations side. When development teams are responsible for responding to alerts in an operations role, development managers can clearly see the business impact of "fast code" and may result in developers getting the appropriate amount of time to create high-quality code.

There are some disadvantages to the "All-Hands" approach to on-call staffing. Increased training, compensation issues, scheduling, danger of lack of tracking and follow-up of incidents are real issues. So is the case where some developers may not be happy with taking on-call responsibilities during out-of-office hours leading to excessive turnover.

### On-call teams in the cloud era

Before the days of SaaS and cloud computing, traditional operations work had little to do with writing code.

Cloud computing has been transformational across many disciplines in IT, enabling innovation, cost savings, scalability, while simplifying IT management. It also speeds up the rate of change.

Change is the number one cause of incidents and outages. Cloud computing enables developers to change code and deploy it to production faster and more often (and, sometimes, without testing and other safeguards), and this means more incidents. That puts companies in varying degrees of constant failure mode.

This shift to developing and deploying software quickly via the cloud now has operations staff building products for developers–instead of end users–enabling the engineering team to directly deploy code to production. As responsibilities shift, Ops and developer teams must work closely together, not just in software development, but also to manage the systems that are running it.

Important in this joint effort, which we now call DevOps, is making sure there are people who are on-call and available to respond to incidents as they emerge. In fact, this work, and the resulting feedback loop, can be critical to team success. Developers are no longer responsible for just making the software, they're also accountable to take on-call shifts as well as fixing problems when they arise. The operations team is then responsible for making sure developers deploy those fixes as quickly as possible.

With this "break down the silos" approach to both software development and incident management, teams can fulfill the true promise of the cloud.

## Site reliability engineering approach to on call

Another approach to service management is Site Reliability Engineering (SRE), developed and popularized by Google and defined as the implementation of DevOps.

> **Google's VP of Engineering and father of SRE, Ben Treynor, defines site reliability engineers as software engineers who design an operations function.**[14]

Site reliability engineers write code for services that support software developers, but they are still responsible for the reliability of production systems. To be able to achieve the required level of reliability, SRE's are given significant authority to implement policies and changes.

In this distributed model, both SREs and developers take on-call responsibilities, and everyone is responsible for supporting the tools they create. This approach differs significantly from the other on-call models we've discussed in that SREs are specifically responsible for the stability of the services on which the organization relies.

In the SRE model, teams measure each service's reliability based on service level objectives (SLO), with the goal of achieving a level that deems the service "stable." Google's Maps service, for example, is measured by SLO and a team of SREs is responsible for its on-call duties.

In the case of new services that have not yet reached a defined level of stability, developers are on call for services that are new, under heavy development, and undergoing a high rate of change. These developers often rely on SREs for best practices that will help them lift their services to the level of reliability they need to reach stability. Once the SREs have stabilized the services, developers can request SRE support for on-call duties. The important difference here is that SREs that handle on-call work are empowered to give on call back to developers if service isn't as reliable as it should be.

## Summary

How you choose your on-call approach should depend on your business realities. If the product is mature enough, the SRE team or Ops engineers can take over on-call during business hours. If engineers have experience in running systems on production, they may be best suited to handle on-call responsibilities. If the system is stable, on call is less of a problem. All these issues go into helping you shape the decision for your on-call approach.

Another factor affecting the approach to on call has been the changes in where software is created and the way in which we approach operations. Cloud and DevOps have been transformational across many industries. Traditionally, there was no need to consider who needs to be on call since it was usually the responsibility of system admins or Ops engineers. But that has changed rapidly in recent years.

As we've seen, there are a number of ways to handle the important on-call function. Software-based systems change all the time, and so should our teams. Starting with a dedicated Ops team doesn't mean you can't experiment with on call in a product team. Start small, gather data and feedback, and iterate from there. The key is to be always evaluating your model so you can reinvent it to meet the demands of your clients as you gather data and receive feedback.

# 03

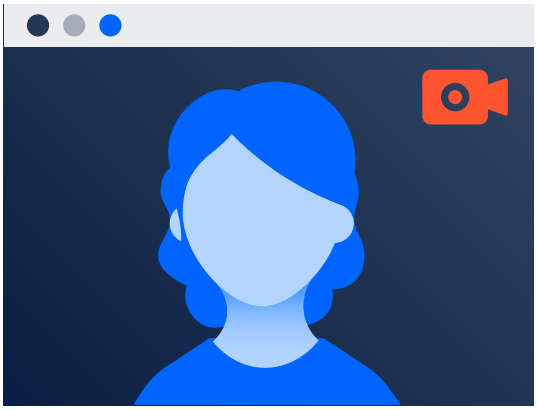
# Establishing roles & responsibilities

# Establishing Roles & Responsibilities

## Introduction

On-call teams require individuals to fill multiple roles with varying responsibilities. Roles vary from active participants to consultants who get involved when on-call teams need more information to resolve an issue. Having defined responsibilities for each role is crucial to maintaining a healthy on-call program.

Lack of clear expectations can create a host of problems, most critically missed alerts leading to serious production outages. Additionally, on-call team members who live in a constant state of the unknown are at risk for burnout.



## Typical roles for on-call teams

The titles may vary from company to company, but typical roles involved in an on-call program include:

- Team leader
- Primary on-call engineer
- Secondary on-call engineer
- Managers
- Customers
- Customer support

The organization's CTO or senior managers who have experience running on-call teams typically define the duties and expectations for each role and obtain buy-in from the broader organization. The roles and responsibilities should be published in a version-controlled application like Google Docs or Confluence and available for all employees to comment and provide feedback. As practices and conditions evolve, individuals can make further comments and propose changes to roles. This level of transparency sets the baseline for a collaborative, cooperative on-call program.

Following are common definitions for each role, primary responsibilities, and its relationship to the rest of the on-call team.

## On-call team leader

The on-call team leader is often a senior member of the organization with solid leadership skills and hands-on experience running applications in production. The primary responsibility of the leader–or team of leaders, as is sometimes the case–is to define the procedures, goals, and metrics that will guide the program. An important responsibility of the Team Leader is to minimize – as much as possible – the impact being a member of the on-call team has on the personal lives of individual team members.

The primary responsibilities of the Team Leader are:

- Defining the responsibilities of primary and secondary on-call roles and establishing on-call procedures.
- Providing training and guidance for on-call team members.
- Setting up the on-boarding process for new on-call engineers.
- Analyzing the collected on-call data and using it to improve on-call performance.
- Helping to determine appropriate response times for alerts.
- Creating a set of priority levels for categorizing various alerts.
- Setting the on-call team schedule and establishing the policy for after-hours and holiday coverage.

## Primary on-call engineer

The ability to successfully respond to incidents depends on the preparation done beforehand. Best practices for the primary and secondary on-call engineer start before actual incidents occur:

### Pre-incident

The ability to successfully respond to incidents depends on the preparation done beforehand. Best practices for the primary and secondary on-call engineer start before actual incidents occur:

- Ensure preferred notification channels are open and operative.
- Ensure access to devices needed to respond to alerts such as a laptop and mobile phone with the required software installed.
- Ensure reliable internet connectivity.
- Ensure the ability to add override to redirect alerts.
- Make every alert a top priority.
- Participate in on-call and incident response training.
- Create a Jira ticket for non-urgent work or when a higher priority alert needs attention.

In the event the primary on-call person isn't able to respond to alerts for a period of time, the secondary on-call engineer should be prepared to take over.;

## During incident

On-call policy documentation should be concise in defining expectations from on-call engineers during incidents. General responsibilities of on-call could also include details such as the expected time to acknowledge alerts with different priorities. Such policies allow large teams to have a common understanding and reduces the toil [link to Google SRE book on the subject]. Other responsibilities during an incident include:

- Acknowledge the alert or incident and start the investigation.
- Own the alert. The person who originally responds must ensure the issue is resolved. During shift changeover, the alert should be properly documented and transferred to the next on-call engineer in the rotation.
- Determine the urgency of the alert.
- Triage the issue or issues.
- Expeditiously escalate according to the playbook when necessary.
- Collaborate with the entire incident response team in case of major incidents.
- Log important events.
- Once the issue is resolved, investigate the contributing factors, or create and assign a ticket depending on time constraints.
- Fix broken monitoring queries and dashboards.
- Create a ticket for recurring issues.
- Fix small issues resulting from alerts –the low hanging fruit–as they arise to minimize unnecessary, time consuming noise.

## Post incident

On-call policy documentation should be concise in defining expectations from on-call engineers during incidents. General responsibilities of on call could also include details such as the expected time to acknowledge alerts with different priorities. Such policies allow large teams to have a common understanding and reduces the toil [link to Google SRE book on the subject]. Other responsibilities during an incident include:

- Share positive and negative feedback with on-call managers.
- After major incidents, conduct and document postmortems as soon as possible and let others know about any changes made during the shift.
- Update docs and runbooks to keep them up-to-date.

## Important notes for primary and secondary on-call engineers

- There is no need to answer every alert immediately. Missed alerts can be handled through escalations.
- On-call teams should have the flexibility to work anywhere they have online access and where they can respond to alerts.
- Primary and secondary on-call engineers do not need to solve every issue by themselves. On-call engineers should be encouraged to get help to ensure they are taking the right actions. Assigned secondary on-call engineers can help solve issues as well.

## Secondary on-call engineer

Best practices require one or two backup engineers for each on-call shift. Escalation alerts can be routed to secondary on-call engineers when alerts aren't answered or when the primary on-call engineer needs help. Secondary on-call engineers assume the same responsibilities as the primary on-call person when involved with an alert. Additional responsibilities of the secondary on-call engineer:

- Offer help to the primary on-call engineer whenever necessary such as when another incident is already in progress.
- Be ready to step in temporarily if the primary on-call engineer isn't available.

## Managers

The management team makes the final decisions affecting the on-call program in the company. Putting operational best practices in place requires strong support from the top, and on call is no exception. First and most importantly, management should acknowledge that building a proper on-call program takes time and effort. As technology leader John Allspaw has said, incidents are unplanned investments. Management is responsible for several key elements of on-call:

- Support having a formal on-call program in place.
- Ensure on-call team members have appropriate hardware and software.
- Compensate on-call team members appropriately considering legal implications.
- Ensure the health and safety of on-call team members.
- Provide a training budget.
- Ensure sufficient team size for effective on call.
- Assign at least three levels of escalations to ensure critical alerts are never missed.
- Periodically take an on-call shift to maintain an understanding of the current on-call work experience.

## Customer support

Best practices require one or two backup engineers for each on-call shift. Escalation alerts can be routed to secondary on-call engineers when alerts aren't answered or when the primary on-call engineer needs help. Secondary on-call engineers assume the same responsibilities as the primary on-call person when involved with an alert. Additional responsibilities of the secondary on-call engineer:

- Communicating the status and resolution progress of incidents to customers.
- Managing customer resolution expectations during an incident.
- Providing customers with truthful information during an incident without compromising the reputation of the company.
- Keeping customers appraised of steps being taken by the on-call team to resolve the issue and when they can expect an update.
- Sharing customer feedback with the on-call team in a timely manner.
- Creating alerts with different priorities when necessary

It's helpful to share information between customer support and the on-call team using the established tools for communication and work tracking.

## Importance of transparency in on call

Regardless of the role, one of the key aspects of successful on-call operations is transparency. In a business environment, transparency means sharing information to foster a sense of trust. Transparency in the on-call environment is critical both for all stakeholders both internal – developers, team leads, and on-call staff as well as external – customers.

On-call teams are often the first people to identify weaknesses in a product or service. They need to trust that they can elevate these issues without retaliation. In the case of incidents, much of what customers need to know will first come from the on-call team. In all cases, these teams need to know that leadership appreciates their own transparency and honesty and has their backs. On-call teams need to know that they're working in a culture of trust through transparency.

