Chapter 11 - Being On-Call

Being On-Call

Written by Andrea Spadaccini[56]
Edited by Kavita Guliani

Being on-call is a critical duty that many operations and engineering teams must undertake in order to keep their services reliable and available. However, there are several pitfalls in the organization of on-call rotations and responsibilities that can lead to serious consequences for the services and for the teams if not avoided. This chapter describes the primary tenets of the approach to on-call that Google's Site Reliability Engineers (SREs) have developed over years, and explains how that approach has led to reliable services and sustainable workload over time.

Introduction

Several professions require employees to perform some sort of on-call duty, which entails being available for calls during both working and nonworking hours. In the IT context, on-call activities have historically been performed by dedicated Ops teams tasked with the primary responsibility of keeping the service(s) for which they are responsible in good health.

Many important services in Google, e.g., Search, Ads, and Gmail, have dedicated teams of SREs responsible for the performance and reliability of these services. Thus, SREs are on-call for the services they support. The SRE teams are quite different from purely operational teams in that they place heavy emphasis on the use of engineering to approach problems. These problems, which typically fall in the operational domain, exist at a scale that would be intractable without software engineering solutions.

To enforce this type of problem solving, Google hires people with a diverse background in systems and software engineering into SRE teams. We cap the amount of time SREs spend on purely operational work at 50%; at minimum, 50% of an SRE's time should be allocated to engineering projects that further scale the impact of the team through automation, in addition to improving the service.

Life of an On-Call Engineer

This section describes the typical activities of an on-call engineer and provides some background for the rest of the chapter.

As the guardians of production systems, on-call engineers take care of their assigned operations by managing outages that affect the team and performing and/or vetting production changes.

When on-call, an engineer is available to perform operations on production systems within minutes, according to the paging response times agreed to by the team and the business system owners. Typical values are 5 minutes for user-facing or otherwise highly time-critical services, and 30 minutes for less time-sensitive systems. The company provides the page-receiving device, which is typically a phone. Google has flexible alert delivery systems that can dispatch pages via multiple mechanisms (email, SMS, robot call, app) across multiple devices.

Response times are related to desired service availability, as demonstrated by the following simplistic example: if a user-facing system must obtain 4 nines of availability in a given quarter (99.99%), the allowed quarterly downtime is around 13 minutes (Availability Table). This constraint implies that the reaction time of on-call engineers has to be in the order of minutes (strictly speaking, 13 minutes). For systems with more relaxed SLOs, the reaction time can be on the order of tens of minutes.

As soon as a page is received and acknowledged, the on-call engineer is expected to triage the problem and work toward its resolution, possibly involving other team members and escalating as needed.

Nonpaging production events, such as lower priority alerts or software releases, can also be handled and/or vetted by the on-call engineer during business hours. These activities are less urgent than paging events, which take priority over almost every other task, including project work. For more insight on interrupts and other non-paging events that contribute to operational load, see Dealing with Interrupts.

Many teams have both a primary and a secondary on-call rotation. The distribution of duties between the primary and the secondary varies from team to team. One team might employ the secondary as a fall-through for the pages the primary on-call misses. Another team might specify that the primary on-call handles only pages, while the secondary handles all other non-urgent production activities.

In teams for which a secondary rotation is not strictly required for duty distribution, it is common for two related teams to serve as secondary on-call for each other, with fall-through handling duties. This setup eliminates the need for an exclusive secondary on-call rotation.

There are many ways to organize on-call rotations; for detailed analysis, refer to the "Oncall" chapter of [Lim14].

Balanced On-Call

SRE teams have specific constraints on the quantity and quality of on-call shifts. The quantity of on-call can be calculated by the percent of time spent by engineers on on-call duties. The quality of on-call can be calculated by the number of incidents that occur during an on-call shift.

SRE managers have the responsibility of keeping the on-call workload balanced and sustainable across these two axes.

Balance in Quantity

We strongly believe that the "E" in "SRE" is a defining characteristic of our organization, so we strive to invest at least 50% of SRE time into engineering: of the remainder, no more than 25% can be spent on-call, leaving up to another 25% on other types of operational, nonproject work.

Using the 25% on-call rule, we can derive the minimum number of SREs required to sustain a 24/7 on-call rotation. Assuming that there are always two people on-call (primary and secondary, with different duties), the minimum number of engineers needed for on-call duty from a single-site team is eight: assuming week-long shifts, each engineer is on-call (primary or secondary) for one week every month. For dual-site teams, a reasonable minimum size of each team is six, both to honor the 25% rule and to ensure a substantial and critical mass of engineers for the team.

If a service entails enough work to justify growing a single-site team, we prefer to create a multi-site team. A multi-site team is advantageous for two reasons:

- Night shifts have detrimental effects on people’s health [Dur05], and a multi-site "follow the sun" rotation allows teams to avoid night shifts altogether.
- Limiting the number of engineers in the on-call rotation ensures that engineers do not lose touch with the production systems (see A Treacherous Enemy: Operational Underload).

