Being an On-Call Engineer

A Google SRE Perspective

ANDREA SPADACCINI AND KAVITA GULIANI

Andrea Spadaccini works in Dublin as a Site Reliability Manager for Google, which he joined in 2012 as an SRE working on the systems that distill, store, and serve all the metrics about Google's Ads platforms. Prior to that, he worked on Linux-based PBX products, hacked on open source CPU simulators, and co-founded a nonprofit for students to get work experience while pursuing their studies. He earned a PhD in computer engineering from the University of Catania, Italy, where he focused mostly on biometric recognition. .

Kavita Guliani is a Technical Writer for Technical Infrastructure and Site Reliability Engineering in Google Mountain View. Before working at Google, Kavita worked for companies like Symantec, Cisco, and Lam Research Corporation. She holds degree in English from Delhi University and studied technical writing at San Jose State University. kguliani@google.com

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. We provide the primary tenets of the approach to on-call that Google's Site Reliability Engineers have developed over years, and explain how that approach has led to reliable services and sustainable workload over time.

Several professions require employees to perform some sort of on-call duty, which entails being available for calls during both working and non-working 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 Site Reliability Engineers (SREs) [1] responsible for the performance and reliability of these services. As such, 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.

We present an informed view of how Google SRE teams organize the on-call aspect of their jobs, and how Google's strong focus on engineering determines numerous aspects of this organization.

We do not describe all the possible ways of organizing on-call rotations in detail. For detailed analysis, refer to the "Oncall" chapter of *The Practice of Cloud System Administration* [2].

Life of an On-Call Engineer

As the guardian of production systems, the on-call engineer takes care of his or her 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 Service Level Objectives (SLOs) agreed to by the team and the business system owners. Typical SLO values are five 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 dispatch pages via multiple mechanisms (email, SMS, robot call, app) across multiple devices.

This page-to-work-towards-resolution SLO is distinct from the service SLOs themselves (e.g., user-facing latency, processing delay, and so on). There is a relationship between the two types of SLOs: the service SLOs imply upper bounds for the page-to-work-towards-resolution SLO. For 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. This constraint implies that the reaction time of on-call engineers has to be on the order of 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 towards its resolution, possibly involving other team members and escalating as needed.

Non-paging 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.

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 and ranges from the secondary acting as a fall-through for the pages missed by the primary on-call to an arrangement in which the primary on-call handles only pages and 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.

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 percentage 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 are responsible for keeping the on-call workload balanced and sustainable across these two axes.

Balance in Quantity

SREs can spend no more than 25% of their time on-call, and another 25% of their time on other types of operational, non-project work. 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 in engineering.

Using the 25% 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 implies enough work to justify growing a single-site team, we can create a multi-site team. A multi-site team can be advantageous for two reasons:

- Night shifts have detrimental effects on people's health [3], and multi-site 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: Operation Underload," below).

However, multi-site teams incur communication and coordination overhead. Therefore, the decision to go multi-site or single-site should be based on the tradeoffs 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 incidents and follow-up activities such as writing postmortems [4]. Assuming that on-call incidents, on average, require six hours of work between investigation, root cause analysis, remediation, and follow-up activities such as writing a postmortem, it follows that the maximum number of incidents per day is two. 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 "Avoiding Operational Overload," below).

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 choose, consciously or subconsciously, when faced with challenges:

- Intuitive, automatic, and rapid action
- Rational, focused, and deliberate cognitive functions [5]

When 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 CRH are known to cause behavioral consequences—including fear—that can impair cognitive functions and cause suboptimal decision-making [6].

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 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 between taking steps at the desired pace when enough data is available to make a reasonable decision and 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 [4]

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 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" [7], which offers an easy to follow and well-defined set of steps that aid an on-call engineer in rationally pursuing 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, value is derived 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 [4].

Avoiding Inappropriate Operational Load

Operational Overload

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

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

Monitoring misconfiguration 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.

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, as 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 onboarded by the SRE team again.

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

A Treacherous Enemy: Operation 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 operation 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 once or twice a month, thus ensuring that each team member is sufficiently exposed to production.

Some teams also run so-called "Wheel of Misfortune" exercises, in which theoretical (or practical) incident scenarios are presented to the team by a dungeon master, much in the style of traditional role-playing games. This exercise is also a useful team activity 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 multi-day testing of infrastructure systems and individual services.

Onboarding New Systems

It is common for SRE teams to become responsible for new systems, a process that typically culminates in handing off pager responsibilities, also called onboarding.

The SRE team needs to engage with the new system well before the onboarding process starts. Ideally, the SREs are involved from the early design phase of the new system, as their knowledge and experience with the production infrastructure can offer an important perspective on the architecture of the new systems. Direct involvement by SREs during the development phase might be necessary as the system approaches its launch, in preparation for a Production Readiness Review (PRR) or Launch Review.

After the new system launches, the application developers may remain on-call for the system until the ownership is transitioned to SRE. A system must meet specific requirements with regards to reliability, Service Level Objectives (SLOs), alerting, and the on-call load before it is onboarded by SRE. The on-call training can begin towards the end of the onboarding process. Generally, the application developers train SREs on the internals of the new systems, explaining the most likely or common failure modes and how to react to these failures. To demonstrate debugging techniques, developers may fake troubleshooting scenarios and demonstrate their resolution to SREs.

All alerts are expected to have corresponding documentation that enables the on-call engineer to take appropriate actions when paged. Upon service handoff, documentation ownership is transitioned to SREs, who are expected to keep the docs up-to-date in collaboration with the application developers.

Conclusion

The approach to on-call we described serves as a guideline for all SRE teams in Google and is key to fostering a sustainable and safe 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.

References

[1] http://www.site-reliability-engineering.info/.

[2] Thomas A. Limoncelli, Strata R. Chalup, Christina J. Hogan, "Oncall," in *The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems*, vol. 2, Pearson Education, 2014.

[3] Jeffrey S. Durmer and David F. Dinges, "Neurocognitive Consequences of Sleep Deprivation," in *Seminars in Neurology,* vol. 25, no. 1, 2005.

[4] Jake Loomis, "How to Make Failure Beautiful: The Art and Science of Postmortems," in *Web Operations: Keeping the Data on Time*, O'Reilly Media, 2010

[5] Daniel Kahneman, *Thinking, Fast and Slow,* Farrar, Straus and Giroux, 2011.

[6] George P. Chrousous, "Stress and Disorders of the Stress System," *Nature Reviews Endocrinology,* vol. 5, July 2009, doi: 10.1038/nrendo.2009.106.

[7] Andrew Stribblehill, Kavita Guliani, "Managing Incidents," *;login:,* vol. 40, no. 2, April 2015: https://www.usenix.org/publications/login/apr15/stribblehill.