## Internal transparency

### Hiring

On-call requirements should be outlined clearly with every new hire to make sure they're willing to make the needed commitment. Details that should be communicated include:

- Frequency and length of on-call shifts
- Expected tasks and responsibilities when carrying out the on-call role before, during and after incidents
- Required locations for on-call shifts – office or remote

### Policy changes

Changes in on-call policies should be communicated promptly and should include the reasons for the change to avoid distracting gossip. Before implementing changes, getting feedback from the team and incorporating it where possible will go a long way toward getting buy-in. In large organizations, it is often difficult to gather input from everyone. In those cases, it may be possible to set high-level policies – for example mandating participation from developers in on-call rotations- then allowing the team to work out specific details.

### Documentation

On-call policies should be documented and available either in both physical and digital form for all team members. The documentation should be version controlled so the latest information is always available. Tools like Confluence or Google Docs are effective tools for keeping policies up to date and accessible.

## External transparency

Transparency must also extend to public communications to build trusted relationships with customers.

### Status pages

On-line status pages are one of the best ways to share updates with customers during and after incidents. On-call staff can manually input updates, but a more effective way is to utilize an automated solution that sends status page updates to both internal and external stakeholders. The goal is to let everyone know what's happening as soon as possible so that they can take necessary actions to protect their services.

Publicly sharing service status updates also reduces the burden on on-call team members by reducing the need to deal with update requests from upset customers. This frees the staff up to focus on fixing the problem. Providing a status portal with regularly updated information signals the team is working on solving the issue and helps set expectations for service restoration. The status page entry should include:

- Incident time and status
- Affected services
- Estimated time to resolution or restoration

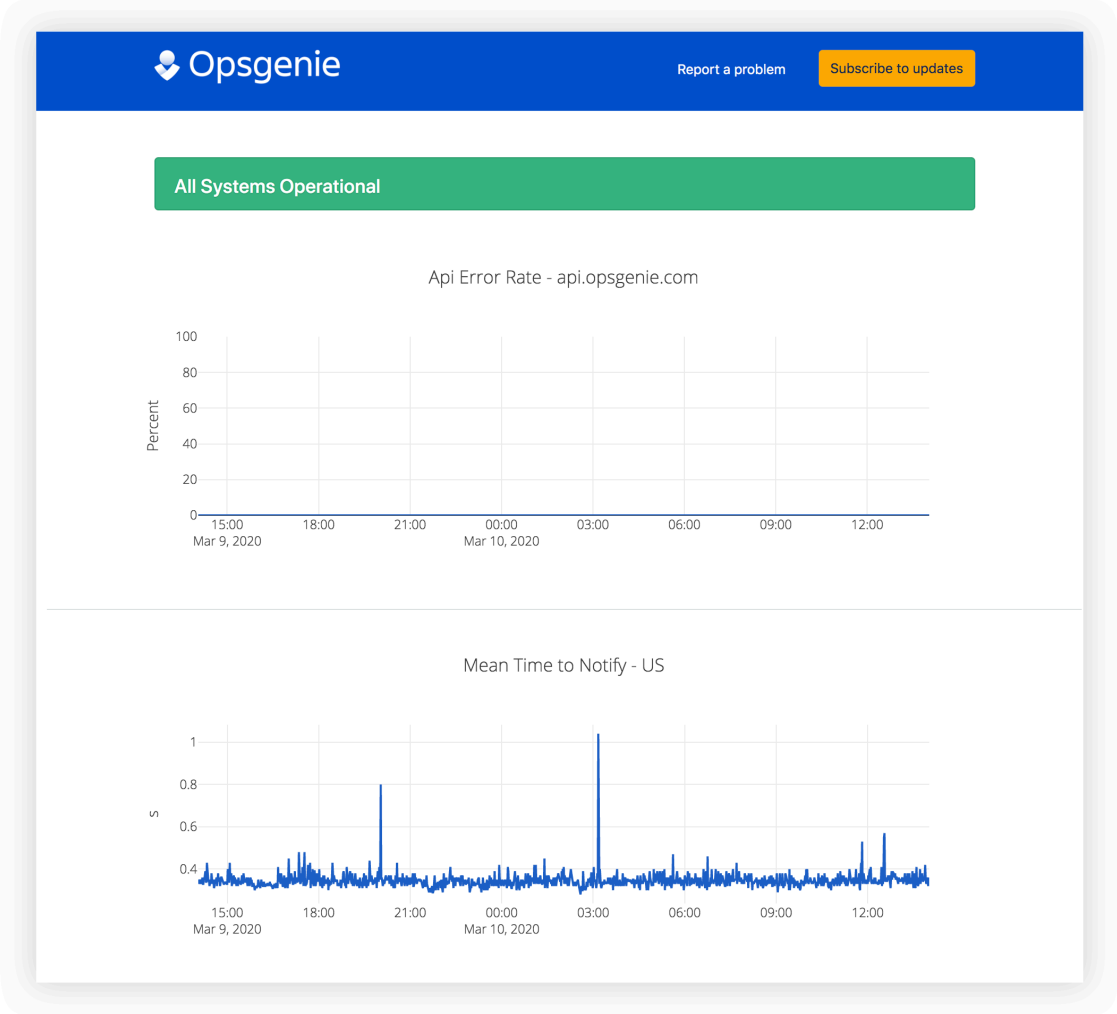
According to recent research, nearly nine out of 10 respondents said that following a bad experience, they are more likely to give a second chance to a company that has developed a track record of transparency. The same survey revealed that 85 percent of respondents are more likely to stick with the company during a crisis.[15] This, in the context of on call, is the kind of trust relationship that companies need to build with their customers.

### Real-time chat

Real-time chat provides ongoing customer communication that is critical for successful incident response. Solutions such as Intercom, Slack or Microsoft Teams are increasingly popular tools for companies to communicate with customers. They become crucial immediate communication vehicles for keeping customers informed and answering questions during incidents much more effectively than email.

### Dashboard metrics

Dashboards are often the first line of communication for on-call staff, allowing them to receive alerts, identify problems, and gather useful information as issues arise. Companies are increasingly sharing important metrics from these dashboards that may be of interest to customers on their status pages. Metrics could include an increase in HTTP response times - a key indicator of slowness. The Opsgenie status page is a good example of a comprehensive dashboard. https://opsgenie.status.atlassian.com/




## Summary

Defining the roles and responsibilities of each on-call team member ensures that alerts and issues are dealt with as quickly and thoroughly as possible.
A documented structure is the best way to make sure that the right people with the right expertise are on the task. This kind of clarity provides teams with direction that reduces confusion, uncertainty, and burnout.

# 04

# Scheduling your staff

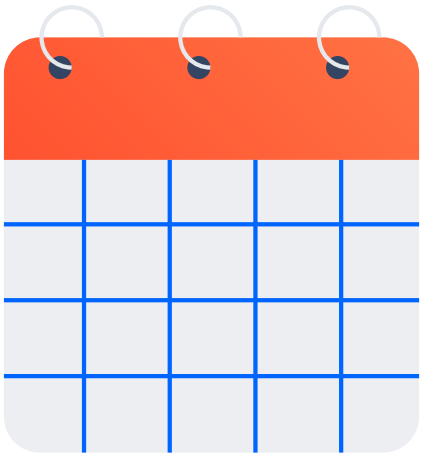
# Scheduling your staff

## Introduction

On-call teams are drivers of system availability and reliability and therefore critical to the success of an organization. To maintain an effective on-call team, it's important that team members are rested and ready to respond when an incident hits.

The key to creating a top performing on-call team is efficient scheduling. On-call schedules are used to ensure full coverage and determine who will respond to specific types of incidents.

There are different approaches to on-call schedule planning, so it is important to select the style that will be most beneficial for your team.



## Scheduling considerations

There are a number of factors to consider when designing an on-call schedule

### Team input

It is particularly important to enable teams to have feedback into the scheduling model so you choose the one that works best for the individual members of the team. Managers should have enough freedom to set up schedules that allow employees to attend to family needs, take care of their health, or pursue training in their off-hours and to make needed adjustments to maintain a positive attitude and high level of morale.

**Why is on-call scheduling that considers the individual team members needs important?**

According to a study by the Economic Policy Institute, on-call employees with irregular schedules are more than twice as likely to experience work-family conflict than employees with typical schedules.[16] An on-call schedule that respects employees' personal lives keeps them from being overtasked, missing sleep, and ensures that the organization will receive maximum productivity while providing job satisfaction.

### Team size

The number of available on-call employees dictates your approach to on-call scheduling. The number of available individuals impacts how you address issues of daytime and nighttime coverage and ways to strike a balance between the organization's on-call obligations and the individual team members' personal time.

In small startups, founders and senior engineers often take all of the on-call shifts themselves. They know the entire system well so they can triage and fix problems on their own.

When there are two people available you can schedule weekly rotations with each alternating full weeks. Being on call and dealing with alerts for an entire week, however, can be exhausting.

Overnight on-call shifts can be more tiring on employees than day shifts. With two on-call team members, it is sometimes better to assign each to two 12-hour shifts so each team member can trade-off the overnight shift.

Scheduling becomes easier with three or more team members participating in rotations, allowing for periods of rest and recovery. More team members also allows for other shifting alternatives and even more flexibility.

### Client needs

Accounting for client needs is crucial when setting up your on-call program since on-call scheduling can have a direct impact on their profitability. According to research firm IHS, system downtime costs North American businesses $700 billion per year.[17] An effective on-call schedule that provides full, consistent coverage ensures you can provide your customers with timely support when an incident happens.

To ensure sufficient coverage at all times, review your site data to clearly identify the highest-traffic times, or times when you're likely to have the most support requests. If you are short-staffed, configure your schedules to ensure coverage at peak times. In any event, you must ensure some level of coverage 24/7 if you run always-on services.

# Implementing your on-call schedule

## Selecting the right scheduling option

There are a number of options from which to choose when establishing your on-call schedule.

### Group agreement

Your team may be able to work out a schedule among themselves that takes into account their personal needs and preferences while meeting all business and customer requirements. Involving them in the process can improve morale and performance. New Relic has implemented this type of collaborative scheduling approach with great success.[18]

### Random selection

This approach assigns shifts objectively to the team. Team members may trade days among themselves to accommodate individual preferences and needs.

### Automated scheduling

With this option, an automation tool is used to create the schedule. Unlike the random distribution of team members, an automated tool can take into account personal needs and preferences, the expected number of page alerts for each time period, and other factors. Although fairer than the random option, automated scheduling is not objective. With large teams and multiple schedules, it is difficult to please everyone. The effectiveness of an automated system relies heavily on the amount and detail of the data input into the tool.

## Shift options

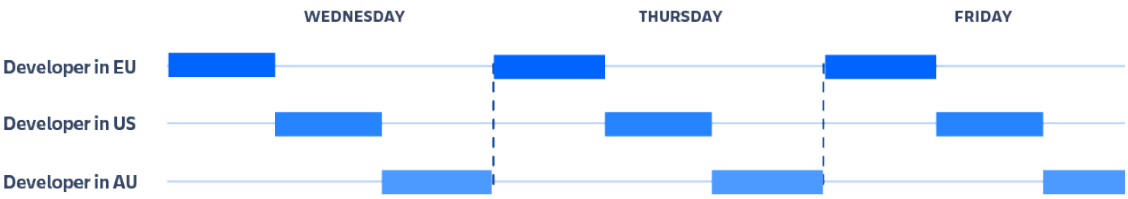
There is no one-size-fits-all approach when it comes to establishing shift options, but there are three important points to consider.

- Shift lengths should leave time for people to deal with any follow-up activities such as debugging the cause of an errant alarm or writing a postmortem after an incident.
- The on-call team should have enough time between shifts to relax and recover
- Whenever possible, prioritize scheduling on-call shifts for business hours to reduce team fatigue.