However, multi-site teams incur communication and coordination overhead. Therefore, the decision to go multi-site or single-site should be based upon the trade-offs each option entails, the importance of the system, and the workload each system generates.

Balance in Quality

For each on-call shift, an engineer should have sufficient time to deal with any incidents and follow-up activities such as writing postmortems [Loo10]. Let’s define an incident as a sequence of events and alerts that are related to the same root cause and would be discussed as part of the same postmortem. We’ve found that on average, dealing with the tasks involved in an on-call incident—root-cause analysis, remediation, and follow-up activities like writing a postmortem and fixing bugs—takes 6 hours. It follows that the maximum number of incidents per day is 2 per 12-hour on-call shift. In order to stay within this upper bound, the distribution of paging events should be very flat over time, with a likely median value of 0: if a given component or issue causes pages every day (median incidents/day > 1), it is likely that something else will break at some point, thus causing more incidents than should be permitted.

If this limit is temporarily exceeded, e.g., for a quarter, corrective measures should be put in place to make sure that the operational load returns to a sustainable state (see Operational Overload and Embedding an SRE to Recover from Operational Overload).

Compensation

Adequate compensation needs to be considered for out-of-hours support. Different organizations handle on-call compensation in different ways; Google offers time-off-in-lieu or straight cash compensation, capped at some proportion of overall salary. The compensation cap represents, in practice, a limit on the amount of on-call work that will be taken on by any individual. This compensation structure ensures incentivization to be involved in on-call duties as required by the team, but also promotes a balanced on-call work distribution and limits potential drawbacks of excessive on-call work, such as burnout or inadequate time for project work.

Feeling Safe

As mentioned earlier, SRE teams support Google’s most critical systems. Being an SRE on-call typically means assuming responsibility for user-facing, revenue-critical systems or for the infrastructure required to keep these systems up and running. SRE methodology for thinking about and tackling problems is vital for the appropriate operation of services.

Modern research identifies two distinct ways of thinking that an individual may, consciously or subconsciously, choose when faced with challenges [Kah11]:

- Intuitive, automatic, and rapid action
- Rational, focused, and deliberate cognitive functions

When one is dealing with the outages related to complex systems, the second of these options is more likely to produce better results and lead to well-planned incident handling.

To make sure that the engineers are in the appropriate frame of mind to leverage the latter mindset, it’s important to reduce the stress related to being on-call. The importance and the impact of the services and the consequences of potential outages can create significant pressure on the on-call engineers, damaging the well-being of individual team members and possibly prompting SREs to make incorrect choices that can endanger the availability of the service. Stress hormones like cortisol and corticotropin-releasing hormone (CRH) are known to cause behavioral consequences—including fear—that can impair cognitive functions and cause suboptimal decision making [Chr09].

Under the influence of these stress hormones, the more deliberate cognitive approach is typically subsumed by unreflective and unconsidered (but immediate) action, leading to potential abuse of heuristics. Heuristics are very tempting behaviors when one is on-call. For example, when the same alert pages for the fourth time in the week, and the previous three pages were initiated by an external infrastructure system, it is extremely tempting to exercise confirmation bias by automatically associating this fourth occurrence of the problem with the previous cause.

While intuition and quick reactions can seem like desirable traits in the middle of incident management, they have downsides. Intuition can be wrong and is often less supportable by obvious data. Thus, following intuition can lead an engineer to waste time pursuing a line of reasoning that is incorrect from the start. Quick reactions are deep-rooted in habit, and habitual responses are unconsidered, which means they can be disastrous. The ideal methodology in incident management strikes the perfect balance of taking steps at the desired pace when enough data is available to make a reasonable decision while simultaneously critically examining your assumptions.

It's important that on-call SREs understand that they can rely on several resources that make the experience of being on-call less daunting than it may seem. The most important on-call resources are:

- Clear escalation paths
- Well-defined incident-management procedures
- A blameless postmortem culture ([Loo10], [All12])

The developer teams of SRE-supported systems usually participate in a 24/7 on-call rotation, and it is always possible to escalate to these partner teams when necessary. The appropriate escalation of outages is generally a principled way to react to serious outages with significant unknown dimensions.

When one is handling incidents, if the issue is complex enough to involve multiple teams or if, after some investigation, it is not yet possible to estimate an upper bound for the incident's time span, it can be useful to adopt a formal incident-management protocol. Google SRE uses the protocol described in Managing Incidents, which offers an easy-to-follow and well-defined set of steps that aid an on-call engineer to rationally pursue a satisfactory incident resolution with all the required help. This protocol is internally supported by a web-based tool that automates most of the incident management actions, such as handing off roles and recording and communicating status updates. This tool allows incident managers to focus on dealing with the incident, rather than spending time and cognitive effort on mundane actions such as formatting emails or updating several communication channels at once.