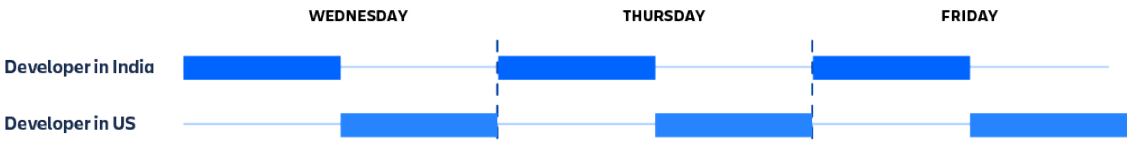
### Follow-the-sun shifting

Organizations that have distributed teams throughout the world have the ability to follow this shifting approach to provide around-the-clock coverage, especially if each location has staff who can be dedicated to on call during their regular business hours. This scheduling approach minimizes the need to schedule team members to off-hours.

As the name suggests, this approach deploys three rotations of on-call engineers staggered to cover three 8-hour shifts- one shift each in Europe, the U.S., and in Australia.



In the example below, there are on-call engineers in India and in the U.S. Each engineer has an on-call duty for 12 hours a day.




Some members of distributed teams will have specialized skills or knowledge of a particular system. In this case, it may make sense to put them on the overnight on-call schedule as a backup in the event of an escalation.

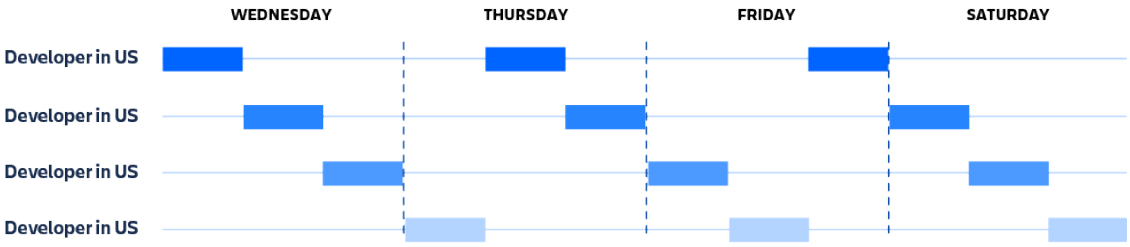
Some companies without facilities or offices in various time zones may still have remote employees scattered across the world. These companies can deploy remote workers in a similar way to use a follow-the-sun-schedule, though making this work for more than three time zones can be challenging. Keep in mind that remote employees may be accustomed to having significant flexibility in their schedules, so consulting them to determine optimum on-call shifting is helpful.

### 8 or 12-hour shifts

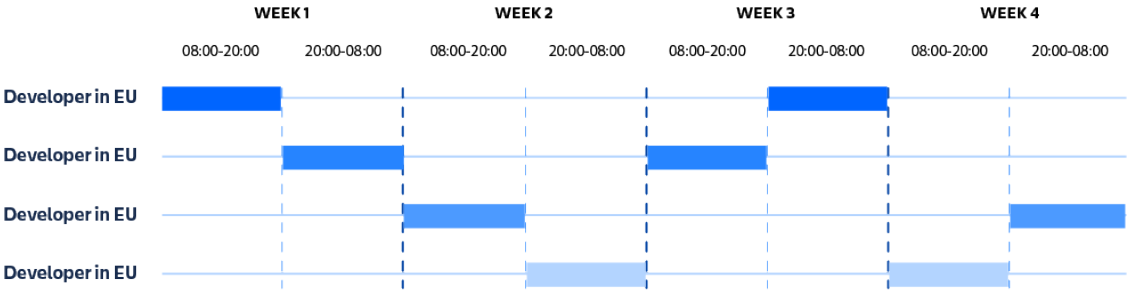
For on call, 8 or 12 hour shifts work best on a team with three to four people. The biggest advantage of this shift length is that it results in someone almost always finishing on call during business hours. This enables a smooth handoff to the following shift members.

One notable disadvantage is handover time. Shorter shifts mean people spend proportionally more time of their shift on handover and some issues might not get sufficient attention or are left incomplete.

Here is an example of a team of four rotating every day with 8-hour shifts:




Most 8 or 12-hour shifts are scheduled for a full week. Here is an example of a team of four that rotates every week with 12-hour shifts:




Week-long schedules are often a more acceptable option than having team members rotate every day. Having multiple weeks off between rotations helps people fully recover.

### 24-hour shift

Full 24 hours shifts is a common approach but requires at least four people to implement successfully. It is often easy to complete work on time, handover issues to the next engineer, and make notes for changes or improvements to the system. The biggest disadvantage is the possibility of an inordinate amount of middle of the night alerts which can cause significant fatigue and reduce engineers' productivity the next day.

### Full week shift

The biggest advantage of a full week of on call is that there is more likely to be a break between rotations, depending on the size of the team. For example a team of five would mean each team member would have no on-call duties for a month. However, full week rotations can be exhausting, especially with days that have many incidents.

**Shift lengths, backup teams, and handovers**

Team members with multiple on-call or lengthy shifts in a row lack the proper time to recover. Giving team members at least two weeks off after a full on-call week can be an effective way to have them reenergize, Minimizing time between shifts can also be a way to reduce fatigue and keep team members up to speed on changes in the system and eliminate the need for retraining. When using a daily shift program, there should be at least 3 days off between on-call shifts.

It's important to have backup on-call team members–ideally at least two–in the event that the primary on-call person can't be reached. We cover this in more detail in Chapter 3.

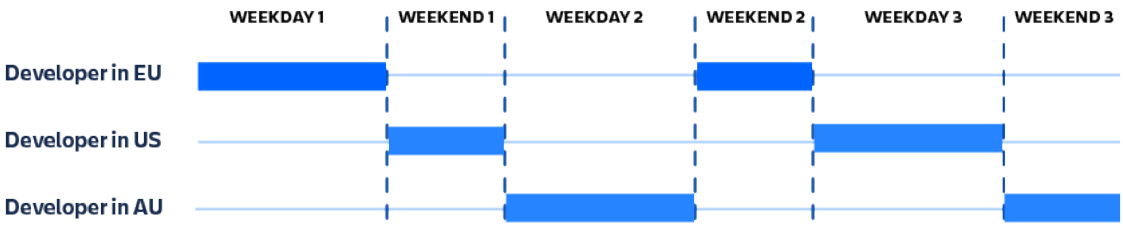
The requirement to be available at all times when on call can be extremely stressful and can lead to burnout. Automated escalation policies that ensure someone will be notified if the person on call isn't available can increase response time and relieve some stress from the primary on-call member.

Make sure there is some overlap in on-call shifts to allow for an appropriate handover between on-call engineers at shift change. This ensures that work is delegated and communicated properly. Various strategies for on-call handover are discussed in Chapter 3.

Backup staff can also be beneficial to help fix problems the primary on-call team member can't handle. The ability to escalate the alert to a more senior person, or one with specialized knowledge is crucial in this situation. Leveraging established escalation policies not only saves time but is also less stressful for primary on-call members.

### Separate weekend rotations

Weekend on-call shifts can be particularly difficult since they require team members to carry their laptops, have internet access, and be ready to work when necessary. The goal is that for the most part, team members can continue with their private lives, but be available to respond if issues arise.



### On-call schedule templates

Using templates to document your schedule is the best way to communicate with all team members. Your template should include:

- Users (who will be on call?)
- Rotation types (weekly, daily, or custom?)
- Restrictions (will you restrict on-call shifts to certain times?)
- Start date and time for the schedule

You can create templates using manual document apps like Confluence or spreadsheet apps like Microsoft Excel. However an automated scheduling tool that can be integrated with other on-call applications is most effective. Templates created with automated scheduling tools offers the most flexible approach since automation helps avoid scheduling conflicts and records changes automatically.

## On-call scheduling best practices

### Communicate effectively

It is particularly important to enable teams to have feedback into the scheduling model so you choose the one that works best for the individual members of the team. Managers should have enough freedom to set up schedules that allow employees to attend to family needs, take care of their health, or pursue training in their off-hours and to make needed adjustments to maintain a positive attitude and high level of morale.

Establish schedules at least a month prior to the on-call work. Communicating well ahead of time enables members to plan around their scheduled shifts. Individual situations may come up so be flexible enough to accommodate changes.

Send out calendar notifications the day before and an hour before each shift starts and shortly before a shift ends. This helps team members prepare for their shift and complete the necessary handover to the next on-call worker prior to the end of their shift. Ideally the notifications should be sent automatically using an on-call scheduling app like Atlassian Opsgenie. Significant schedule changes should be quickly communicated to the entire team and each team member should have visibility to the entire team schedule.

## Be flexible

Allow primary on-call team members to divert alerts or pages to the backup person on call when necessary. Keep a record of these changes to ensure proper compensation.

## Develop a collaborative culture

Having a supportive team can make a huge difference in both employee satisfaction and on-call effectiveness. If someone has a particularly active night, encourage other team members to step up and offer to take the next shift. If personal emergencies or significant life events come up, team members need to know they have a support system that takes over for them. Fostering the kind of culture where team members take care of each other can significantly lighten the burden of on-call work.

## Ensure continuous improvement

Your on-call schedule doesn't have to be done once and then set in stone. Check in regularly to ensure that the schedule is working as well as possible for the team, helping prevent and resolve issues quickly, and is as truly effective as it can be–both for customers and for staff.

Collect feedback from team members through polls, forms, and during retrospective meetings. Every two or three weeks hold an on-call meeting to review alerts and schedules. Use data to spot potential issues that people may not feel comfortable sharing and take corrective action. Make appropriate changes to the schedule design based on the feedback you collect. Be aware that on-call schedules will change as your product and your business evolves.

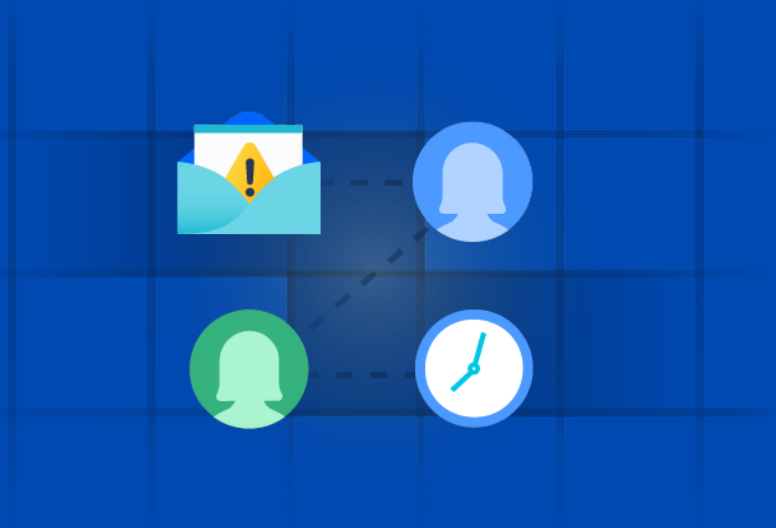
**Healthy schedule checklist**

Building schedules the right way is important for an effective, collaborative on-call program. This Top Ten list of best practices will help you optimize your on-call schedules.

1. Share schedules at least one month prior to on-call.
2. Rotations should be no longer than two weeks and repeating on call no longer than two weeks.
3. Daily on-call time is no longer than 12 hours.
4. The next consecutive on call is at least 3 weeks away.
5. Rotations overlap for half an hour for proper on-call handover.
6. Backup schedules indicate who should be on secondary on call and contacted in the event of an escalation.
7. Team members can request overrides from the teammates for immediate needs.
8. Send prompt notifications for schedule and rotation changes.
9. Record all schedule changes.
10. Discuss schedules with the team at least every month.

## Summary

On-call schedules have a significant impact on the morale and effectiveness of your team. The key is finding the right balance between the demands of on-call coverage and the needs of the individual team members. Experiment with the suggestions offered here, collect feedback from your team along the way, adjust accordingly, and you'll find you have an effective, successful–and happy–on-call team.



# 05

# Developing the alerting process

# Developing the alerting process

## Introduction

On call's first notification of an incident comes through an alert. Many tools create alerts for any number of reasons. Alerts can become the best or worst friend of the on-call team depending on how they are handled.

Today's IT environment needs monitoring on different levels of the system and teams use many different tools to achieve visibility. Consolidating alerts coming from these multiple sources is a challenge that the on-call team faces on a daily basis. If alerts aren't actionable, time is wasted on response and resolution. Investing in your alert system is crucial to the success of your on-call program.

# Alerting best practices

## Events vs. alerts

It is particularly important to enable teams to have feedback into the scheduling model so you choose the one that works best for the individual members of the team. Managers should have enough freedom to set up schedules that allow employees to attend to family needs, take care of their health, or pursue training in their off-hours and to make needed adjustments to maintain a positive attitude and high level of morale.

Different on-call tools have varying terminology when it comes to alerts, events, incidents, notifications, pages, and other terms. These labels are used to define different but closely related things.

In this book, we define an event as a change while an alert is a change that requires notification to someone to act on it.

## Informational alerts

Events and alerts go hand in hand. On-call team members may choose to generate an alert through an automated monitoring system for informational purposes. Care should be taken, and alerts should be created only when a response is required. Many alerting tools provide ways to filter, tag, and suppress notifications for informational alerts. Leverage those features whenever necessary but think twice before sending an “informational alert.”

## Choosing symptoms to monitor

Uptime has become key to running a successful online business. Most businesses need to catch problems before customers are affected.
The decision to alert on-call teams for potential problems has become an important challenge. Separating alerts that need immediate attention - such as slow page response time, raising 5xx HTTP responses, or issues that impact critical path processes - from ones with less urgency is critical for effective on-call response. Defining and leveraging Service Level Objectives (SLO) can help manage levels of urgency.

The SLO is a fundamental concept in Site Reliability Engineering (SRE), Google's modern approach to operations. As outlined in Google's SRE book, SLOs define a target value or range measured by an indicator such as latency or availability. "Error-budget" – another key SLO term - indicates the number of failure times before a team no longer meets the established SLO objectives. For example, an SLO goal to return HTTP response less than 100ms for 99.9% of the time over 30 days, return responses that take more 100ms time for 0.1% time are acceptable.

The Google SRE process defines SLO thresholds using six different methods – target error rate, increased alert window, incremental alert duration, alert on burn rate, multiple burn rate alerts, and multi window, multi-burn-rate alerts. Reliability metrics like SLOs are important for determining when to alert an on-call engineer.

Alerts should be created only when a problem needs immediate human action. The on-call team may still want visibility into potential problems so alerts can be marked low priority and automatically create a Jira ticket and can be handled at a later date.

## Alert creation

Software tools create alerts to notify users about monitoring, logging in, or potential security problems. In addition, alerts can come from continuous integration/continuous delivery (CI/CD), ticketing, automation, or chat tools.

Another common way to create automated alerts is via APIs. Depending on the use case, alerts may also be created directly from code. Automated alerting is the preferred way to detect software or infrastructure problems because they can be addressed and resolved before customers are affected. Automated alerting also offers greater context and debugging opportunities than manually-reported alerts. Some alerting solutions also enable automatically tying support calls to on-call schedules and routing alerts to the on-call team on duty.

Another way to create alerts is through the traditional ticketing method. Often customer service teams are empowered to create tickets. If they have access to alerting tools via APIs, chat rooms, or an alert reporter, they may choose to notify engineers directly. As the organization and activity scales, this type of manual alerting should be automated and tied into a ticketing tool. A ticketing tool integrated with an alerting solution can execute rules for the planned action automatically.

Regardless of their origin, all alerts should be collected in a central location, ideally using an automated incident response process.

## Leveraging contextual information

Adding tags and properties to an alert not only aids in effective routing but saves the responder time by providing a context for the issue. Less time is spent investigating and formulating a response plan when additional information is given within the alert. There is often no single source of information for determining the urgency and impact of an incident. Information is scattered throughout logging, monitoring, ticketing, and configuration management tools. Avoid context switching by providing supplemental details during alert creation.

## Creating actionable alerts

For an alert to be actionable, it must be clearly communicated and describe the urgency and scope of impact of the issue. Preferably, the alert directs responders in the right direction to avoid or minimize damage.

**Actionable alerts:**

- Notify the appropriate people at the right time.
- Suggest how to start investigating the issue with contextual information.
- Enable common tasks to be performed through automation.

## Define naming conventions

Naming conventions need to be clearly defined to enable quick action when addressing alerts and to help scale the process as the number of services and alerts increase. On-call team members are able to determine what is important faster during alert updates and incident response when using common definitions. Responders can see what is important at a glance and run queries that match certain criteria. Here is an example of a common naming convention:

***#AlertNumber [environment] [region-code] [detail]***

***#711 [prod] [us-east-2] Notification send failures for channel X increased %20 over the last 5 minutes.***

## Use tags

Tags are widely used in the operations world to attach labels to alerts for easier identification and categorization. Through the use of tags, on-call teams members can limit results and execute actions like searching alerts, routing alerts to a different team, and suppressing notification for a period of time. Tags can represent priority, API name, source, datacenter region, and others.

## Centralized alerting

An image showing alerts going to people from multiple alerting sources vs. an image showing alerts grouped under one tool and distributed from there. Designating a central place for all alerts benefits the on-call team in several ways including:

- Better assessing the blast radius of incidents through reporting and analytics
- Reducing noise by consolidating alerts
- Routing alerts to the right teams through preferred notification channels
- Providing a better view of the health of the system, services, and on-call process

Modern incident and on-call management systems offer a central place for all kinds of alerts. This consolidation is helpful for on-call teams to effectively deal with alerts. However it can create security concerns. It is imperative to assess risks and use well-defined authentication and authorization protocols between integrations to secure sensitive information that may be coming from monitoring tools or the alerting tool.

## Assigning alert priorities

Understanding the importance of an alert requires assigning an alert priority. A high priority alert may use a more aggressive escalation policy and notify team members within minutes, while a low priority alert might decide to defer alerting altogether. Priority-based alerting offers flexibility in routing alerts the right way.

You will benefit from creating priority levels like the one below. Adding priority color coding is also a common practice.

| Priority | Description | Example case |
|---|---|---|
| **P1**<br>**Critical** | More than 1% of users fail during the checkout process. | A significant problem affecting many users to purchase items. |
| **P2**<br>**High** | Many users can't access the items under the books category. | A significant problem affecting many users to purchase items partially. |
| **P3**<br>**Moderate** | Comments aren't working for many users. | A problem affecting user purchase decision or shopping experience. |
| **P4**<br>**Low** | A link redirects users to the wrong page. | A problem affecting a small number of user's shopping experience. |
| **P5**<br>**Informational** | Campaign emails don't work for a small number of people. | A problem affecting promotion opportunities for a small number of users. |

| Severity | Description | Examples |
|---|---|---|
| 1 | A critical incident with very high impact | • A customer-facing service, like Jira Cloud, is down for all customers<br>• Confidentiality or privacy is breached<br>• Customer data loss |
| 2 | A major incident with significant impact | • A customer-facing service, like Jira Cloud, is down for all customers<br>• Confidentiality or privacy is breached |
| 3 | A minor incident with low impact | • A minor inconvenience to customers, workaround available<br>• Usable performance degradation |

These table contain high-level information and act as a guide to help teams select alert priorities. They do not identify which individual alerts go into each category. The tables can be expanded or additional tables can be created to provide more detail.

For example, if an alert is triggered when “more than 1 percent of users receive a failed response when they click the purchase button,” the on-call team can refer to the table to determine its appropriate category.

Alerts may be placed into one priority level when they are created but may change following further investigation. On-call team members should be able to raise or lower the priority dynamically. Updating the priority level could trigger other rules that will help orchestrate the incident response with automated actions such as an auto-remediation action that adds more instances to a load balancer or pages the incident response team to obtain more help through video conferencing.

## Delivering reliable notifications

Reliable incident notifications play a critical role in on call. Missing or delayed responses result in wasted time and money. The alerting tool should have multiple back-ups for each notification channel. For example, if the primary SMS provider is having a problem, the secondary provider should be used. The tools should also offer flexibility in how notifications can be delivered including phone calls, mobile push notifications, or SMS messages, depending on the time of day.

The priority level is often an important factor in determining how a notification is delivered. High priority alerts should generate an immediate phone call. Lower priority alerts can be delivered via e-mail. Flexibility is important, but the process should include precautions to avoid mistakes and missed alerts. Setting global notification policies for critical alerts, for example, eliminates the possibility of error or missed notifications.

## Tracking alerts

It's important to track each alert as it moves through its lifecycle. Log each action taken for each alert as they occur such as:

- First notification sent
- Success/failure of notification
- Time on-call team member acknowledged the alert
- Who saw the alert?
- Who asked for the alert?
- Which stakeholders were notified via which channels?
- Which endpoints were informed through webhook?
- What actions were taken on the alert such snooze, escalate the next, etc.?

Tracking alerts is critical to improving the alerting and on-call process. This type of data can help identify on-call team members who need more training on certain notifications, level of response to alerts, the percent of alerts that need immediate action and more.

## Review alert reports

Alert reports or dashboards provide data visualization and analytics capabilities of alert activity. They are useful in spotting "alert smells" when the right queries are performed. Teams can spot problems in certain areas by reviewing metrics like alerts per source, alerts by tag, alerts per service, alerts closed by team, alerts acknowledged by user and more.

Set up a weekly or biweekly meeting to review these reports with the on-call team to look for areas of improvement and to develop corrective actions. Sometimes quick actions solve problems but often problems require deeper analysis and determining effective solutions takes longer.

## Alerting tooling requirements

Automated alerting tools can be extremely beneficial when it comes to handling alerts. However feature sets for these tools can vary greatly, so it is important to make sure the tool you select has the following capabilities.

| Feature set | Questions to ask |
|---|---|
| **Works globally** | Can I send notifications (SMS, voice, email) to almost anywhere in the world? |
| **Multiple notification methods** | Does the tool have multiple notification capabilities like email, SMS, phone, mobile app push?<br><br>Does the tools allow team members and stakeholders to quickly get up to speed on the incident? |
| **Source of truth** | Can team members and stakeholders use the tool to locate all other details of the incident and response activities? |
| **Timeline** | Does the tool aggregate a chronological timeline of key events? |

## Leveraging automation

Alerts enhanced with contextual information can be more powerful if responders can automatically retrieve supporting information with one-click actions. Tasks that can be automated include gathering investigative information, taking remedial actions, and automatically initiating communication and collaboration.

Modern on-call and incident management solutions offer many ways to reinforce responsibilities for on-call teams and other involved stakeholders. Strive to minimize human involvement as much as possible by automating on-call processes such as notifying an on-call team member, running a script, or sending status page updates to customers.

The benefits of automation extend well beyond the obvious use cases. A good example is deploying an application that overrides a mobile phone's Do Not Disturb (DND) mode. If an on-call team member forgets to disable DND, the on-call app will override the default setting during the on-call rotation. Similarly, smartwatches can deliver notifications directly to on-call members when they are away from their desks.

Team managers can leverage on-call management software to eliminate manual processes such as ensuring someone is on-call for a service or calculating the on-call members' compensation.

Modern tools allow on-call team members to choose their own notification channels like SMS, voice call, mobile push notification, or email. Some tools also allow on-call managers or administrators to define global overrides to ensure critical alerts are never missed because of a configuration error.

## Gather more investigative information

Automation can dramatically accelerate investigative actions to retrieve relevant information. Time-consuming jobs such as manually logging into a service and running a query can be automated using action buttons attached to the alert.