Finally, when an incident occurs, it's important to evaluate what went wrong, recognize what went well, and take action to prevent the same errors from recurring in the future. SRE teams must write postmortems after significant incidents and detail a full timeline of the events that occurred. By focusing on events rather than the people, these postmortems provide significant value. Rather than placing blame on individuals, they derive value from the systematic analysis of production incidents. Mistakes happen, and software should make sure that we make as few mistakes as possible. Recognizing automation opportunities is one of the best ways to prevent human errors [Loo10].

Avoiding Inappropriate Operational Load

As mentioned in Balanced On-Call, SREs spend at most 50% of their time on operational work. What happens if operational activities exceed this limit?

Operational Overload

The SRE team and leadership are responsible for including concrete objectives in quarterly work planning in order to make sure that the workload returns to sustainable levels. Temporarily loaning an experienced SRE to an overloaded team, discussed in Embedding an SRE to Recover from Operational Overload, can provide enough breathing room so that the team can make headway in addressing issues.

Ideally, symptoms of operational overload should be measurable, so that the goals can be quantified (e.g., number of daily tickets < 5, paging events per shift < 2).

Misconfigured monitoring is a common cause of operational overload. Paging alerts should be aligned with the symptoms that threaten a service's SLOs. All paging alerts should also be actionable. Low-priority alerts that bother the on-call engineer every hour (or more frequently) disrupt productivity, and the fatigue such alerts induce can also cause serious alerts to be treated with less attention than necessary.See [Dealing with Interrupts](#) for further discussion.

It is also important to control the number of alerts that the on-call engineers receive for a single incident. Sometimes a single abnormal condition can generate several alerts, so it's important to regulate the alert fan-out by ensuring that related alerts are grouped together by the monitoring or alerting system. If, for any reason, duplicate or uninformative alerts are generated during an incident, silencing those alerts can provide the necessary quiet for the on-call engineer to focus on the incident itself. Noisy alerts that systematically generate more than one alert per incident should be tweaked to approach a 1:1 alert/incident ratio. Doing so allows the on-call engineer to focus on the incident instead of triaging duplicate alerts.

Sometimes the changes that cause operational overload are not under the control of the SRE teams. For example, the application developers might introduce changes that cause the system to be more noisy, less reliable, or both. In this case, it is appropriate to work together with the application developers to set common goals to improve the system.

In extreme cases, SRE teams may have the option to "give back the pager"—SRE can ask the developer team to be exclusively on-call for the system until it meets the standards of the SRE team in question. Giving back the pager doesn't happen very frequently, because it's almost always possible to work with the developer team to reduce the operational load and make a given system more reliable. In some cases, though, complex or architectural changes spanning multiple quarters might be required to make a system sustainable from an operational point of view. In such cases, the SRE team should not be subject to an excessive operational load. Instead, it is appropriate to negotiate the reorganization of on-call responsibilities with the development team, possibly routing some or all paging alerts to the developer on-call. Such a solution is typically a temporary measure, during which time the SRE and developer teams work together to get the service in shape to be on-boarded by the SRE team again.

The possibility of renegotiating on-call responsibilities between SRE and product development teams attests to the balance of powers between the teams.[57] This working relationship also exemplifies how the healthy tension between these two teams and the values that they represent—reliability versus feature velocity—is typically resolved by greatly benefiting the service and, by extension, the company as a whole.

A Treacherous Enemy: Operational Underload

Being on-call for a quiet system is blissful, but what happens if the system is too quiet or when SREs are not on-call often enough? An operational underload is undesirable for an SRE team. Being out of touch with production for long periods of time can lead to confidence issues, both in terms of overconfidence and underconfidence, while knowledge gaps are discovered only when an incident occurs.

To counteract this eventuality, SRE teams should be sized to allow every engineer to be on-call at least once or twice a quarter, thus ensuring that each team member is sufficiently exposed to production. "Wheel of Misfortune" exercises (discussed in [Accelerating SREs to On-Call and Beyond](#)) are also useful team activities that can help to hone and improve troubleshooting skills and knowledge of the service. Google also has a company-wide annual disaster recovery event called DiRT (Disaster Recovery Training) that combines theoretical and practical drills to perform multiday testing of infrastructure systems and individual services; see [Kri12].

Conclusions

The approach to on-call described in this chapter serves as a guideline for all SRE teams in Google and is key to fostering a sustainable and manageable work environment. Google's approach to on-call has enabled us to use engineering work as the primary means to scale production responsibilities and maintain high reliability and availability despite the increasing complexity and number of systems and services for which SREs are responsible.

While this approach might not be immediately applicable to all contexts in which engineers need to be on-call for IT services, we believe it represents a solid model that organizations can adopt in scaling to meet a growing volume of on-call work.

[56]An earlier version of this chapter appeared as an article in *;login:* (October 2015, vol. 40, no. 5).

[57]For more discussion on the natural tension between SRE and product development teams, see Introduction.

previous

Chapter 10 - Practical Alerting

next

Chapter 12 - Effective Troubleshooting