Retrieving application logs, attaching graphs, or pinging a server saves a significant amount of time during the investigation of the issue. In addition, automating the retrieval of this data using predefined queries and actions reduces the chances of human error.

## Take remedial actions

Having one-click remedial actions saves a significant amount of time by automating complex workflows. Repetitive tasks that can help resolve an incident can be defined as an action on the alert. Restarting a process or a server is a common remedial action example. Another example is adding servers to the load balancer to handle a spike in the request count. Automation makes sense for most repetitive tasks, but the key is providing flexibility for manual intervention when necessary so that people can still decide when to take critical action - especially when dealing with high impact issues.

## Initiate communication and collaboration

In case of a high-priority incident, incident response team members need to communicate using a reliable, unified tool that is easily accessed. A button on the alert which allows team members to join a call or video conference bridge is an example of alert enrichment. Enriched alerts help your team resolve incidents faster by allowing responders and stakeholders to collaborate instead of simply trading messages. Opsgenie's Incident Command Center (ICC) is another unified solution for bringing the incident response team together. Teams gain access to the alerts that are being worked on and can exchange information from an enriched view.

Enriching alerts can drastically improve mean time to resolve (MTTR). Alert enrichment is a continuous process that allows responders to build off of previous experiences to streamline workflows and response procedures. System administrators or operation engineers in network operation centers (NOCs) use these guidelines to automate as much as possible

## Notifying the correct people at the right time

Automated applications can be easily configured to review alert details, apply filtering to suppress noise, and send alerts to predesignated teams. State of the art tools enable users to introduce complex logic to their routing workflows. Using specific details and timing from the alert, users are better able to determine urgency and improve routing. For example, if an error is reported from a specific server that affects a user's "Gold Level Customers," a corresponding property or tag can be added to the alert specifying this information and route it with elevated priority.

## Maintaining alerts

Automated alerting helps avoid problems faced when alerting was handled manually by a group of individuals monitoring a dashboard. Automation can minimize alert fatigue by assigning the right priorities to create alerts properly and make them actionable. Maintaining alerts requires continuous learning and improvement.

## Runbooks

Runbooks – sometimes known as Playbooks – contain a list of the procedures and operations for on-call responders to follow. Having a documented runbook is key to reducing the MTTR. Runbooks are especially useful as on-call team members develop more experience working with the tooling and the on-call environment. They provide a useful "how-to" guide that can help responders stay calm during what is often a hectic environment.

Automating Runbooks helps keep repeatable tasks in one place and helps greatly when they need to be updated.

### Creating runbooks

Creating a runbook is a continuous process. Instructions must be precise so everyone is clear on the meaning of each statement. Using an agreed-upon standard format and keeping them in a central location will lead to increased use. Digitizing the runbook to enable responders to access it online is also beneficial. The runbook should be the place you keep all instructions that can't easily be attached to an alert. All runbooks should include information such as:

- **Symptoms** – Indications to look for to determine that there is a problem
- **Pre-checks** – What checks can I perform to be absolutely sure this is the right guide to follow?
- **Resolution** – What actions must I take to resolve the issue?
- **Escalation** – Who can I talk to in order to answer questions about a process?
- **Post-checks** – How can I be sure the issues are solved?
- **Rollback** – How can I undo my fix if necessary?

Here is a typical runbook Format:

| Stage | Details |
|---|---|
| **System Background** | When to use this runbook |
| **Assumptions** | Important assumptions such as the state of clusters, access patterns |
| **Procedure Steps** | Step by step instructions to follow to resolve the problems such as “disable shard allocation,” “shutdown nodes,” “bring up master nodes.” |

### Keeping runbooks up to date

Runbooks can easily misdirect on-call team members if they aren’t up to date. Updating runbooks should be a regular part of Jira tasks. Responders should also be responsible for updating runbooks if there is a missing or incorrect instruction. Runbooks should be documented using version-control to ensure responders are using the most up to date version. That is why teams often attach runbook links instead of full text-based runbooks. Tools like Confluence offer version-controlled rich text editors. Some can also execute scripts and retrieve content dynamically.

## Minimizing alert fatigue

Alert fatigue–also known as alarm fatigue–is when an overwhelming number of alerts desensitizes the people tasked with responding to them, leading to missed or ignored alerts or delayed responses. This issue is compounded by the fact that many alerts are false alarms. In the medical industry, research shows that anywhere from 72 percent to 99 percent of all clinical alarms are false. In security, one survey found that 52 percent of alerts were false and 64 percent were redundant.[19]

The operative term here is “false alarms.” The situation is similar to the boy who cried wolf in the familiar children’s story. After too many false alarms, responders are likely to start ignoring alerts, even the ones that warrant investigation. The result: potential outages and downtime.

Alert fatigue is a problem in many industries. Especially in healthcare where the consequences could be fatal. In 2010, a Massachusetts hospital patient died after alarms signaling a critical event went unnoticed by 10 nurses. The patient safety officials admitted that many reported deaths are caused by malfunctioned, turned off, ignored, or missed alarms.[20]

Another study found that monitors in a hospital's intensive care unit with beds for 66 adults generated more than 2.5 million unique alerts in one month. This number corresponds to 187 warnings per patient per day. Another key stat was that 88.8 percent of the 12,671 annotated arrhythmia alarms were false alarms.[21]

These are only two of the many studies that highlight the damaging – sometimes deadly – consequences of alert fatigue. The good news is that there are ways to minimize this pervasive problem.

Kishore Jalleda, head of production engineering at Yahoo's Publisher Product unit, has spoken about how his organization fixed alert fatigue by creating the right incentives. He assigned each team a "noisy alert budget" every month. Those teams that went over the budget were temporarily denied SRE support. Those that came in under were "rewarded" with world class SRE support – fast response rate and attention to every alert. The results were dramatic: a 90 percent drop in false alerts, five minute SRE response time, and an increase in uptime by a whole 9.[22]

Here are five steps you can take to minimize alert fatigue.

## Distribute responsibilities

With a small on-call team, alert fatigue is inevitable. One way to minimize that is spreading the on-call responsibilities – including adding developers into the rotation. Having developers help out in on call increases accountability, improves code creation, and often results in fewer alerts down the road. Automatically alerting using well-planned escalations that routes the issue to the appropriate team reduces having every alert go through the on-call team. Distributing responsibilities means that teams will ultimately deal with fewer alerts, minimizing the stress on a small on-call team.

## Set clear reliability targets

Each team should have its own reliability objectives, or what SRE teams often call service level objectives (SLOs). Setting them requires understanding each service and its importance. Use caution when choosing SLOs that can be tied to business metrics. For example, 90 percent test coverage may be an important metric for the reliability of service for the team, but it doesn't mean anything to clients, who are usually interested in metrics like availability, error rate, request latency, and system throughput.

Once objectives are in place, the next step is tying those objectives to the right incentives to create a culture of ownership. Site Reliability Engineering practices are a great way of dealing with this problem – as shown in the Yahoo example.

## Create alerts with care

Uptime is key to running a successful online business – most organizations need to catch problems before customers are affected. The decision to alert engineers of potential problems is a common challenge. Separating alerts that need immediate attention from those that don't is also critical for maintaining a positive on-call experience.

Problems such as latency, raising 5xx HTTP responses (error rate), or, in general, any failures that might exasperate internal and external clients fall into this category. We can leverage SLOs to define what is important by ensuring that alerts have the right priorities set according to the severity of the problem. If there are many alerts that seem similar to each other, no one can focus on what's important. Reference the key principles of actionable alerts, such as routing them to the right people at the right time, providing context, giving clues to start investigating, and solving the issue.

## Collect data and regularly review alerts

To identify an alert fatigue problem, don't wait for employees (or customers) to complain – collect data. Tying everything to a consolidated alerting solution like Opsgenie allows you to trace the alert's lifecycle as well as evaluate on-call and service performance. These tools provide key metrics such as alerts per priority, daily created alerts, alert sources over a period of time, mean time to acknowledge, mean time to resolve, and more. Over time, this data can be used to assess various indicators of alert fatigue.

Based on your findings, respond with actions like adjusting the thresholds, changing priorities, alert grouping, and deleting unnecessary alerts. For example, if there are many alerts that can be traced to no action or no impact, get rid of them.

On-call team members may occasionally miss alerts, so it is important to monitor response times and take action if they're not within SLAs. When there is a concern about response times, managers should discuss the reasons with their on-call team members to learn how to avoid similar situations in the future.

## Create an "effective alerting" culture

Culture change is key in truly solving the alert fatigue and management buy-in, and support is imperative. Share key statistics with management stakeholders to justify the extra effort needed to work on alerting. Once management is on board, introduce useful routines to create the alerting culture that all on-call teams need. There are a few ways to do this: regular meetings with the team to go over alerts, reducing repetitive alerts, and holding engineers responsible for taking corrective actions that are long-term solutions. Finally, always remember to ask for feedback from teams on a regular basis to get a pulse on how they're feeling. Every team is unique in its own way, and you can use that information to the team's advantage to create an alerting culture that works for them.

Remember, fast fixes won't solve the problem of alert fatigue. Taking proactive actions for each of the principles outlined above helps get at the root of the issue. Start small by introducing these five tactics slowly. Then, after implementation, regularly review the alerting reports and, most importantly, create an effective alerting culture. The fight against alert fatigue isn't a short one, and it requires everyone's participation and learning from mistakes and achievements.

## Summary

Applying best practices in alerting is key for a healthy and successful on-call program. Automated and centralized alerting offers opportunities to create alerts in a flexible manner that works for all team members. Alerts need to be created using the right symptoms and setting the proper priorities to reduce false positives that may cause alert fatigue. Alerts need to be actionable, include contextual insights to reduce, trigger investigative and remedial actions, and include collaborative team communication to reduce MTTR.

Maintaining an effective alert process is a continuous effort. The alerting process needs to scale along with the systems. Defining naming convention, using tags, tracking an alert's lifecycle, and reviewing alert reports with regular meetings are key to maintaining a successful alert process.

# 06

# Setting the escalation path

# Setting the escalation path

## Introduction

In an on-call setting, escalation is the process of notifying backup team members, more highly technical engineers, or managers to ensure that incidents are addressed as quickly and effectively as possible. An escalation is triggered when an issue arises that can't be properly addressed by the first-line on-call team member or members. Escalations are an integral part of any proper on-call setup.

Constructing the escalation process properly is key to on-call success in driving reliability, productivity, and team morale.

# Importance of escalations

## Reduce notification noise

Since escalations typically only allow for initial notifications to be sent to a subset of the on-call team, they can be quite useful in reducing notification noise. An automated alert system is aware of the scheduled on-call member and notifies him or her of the incident first. If that team member does not acknowledge the alert, the system is programmed to notify the next designated team member. In each step of the escalation, the responders have the option to notify certain other responders, like the secondary on call, a senior engineer, or even a CTO. This process minimizes a general notification to the entire team which can cause confusion as to who is handling the incident.

## Ensure critical issues are addressed

The escalation process avoids the problem of a “single point of failure” by scheduling backups that ensure ongoing coverage. In the case of a critical incident where minutes matter, the escalation policy may likely override the backup and immediately notifying more senior team members.

# Reasons for an escalation

## Alert not acknowledged

If the on-call team member doesn’t respond to the alert within the specified time frame, the next rule in the policy escalates the alert to another team member.

## Alert not closed

If the on-call team member fails to close the initial alert in the system within the allotted time period an escalation alert will be triggered to close it.

## Inability to respond to an alert

If the on-call team member can’t respond to the alert for any reason, he or she can escalate to another available team member or manager using the predefined escalation rules .

### Alert response required higher level of knowledge/experience

If the on-call team member does not feel they have the requisite knowledge to handle the initial alert, he or she can escalate to a more experienced person. This is a common scenario that new on-call engineers are likely to experience. Yet, often this is due to lack of experience and misrouted alerts. This type of escalation can be avoided in an automated system with established rules that initially routes each alert to the appropriate person.

## Escalations policies & procedures

Escalation policies establish the procedures for instances when the primary on-call team member is unable to handle the alert. A policies and procedures manual spells out the way in which escalations should be handled.

Less is better when it comes to designing your escalation policies and procedures. Creating complex flows can introduce errors that might inadvertently escalate alerts to the wrong or too many individuals, or worst case, cause critical alerts to be missed.

Once developed, it's important to share your policies and procedures with all on-call teams so everyone handles escalations in a standardized way. This makes it easier for your on-call teams to provide consistent service.

### Time

The policy should detail the optimum time for execution of an escalation. The time lag depends on the severity level of the alert and the established level of system reliability. A low severity alert may go unanswered for 15 minutes while serious alerts may require escalation after three minutes.

Larger organizations that provide services to a wide range of clients with different priorities may employ a more structured escalation policy tied to a negotiated service level agreement (SLA) with specific service level objectives (SLOs), a concept popularized within the Site Reliability (SRE) movement. SLOs could include targeted HTTP response rates.

## Condition

The escalation policy should outline the required action or condition that determines the alert has been addressed and the escalation process can be terminated.

## Responsible party

The policy should detail the responsible party to be notified at each escalation step. Here is a sample table that outlines an escalation policy.

| Time | Condition (Rule) | Responsible Party |
|---|---|---|
| **Immediately** | Outage detected, first alert sent | On-call engineer in Ops schedule |
| **5 min.** | If the alert is not acknowledged | On-call engineer in Backup Ops schedule |
| **10 min.** | If the alert is not acknowledged | Ops team |

## Setting severity levels

The severity, or priority, level of an alert is established by the organization based on external and internal needs and is embedded as an integral part of the alert rules in the alert monitoring tool. Priority levels determine the urgency with which alerts are to be handled.

One common format uses five priority levels broken into three escalations. The following chart explains the escalation procedure for each priority level.

| Priority Level | Escalation |
|---|---|
| **P5 (informative)** | No one. Make alerts available without notification. |
| **P4 - P3 (moderately important)** | Ops_team_escalation<br>0 min. -> Notify on-call users in Ops_team_schedule<br>10 min. -> Notify on-call users in Ops_team_backup_schedule<br>20 min. -> Notify Engineering Manager |
| **P2 - P1 (important)** | Ops_team_critical_escalation<br>0 min. -> Notify on-call users in Ops_team_schedule<br>5 min. -> Notify on-call users in Ops_team_backup_schedule<br>10 min. -> Notify Ops_team<br>15 min. -> Notify CTO |

Escalations can be directed using routing alerts or triggers depending on their urgency. Routing rules are conditional checks to route incoming alerts based on the content of the alert. Routing rules are different from escalation rules. Escalation rules, which are conditional, are used to determine who should be notified as an issue escalates. A routing rule relies on the content of the alert to determine how it is routed.

## Routing the escalation

Many tools allow alert tagging with custom data inputs. These tags might include key-value pairs or freeform text. Priority levels and routing rules should be incorporated into your automated on-call tool. It's important to use a consistent tagging system, using only required data to minimize system complexity, scale, and maintenance.

## Ensuring backup

Multiple escalation rules ensure that you have enough backup people to eliminate any gaps in the escalation process. The number of backups typically depends on the severity level of alerts. For high severity alerts, there should be at least three escalation rules involving at least three different individuals.

## Major incident escalation

High-priority incidents require a specialized approach and should be quickly escalated to the incident response team. The incident response team should meet as soon as possible – virtually or physically – to discuss the incident and execute a formal response plan that, ideally, has been previously rehearsed. Escalations of this type typically cannot rely on automated alerts or responses since specific individuals must be called in to resolve the issue.

## Empowering the service desk

In a traditional IT organization, developers or operations engineers acted as the last line of defense in an escalation. Customer issues typically went through Tier 1 Support, then Tier 2 Support and sometimes to the IT operations engineers- a lengthy process that can take a substantial amount of time.

Today's fast-paced world with demanding IT reliability targets, and detailed monitoring drives more frequent incidents. As the first recipients of these incidents, the customer support team should have a tool and system that designates the appropriate on-call team members or engineers for each of these incidents. This process eliminates the gap between customer service and engineering.

One approach is to integrate the tools for ticketing and incident management. Tickets with specific tags and priority levels notify pre-designated team members. The critical step in this process is making alert updates available to the service desk so they can relay status updates to customers. An alternative approach would be to create status pages for internal stakeholders to follow the process after the escalation of the incident.

## Types of escalations

Escalations can be triggered by an automated system or manually by an on-call team member. Regardless of how the escalation is generated, you should develop clear escalation policies that make sure the right alert goes to the right person at the right time. A documented escalation policy is key to accelerating response time and "notification noise."

ITIL, which provides a library of IT service management best practices, introduces two types of escalations: functional escalation and hierarchic escalation.

Functional escalation is the process of escalating the incident to a second level support group if the first point of contact cannot resolve it. The second type, hierarchical escalation, is used when incidents are a high priority, and there is a need to notify appropriate IT managers and executive management about the incident.[23]

In the DevOps environment, the escalation process is used to determine who should be notified next, in which order, and in what time frame.

## Benefit of automated escalations

In this era of real-time operations, sometimes more alerts originate with automated monitoring systems than with a customer ticket submitted via the service desk. As discussed in Chapter 5, Developing the Alerting Process, there is tremendous value in automating the alerting and escalation process. A manual system that relies on electronic documentation collecting electronic dust on a shared file is doomed to fail – often at the most critical times. Relying on on-call team members to manually escalate incidents by calling someone from a long list of names is a recipe for disaster.

The automation tool you use should be flexible enough to reroute automated escalations to alternate users or teams with a simple click. This is especially useful when the on-call team member knows that the problem can be solved faster by someone else or when he or she needs to add another responder to the problem or escalate to an incident response team. Manual re-assignment or delegation of an alert is also helpful when the first contact is involved in another incident and needed to escalate the subsequent alert to the next on-call team member.

Policies for re-assignment or delegation should be clearly defined, documented, and updated in the system's alerting logic.

## Summary

An effective escalation plan clearly defines the roles and responsibilities of each person on the team and lets them know when they need to be available. Clearly outlining the escalation process can significantly reduce alert noise across the organization and enable the right people to focus on the right issues at the right time. Most importantly, a good escalations plan can help you reach your reliability goals, keep your on-call teams happy, and deliver superior service to your customers.

# 07

# Selecting the right compensation model

# Selecting the right compensation model

## Introduction

Compensation models for on-call pay vary based on local laws, company culture, and management practices. Choosing the right one is critical to building a motivated, effective on-call team. An effective on-call compensation plan is one that ensures your company has appropriate incident coverage while recognizing the efforts and time of your on-call team members. On-call team members who feel their time is being valued and respected are more likely to care about the business and contribute to its success.

# Typical on-call compensation models

## Incentivized on call

The Incentivized on-call model compensates employees for on-call hours with a combination of extra days off, flexible hours, and higher salaries.

Hosted Graphite's co-founder, Charlie von Metzradt, says that in addition to higher salaries for SREs, the company provides an automatic and mandatory day off to compensate for hours worked in on-call shifts.[24]

Providing incentives for on-call team members tends to increase ownership of projects and services which helps build a resilient system that leads to successful handling of customer issues. It can also reduce alert volume if responsibilities are properly shared between service owners and developers. Providing extra time off and paying competitively also lets employees know their work is valued and appreciated.

## Pay for scheduled overtime

In this model, on-call team members are directly compensated for the time they are on call or are scheduled to work whether or not they are called on to respond to an alert or an incident. The obvious advantage of this model is that on-call team members know that they are getting direct, immediate compensation for carrying a pager, cell phone, and laptop and for being available during off-hours.

## Pay for time spent on issues

Under this model, on-call team members are paid only for the time they spend responding to and resolving issues. Compensation can be in the form of an hourly rate based on time, or for the number of alerts or issues worked.

While on-call team members are compensated for the extra time and work performed during off-hours, this model removes the incentive to respond to and resolve alerts quickly.

### Combination scheduled overtime and time spent

This model provides base pay for being on call as well as additional compensation for alerts received or issues worked. The advantage is that on-call team members feel well-compensated for the extra time and effort while on call and provide additional pay for complex issues that require extensive investigation and resolution. One drawback to the model, as with the "time spent on issues" model is that it may remove the incentive for quick response and resolution.

## Determining fair and competitive compensation

To determine a compensation model that is fair to your on-call team members while being competitive in the marketplace, you should first determine the level of on-call coverage you need. This requires taking into account several considerations.

### Hours spent being on call

The number of hours required for on-call work may be extensive considering the different variations for different teams, context switching, overrides for urgent work, and complex incidents involving multiple on-call members.

### Number of alerts received on- and off-hours

This number is critical to determine the extent of your on-call requirement such as overnight, after hours or simply business hour coverage.

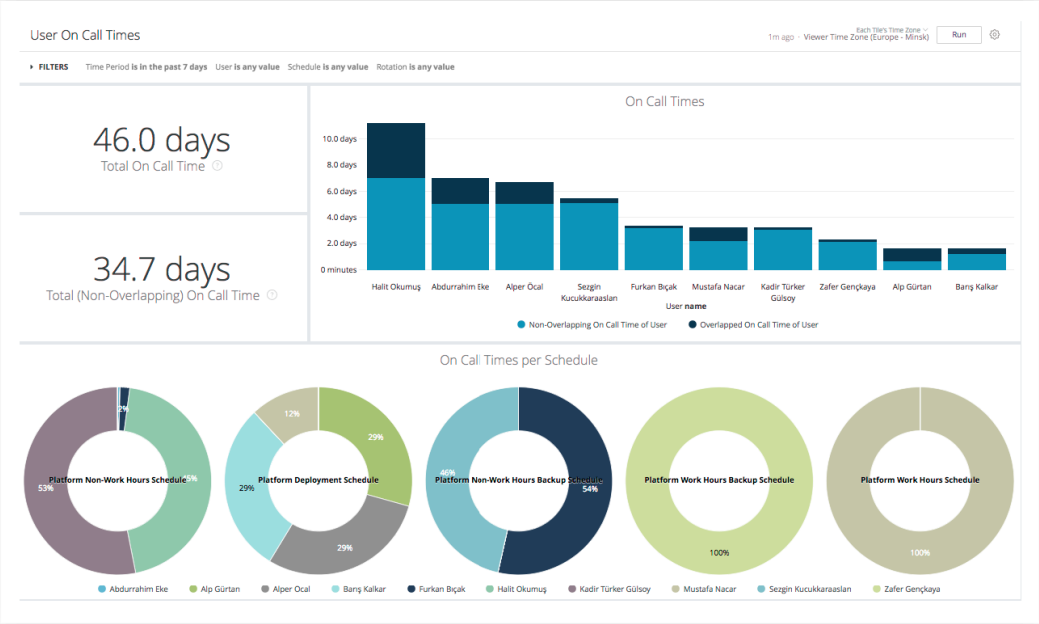
### Time spent working on incidents

The complexity and importance of incidents can vary greatly from several minutes to many hours. Understanding what your clients require is crucial to determining your on-call staffing needs and ultimately your compensation model.

### Meantime to acknowledge or resolve

It's important to determine the necessary response time for your customers in order to create the appropriate incentives and compensation model. Measuring the mean time to acknowledge and resolve over a period of time helps managers decide on the appropriate model and required incentives.

While measuring hours spent on call, alert volume, time spent on incidents, and MTTA (Mean Time to Acknowledgement) or MTTR (Mean Time to Resolution) are important to evaluate performance, it can be difficult to keep track of these metrics, especially for a larger team or across multiple teams. The process can be streamlined by using automatic metrics collection via on-call scheduling tools such as Opsgenie, Confluence, or Google Docs.




## Summary

Although monetary compensation is an effective motivator for on-call teams, it is not the only method. It's also important to foster a positive culture which can be achieved through flexible work schedules and comp time. Most importantly, never forget that on-call work can be grueling and interruptive to an on-call team member's personal life.

Showing genuine appreciation for the commitment to a successful program can be nearly as important in encouraging retention and motivating employees to make positive business decisions. If employees are well-cared for, they will, in turn, care about the business and contribute to its success.

# 08

# Training your team

# Training your team

## Introduction

Training is crucial to getting your team ready to handle the challenges of on call. Sharpening the team members operations and communication skills not only helps them meet aggressive uptime goals, it also helps reduce the inherent stress of the on-call role. Training should be a key focus for anyone who may be assigned on-call duties – primary, backup, developers, DevOps titles, SREs, or system operations.

A common practice in the software industry is for new hires to “deploy on the first day.” However, taking an on-call shift on the first day of a new job is challenging especially without being familiar with the tooling and the environment before responding to problems. The best approach is to start training on-call team members as soon as possible, in parallel with other training and aim for a first on-call shift in the first month.

Ultimately, the best way to learn about on call comes from actually doing the job. You can accomplish this through the practice of “shadowing,” where on-call team members can get a true feel for the experience. Combining training with on-the-job experience will help your team effectively handle the on-call challenge.

## Set the tone for training

Taking these initial steps will set the right tone to ensure the effectiveness of your training program.

### Establishing the on-call culture

Brian Chesky, CEO and co-founder of Airbnb says that “culture is simply a shared way of doing something with passion.”[25] To set the tone for effective training, it’s important to create a shared passion for on-call while also considering the needs of the individual members of the team.

### View incidents as opportunities

Incidents can be learning opportunities if you properly collect and analyze data. The focus of the analysis should be on determining what can be learned from the incident to help improve the product or process. Avoid stressing the failure aspect of incidents by assigning blame or giving non-constructive feedback.

Incidents are clear indications of your system and process weaknesses, not the weaknesses of your people. Conduct post-incident analysis that leads to long-term corrective action that ensures the same problem will not reoccur.

### Encourage collaboration and support

Nurture a culture where no one feels they are alone on the on-call battlefield. Automated escalations can help to ensure people have a back-up, but it’s more important that everyone knows that each team member has their back before, during, and after an incident.

### Accept ownership

Encourage team members to discuss the problems among themselves and take ownership to solve them. Don’t expect the next on-call team member to tackle the challenges. Improving the on-call environment and ensuring the overall reliability of the team is everyone’s responsibility.

# On-call training process

## Use of runbooks

Runbooks (or playbooks) are useful guides for on call. Runbooks include instructions for managing systems and the processes to follow when handling incidents. Becoming familiar with runbooks should be a critical component of the training process. We discuss runbooks more extensively in Chapter 5 Developing the Alerting Process.

## Monitoring, logging, distributed tracing tools

Every team varies in the tools they use to track operational health, application performance, and resource utilization. The training program should include making the on-call team member familiar with relevant diagnostic tools such as monitoring, logging, and distributed tracing tools. They should also have the necessary privileges, credentials and important ChatOps commands to access these tools.

## Assessing incident severity levels

The training program should include instructions on assessing incident criticality and assigning priorities. We discuss incident severity in more detail in Chapter 6 Setting the Escalation Path.

## Setting escalation paths

Modern teams automate escalation procedures to reduce delays but each team needs to be aware of the possible paths and the related teams. Your training program needs to include the process of escalation in on-call. We discuss incident severity in more detail in Chapter 6 Setting the Escalation Path.

## Communication during an incident

Effective communication is the key to solving complex incidents. When many people work on an issue - especially something stressful like an on-call incident - effective communication becomes more difficult. Train your team members on the use of your chat, voice, or video call tools for real-time communication. Practice real-life incident response scenarios through appropriate communication channels with multiple teams included. We discuss incident communication in more detail in Chapter 5 Developing the Alert Process.

## Post-incident handover, postmortems, reports, and feedback

Follow-up activities in on-call are critical since that is where the most learning takes place. On-call teams need training in the methods to apply good practices and use metrics and tracking tools effectively. Incident handover training should include the ownership of incidents throughout its lifecycle, key points to share, and all follow-up activities. Postmortems should include sharing of postmortem templates and tooling details, on-call reports and dashboards to help review the team's performance.

## Shadow training

The goal of an effective onboarding program is speeding up the adaptation process of on-call team members and making them more productive. In many companies, onboarding has become a challenge for both new hires and existing employees moving into an on-call role.

Companies used to wait for over a year before putting their people on-call allowing them enough time to become "experts." That is unrealistic in today's fast-paced environment where on-call team members must be deployed quickly. Although it is hard for a new on-call team member to respond to real alerts, they must gain some level of experience with the deployment process and the monitoring and alerting tools. That is why a typical expectation is for new on-call team members to be deployed by the end of their first month.

Shadow training helps speed up the process of getting on-call team members ready to participate in an on-call rotation. Shadowing is a practice similar to pair programming, a common practice that helps engineers learn from each other. This is accomplished by scheduling an experienced team member with a trainee at the same time.

In the shadowing process, the on-call trainee works side by side with the primary on-call team member. He or she handles incidents in a parallel, non-production environment while the primary team member actually handles the incident. Shadow users mimic what the primary on-call team member does without impacting the primary on-call engineer's duties.

### Shadowing example:

Jennifer is a senior engineer with over a year of on-call experience. Mike has just joined the team with the ambitious goal of joining the on-call rotation after three months on the job. For Mike to be able to do this, he needs to gain more experience with the environment. The team teaches Mike with shadow on-call shifts.

Once the shadow environment is setup, both Jennifer and Mike receive the same alerts. Mike doesn't make any changes in production but instead uses his read-only access to view the dashboards and run queries. As he has his own alert, he can acknowledge, resolve or take notes on the process. After the original alert is closed, Mike receives feedback from Jennifer on the incident. In the case of major incidents, Mike would join the war room channels or the video call as an observer.

**Start training with the daytime on-call shift**

Daytime and nighttime on call are different experiences. A new team member should take part in rotations as a primary responder after doing a series of shadowing experiences. The first few months as a primary on-call responder are usually more stressful because there will still be many unknowns. New on-call team members need to ask questions and usually hesitate to ask people during the night shifts, where there is normally a limited number of members. Training during the day makes it easier to find help without the need to wake someone up in the middle of the night.

### Incorporate training feedback

Training doesn't end when an on-call team member learns the organization's approach, practices, and tooling. Every on-call team member brings new ideas and experiences to the table. You should use them to improve your organization's procedures, runbooks, documentation, the training processes and responsibilities, and the overall on-call experience.

### Document everything so other people can learn

Most systems are so large that no one individual can see the whole picture. Documenting and updating systems and procedures and incorporating them into your training program is critical to creating a learning organization.

## Training for major incidents: Game Days

To properly prepare for a major, prolonged outage caused by a fire or other natural disaster, it's critical to train all team members–not just new and junior people. This training often incorporates the concepts of chaos engineering – the discipline of experimenting on a software system in production in order to build confidence in the system's capability to withstand turbulent and unexpected conditions. It's important to run failure experiments in production because that can provide many "fun" surprises that you won't get in a development environment. Running failure experiments in production could be challenging at first, so start small in the development environment.

Chaos Game Days are often associated with chaos engineering and can be effective in running failure experiments to prepare for major incidents. Many companies use Game Days both for SRE training and onboarding, and for anyone who will be handling the on-call team function. The benefits of Game Days go well beyond on-call and incident response and can be beneficial for development engineers to help gain necessary Ops skills that allow them to make systems more reliable.

One of the important points is deciding whether to run Game Day experiments in a development or production environment. NewRelic suggests that if you have SLOs in place with service level indicators (SLI) and you haven't exceeded your "error-budget," then you should run your GameDay in production.[26]

Running a Game Day begins by documenting the objectives of the exercise to help set up the right scenario and evaluate your success following the event.

The next step is to write the scenario. Usually, a senior team member suggests several challenging ideas and could include issues such as slow or failing DB queries, high HTTP 5xx error rate, or a failing Redis cluster.

Once the scenario is established, you are ready to assemble the team.
Remote team members can join using a chat app or incident command center.
Rather than scheduling the Game Day in advance, you can surprise the team and run the chaos engineering experiment without notice.

As part of the Game Day experience, include a checklist of items that align to the objectives. Following the exercise, evaluate the team's performance and develop action items based on the team's performance. Conducting and documenting a postmortem after the Game Day is critical to enable the team to review – and learn from - the events of the day.

Finally, have some pizza and end the day with more fun :)

The idea of running experiments on production became popular with Netflix's Chaos Monkey as the first chaos engineering tool. Since then many other companies have adopted this approach, making it simple to run your own GameDay. For a list of popular tools, check out this list from GitHub.

# Other training resources

## Internal documentation

Many organizations conduct training using version controlled documents and important information like internal policies, how-to-use guides, tooling manuals, and runbooks available on tools like Confluence, Dropbox, or Google docs. To use these documents as effective training resources, they must be kept up-to-date.

## Books

There are a number of excellent books on the on-call function that can be used as supplements to your training program. Two in particular referenced in this book include Incident Management for Operations by Rob Schnepp, Ron Vidal, and Chris Hawley, and Google's SRE book and workbook. These each have valuable lessons and guides for on call and incident response. The Atlassian Incident Handbook is also a valuable resource for help with effective incident management.

## Online courses

Online courses for on call and incident response are readily available from multiple sources. Look for ones that deal specifically with modern on-call and incident management practices. There are a number of Incident Management Courses listed on LinkedIn. Since on-call and incident response policies and procedures are often specific to each organization, many companies create their own online resources for their on-call teams.

## Podcasts

Podcasts have risen dramatically in popularity and several are directly related to on call, incident response, DevOps, SRE, and resilience. One of the best is On-call Nightmares hosted by Jay Gordon on which he talks with experienced technologists and relates stories of on-call experiences and lessons learned.

Podcasts are relatively inexpensive to create, and you can leverage them as part of your training program. You can record a number of incident response calls along with commentary from senior team members that highlight effective ways to handle certain issues.

## Atlassian Team Playbook

The Atlassian Team Playbook is a comprehensive example of a playbook. It includes step-by-step instructions for tracking team health and exploring innovative ways of handling on call and incident response. The Team Health Monitor assesses a team against eight attributes commonly found among healthy teams. The tool enables you to gain a better understanding of team strengths and ways to address weak spots.

The playbook includes step-by-step instructions for tracking your team's health and new ways of working (plays) that help your teams get things done. Once you determine the attributes of the team you'd like to strengthen, you use the appropriate plays to improve your team's overall health. There are plays for different team types and health monitor attributes.

## Game plans

Game plans are collections of recommended plays for specific purposes. The Incident Response game plan has four exercises that can be used for on-call training.

- Pre-mortem: Imagine all the reasons your project could fail and then figure out how you can prevent those problems while there's still time.
- Incident response values: Identify what your team values most during incident response and create a plan to live those values consistently.
- Incident response communications: Good incident response doesn't just mean fixing the problem – it means being transparent with customers as well.

## Games

On call and incident response are suitable topics for training using simple game playing. Many companies use games to improve skills like decision making, effective communications, writing blameless postmortems, and more.

Some games are open source and readily available to play. According to Google's SRE book, they use disaster role-playing games–sometimes referred to as "Wheel of Misfortune"–for SRE training. Pavlos Ratis created a website simulating this game.

At Monitorama 2018, Franka Schmidt gave a talk on how to build an interactive game to teach incident response. Schmidt suggests using a tool like Twine, a tool for making text-adventure games to create your own games.[27]

## Summary

Training is critical to on-call success. How you approach training depends on the size of your team and the complexity of your systems. But whether you're training a team of five or a team of 500 the most important thing to focus on is making your training program relevant, engaging, and continuously improved through trainee feedback. Teams aren't trained until they believe they're prepared and feel supported and the only way you can determine that is if you listen to what they have to say. Only then will you know whether your training program is successful.

# 09

# Learning through experience

# Learning through experience

## Introduction

Why is that statement so true? For those who have been paying attention to the evolution of how software is built and operated over the last decade, the reason is obvious: Change is everywhere and it is accelerating at a faster rate than ever. To keep up, businesses have to continually reinvent the game and unleash the potential of every individual and team in their organization.

How do we accomplish that when it comes to on-call teams? By building a culture of ongoing learning that analyzes failures, evaluates successes, and leverages insights to enable continuous improvement.

> **You are either building a learning organization or you will be losing to someone who is.**[28]
>
> ANDREW CLAY SHAFER, VP TRANSFORMATION, RED HAT & DEVOPS PIONEER

## Integrate continuous learning

Collaboration practices play a key role in building a learning environment for both individuals and teams involved in on call. Creating clear escalation paths and major incident response plans helps foster learning through collaboration. Shared chat rooms and on-call review meetings are also useful to create collaboration opportunities and feedback loops.

Feedback and transparency are key drivers of on-call improvement. Encouraging honest feedback allows on-call team members to question existing policies and structures and suggest possible improvements.

Experimenting with different on-call scenarios is one of the best ways to improve on call. For example, if participation in scrums is taking time away from active on-call duties, try eliminating the need for on-call team members to attend. Or you could ask for help to identify and fix potential alerting issues until on-call is stable. Updating and experimenting on a continuous basis is crucial to adjusting policies to keep pace with changing systems.

Practices such as shadowing, using runbooks, and conducting game days which we discussed in Chapter 8 Training Your Team, all help bring people up to speed and learn on the job.

Enforcing global policies doesn't necessarily help build a culture of learning. Use feedback from teams and be flexible to allow teams to make decisions that work best for them.

Incorporating feedback from developers and Ops engineers is also an integral part of ongoing learning for on-call teams.

## Leverage data to ensure satisfaction

There are two major stakeholders in the on-call process: the customer who relies on your organization for ongoing service and your staff who must carry out the often stressful duties of on-call. Successful on-call aims to achieve high levels of satisfaction for both groups. Leveraging available data is often the most effective way to gauge the satisfaction level of each group.

Automated on-call tools are central to on-call scheduling and contain large amounts of data that can help you discover potential areas of improvement. Most metrics derived from this data can indicate the satisfaction level of both stakeholders.

Reliability objectives such as SLOs play a key role in defining service levels that are important to your customers. Creating alerts tied to these objectives measures your performance in achieving customer satisfaction.

Alerting severity and frequency metrics are also important because they indicate potential problems for both customers and internal staff. Tracking these measures can uncover potential problems and allow you to initiate corrective action.

Maintaining high levels of satisfaction within your on-call/incident responder teams is also key to a successful on-call program. Reviewing on-call scheduling metrics can reveal team members who are taking an inordinate amount of on-call shifts which can lead to burnout. Having this information allows you to review the staffing of your teams to make shifting assignments more equitable.

Metrics that can help you measure performance and satisfaction levels of both customers and staff include:

- Overall responsiveness on alerts
- Mean time to resolve incidents (aka. MTTR)
- Mean time to acknowledge incidents (aka. MTTA)
- Customer impact of incidents (several factors contribute to this. Could be a combination of multiple metrics which exists before and after the incident)
- Alert severity levels
- Alerts per source
- Alert distribution per day of week
- Alerts escalated by team
- Frequency of incidents with certain tag/parameter
- Number of notifications
- On-call duration for each rotation
- On-call frequency

## Conduct blameless postmortems

Failures will always happen in an on-call setting. However it is critical that you learn from every incident to ensure the same error does not reoccur. It is imperative to get to contributing factors of failures so that you can learn from them.

An effective way to investigate incidents and capture lessons learned is by conducting an incident postmortem, also known as a post-incident review. A postmortem is a documented record that describes the incident's impact, actions taken to mitigate or resolve the incident, the contributing factors of the incident, and the corrective actions taken to ensure the incident doesn't happen again. Postmortems present opportunities to uncover vulnerabilities in your system that you can target for improvement.

The natural tendency of people following high impact incidents is to look for someone to blame – a fruitless task that helps no one. On the contrary, "scapegoating" can have the reverse effect as people become afraid to make mistakes, avoid taking risky jobs, and worst case, fail to tell the truth when they are responsible for an error. Incidents will always occur in complex systems so blaming is a futile exercise. An organization that realizes this can take an important step toward conducting blameless postmortems. Many teams, including Atlassian and Google, have adopted the tenants of the blameless postmortem in order to avoid the pitfalls of the blame game.

The people who were involved are the best source of information when it comes to creating the incident postmortem report. It is important to start working on writing the postmortem as soon as possible while the details are still fresh in people's minds. You should devote enough time to fully investigate the issues surrounding the incident. Google estimates that it takes an average of six hours to conduct root-cause analysis, determine remediation, and complete follow-up activities. Google takes the time necessary to initiate corrective measures to operational load returns to a sustainable state after each quarterly review. The time you need will vary depending on the severity of the incident and the extent of the investigation.

Reviewing a completed incident postmortem report is crucial to close out unresolved issues and capture improvement ideas to consider in the future. An effective way to finalize an incident postmortem report is to schedule a monthly meeting with the on-call team members, engineering, customer support, account managers, and any other interested party. These meetings should be used to review current and past incident postmortem reports to share lessons-learned.

In order for postmortems to be effective in allowing you to build a culture of continuous improvement, you should implement a simple, repeatable process in which all affected parties can participate. Your process will depend on your culture and your team. At Atlassian, we've developed a method that works for us and you can read more about it in-depth in our incident handbook.

## Evaluate successes

Analyzing incident failures to uncover contributing factors is important but evaluating successes can also be an effective way to unlock potential areas of improvement. People may often be more open to discussing why something succeeded rather than why it failed. This valuable dialogue can lead to more insights that drive ongoing improvement.

Looking only at failures can deprive you of significant opportunities to improve. As Ryan Kitchens, Senior Site Reliability Engineer at Netflix has said, "We don't define success by the absence of failure. It is the ability to cope with the things changing around us continuously."[29]

For example, a report may indicate a successful trend in lowering the number of alerts. Asking what was done well to achieve the reduction may reveal the cause to be a change in the alerting threshold. This could lead to a complete review of all alerting parameters to develop even more innovative solutions.

Asking the right questions when you are successful can help uncover other best practices and policies that enable you to unleash the full potential of your on-call teams. Questions to consider when evaluating successes:

- What did we do well?
- What can we do better?
- How do we incorporate what we learned into other areas of our on-call process?

## Summary

There are many ways to train your on-call teams so that they perform well. However, lessons learned from real-world experiences – both failures and successes – can be the most fertile resources for continuous improvement. Continuously learning from incident analysis and incorporating those insights into existing processes and procedures can be the best way to ensure that your team is always performing at the highest level possible.

## REFERENCES


1 Accelerate, The Science Behind DevOps, Building and Scaling High Performing Technology Operations, by Nicole Forsgren, PhD, Jez Humble, and Gene, Kim, published by IT Revolution Press , Portland, Oregon, 2018

2 The Phoenix Project, A Novel About IT, DevOps, and Helping Your Business Win, by Gene Kim, Kevin Behr, and George Spafford, published by IT Revolution Press, Portland Oregon, 2013

3 The DevOps Handbook, How to Create World Class Agility, Reliability and Security in Technology Organizations, by Gene Kim, Patrick Debois, John Willis, and Jez Humble, published by IT Revolution Press, Portland Oregon, 2016.

4 The Cost of Downtime, by Andrew Lerner, Garner Blog Network, July 16, 2014

5 Robert Sapolsky discusses physiological effects of stress, by Mark Shwartz, Stanford News, March 7, 2007.

6 The Employee Burnout Crisis: Study Reveals Big Workplace Challenge in 2017, Kronos Inc and Future Workplace, January 9, 2017.

7 Close to 60 Percent of Tech Workers are Burnt Out – Credit Karma Tops the List for Most Employees Suffering from Burnout, by Kyle McCarthy, Blind Workplace Insights, May 29, 2018.

8 Burnout is real – here's how to avoid it, by Tracy Middleton, Atlassian Work Life, June 24, 2019.

9 Extended work availability and its relation with start-of-day mood and cortisol, by Dettmers J, Vahle-Hinz T, Bamberg E, Friedrich N, Keller M., Journal of Occupational Health Psychology, January 2016.

10 What can developers learn from being on call, by Julia Evans, blog post, June 2017

11 Who's on call?, Susan Fowler blog post, September 6, 2016.

12 How we fixed our on call process to avoid engineer burnout, by Brian Scanlan, Inside Intercom blog post,

13 How we do on-call at Monzo, Monzo blog post, September 20, 2018

14 What is 'Site Reliability Engineering'? by Nial Murphy, Google website, accessed 2/5/20

15 #BrandsGetReal: Social media & the evolution of transparency, Sprout Social, 2018

16 Irregular Work Scheduling Consequences, by Lonnie Green, Economic Policy Institute, April 9, 2015

17 Businesses Losing $700 Billion a Year to IT Downtime, Says IHS, by Matthias Machowinski, IHS Markit, January 25, 2016.

18 On-Call and Incident Response: Lessons for Success, the New Relic Way, by Beth Adele Long, New Relic blog post, October 24, 2018

19 Understanding and fighting alert fatigue, Atlassian blog post https://www.atlassian.com/incident-management/on-call/alert-fatigue

20 Over-monitoring and alarm fatigue: For whom do the bells toll? by Shelli Feder, Majorie Funk, Heart & Lung, Nov-Dec 2013

21 Insights into the Problem of Alarm Fatigue with Physiologic Monitor Devices: A Comprehensive Observational Study of Consecutive Intensive Care Unit Patients, by Barbara J. Drew, Patricia Harris, Jessica K. Zègre-Hemsey, Tina Mammone, Daniel Schindler, Rebeca Salas-Boni, Yong Bai, Adelita Tinoco, Quan Ding, Xiao Hu, PLOS One, October 22, 2014

22 Want to Solve Over-Monitoring and Alert Fatigue? Create the Right Incentives! by Kishore Jalleda, Yahoo, Inc., SRECON, August 30, 2017

23 ITIL Incident Escalation, ITIL website, accessed December 2019

24 How to compensate developers and other engineers who are on call and have to respond to emergencies like code fixes or outages, Quora, October 4, 2017

25 Don't Fuck Up the Culture, by Brian Chesky, Medium, April 20, 2014

26 How to Run an Adversarial Game Day, by Terri Haber, New Relic blog post, January 22, 2019

27 The Oncall Simulator, Building an interactive game for teaching incident response! By Franks Schmidt, Monitorama 2018, Portland OR

28 10 Must-read DevOps resources, by Chris Short, Opensource.com, December 6, 2017

29 How did things go right? Learning from incidents, by Ryan Kitchens, Netflix, Presentation at SRECON Americas, March 25, 2019

# About the author

### Serhat Can

Serhat Can is a versatile engineer who has built and operated products as part of Atlassian's Opsgenie team since 2015. As an engineer and DevOps evangelist his main interest is helping teams build better on-call and incident response practices.

He is a frequent speaker at international conferences where his talks focus on DevOps, incident response, on call and the AWS ecosystem.

Serhat organizes many conferences and community events inTurkey where he lives. He's an AWS Community Hero. He helps coordinate DevOpsDays in more than 80 cities around the world, and publishes content on his own blog and personal website.

Stay in touch with Serhat at twitter.com/srhtcn and linkedin.com/in/serhatcan.

ATLASSIAN