splunk> + VictorOps

Resilience First:

SRE and the Four Golden Signals of Monitoring

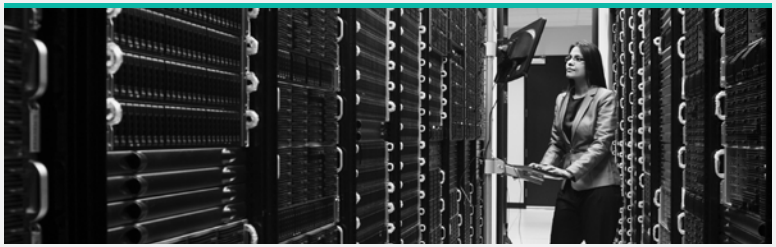
TABLE OF CONTENTS

What is SRE?

SRE (site reliability engineering) is a discipline used by software engineering and IT teams to proactively build and maintain more reliable services. SRE is a functional way to apply software development solutions to IT operations problems. From monitoring to software delivery to incident response – site reliability engineers are focused on building and monitoring anything in production that improves service resiliency without harming development speed.

Site reliability engineering is often used as a highly-integrated method for tightening the relationship between developers and IT teams. An SRE's biggest role is to improve the overall resilience of a system and provide visibility to the health and performance of services across all applications and infrastructure. Site reliability engineers can actively write code to improve service resilience and flexibility. Then, they can help spread information across DevOps and business teams – encouraging a blameless culture focused on workflow visibility and collaboration.

The Traditional Approach to Service Management

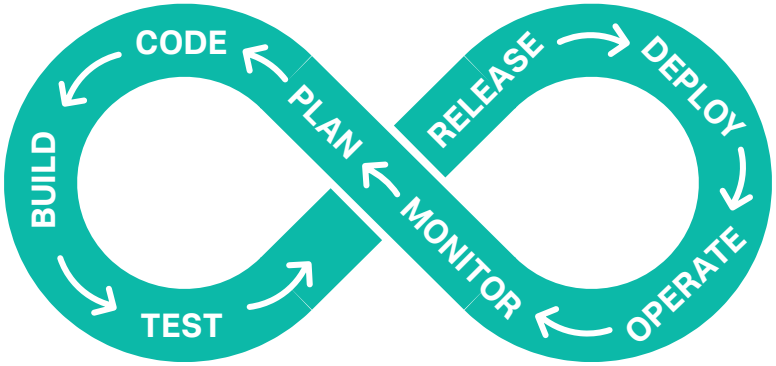


ITSM, or IT service management, has been around since the beginning of computers. System administrators (sysadmins) would handle everything from assembling software components, deploying them and responding to incidents. Then, with the introduction of personal computers, IT professionals needed to define universal principles for reliably handling applications and infrastructure.

The growing adoption of technology gave way to the IT Infrastructure Library (ITIL) – a set of defined rules for all of IT operations. For a while, defined rules worked well – software developers wrote the code and gave it to sysadmins who were in charge of configuring and deploying the services. However, this proverbial fence led to a natural division of labor between software developers and sysadmins.

Then, with the birth of the internet and highly complex, integrated systems, Agile software development practices and CI/CD became a necessity. In order to keep up with the faster delivery of always-on services, IT service management practices also needed to change.

Enter DevOps



DevOps was adopted in response to the shift in development and release practices. DevOps is a method for tightening feedback loops and improving collaboration between software developers and IT operations. A DevOps methodology gets IT teams involved earlier in the software delivery lifecycle while also increasing developer accountability for services in production. Application and infrastructure testing is automated and integrated throughout more of the development lifecycle and developers take on-call responsibilities for the services they build.

Why DevOps Matters Free Guide

With the faster delivery of more complex applications and cloud infrastructure, teams needed a way to proactively address reliability concerns – leading to the creation of the modern practice of SRE.

The Origin of SRE

The discipline of site reliability engineering first popped up at Google. They needed to make a shift towards an IT-centric organization, aligning everyone across the business – from engineering to sales. SRE became a way to move forward with this vision. Google's then VP of Engineering, Ben Treynor, defined SRE as:

> "Fundamentally, it's what happens when you ask a software engineer to design an operations function."
>
> -Google Interview with Ben Treynor

Google started to treat issues that were normally solved manually as software problems – formalizing an SRE team to apply software development expertise to traditional IT operations problems. With developers focused solely on making operations better, the team can build resilience into their services without harming development speed – automating numerous manual tasks and tests, increasing visibility into system health and improving collaboration across all of IT and engineering.

The Core Components of SRE

> "40-90% of the total costs of a system are incurred after birth."
> -Google's SRE Book

While most DevOps and IT professionals are constantly focused on improving the development process, a large number of teams don't focus on their systems in production. But, the vast majority of application and infrastructure costs are incurred after deployment. It stands to reason that development teams need to spend more time supporting current services. In order to reallocate their time without impeding velocity, SRE teams are forming – dedicating developers to the continuous improvement of the resilience of their production systems.

The core responsibilities of SRE teams normally fall into five categories:

1. Availability:

Setting service-level objectives, agreements and indicators (SLOs, SLAs and SLIs) for the underlying service. SLIs are the actual unit of measurement defining the service level that customers can expect of the system. SLIs form the basis of SLOs which are the desired outputs of the system (e.g. 99.99% availability, etc.). SLAs are based on SLOs and given to customers to communicate the expected reliability of the service they'll be using, and the way the team will react if those numbers aren't met.

- What are the expectations of internal teams, customers and other external stakeholders?
- What's the overall importance of service to most organizations?
- Is 99.999% availability really necessary?

These types of questions listed above will help facilitate the speed at which businesses can reliably release new features and services – dictating the way SRE teams initially set SLOs, SLAs and SLIs.

Over time, as SRE teams spend more time working in production environments, engineering organizations begin to see more resilient architecture with further failover options and faster rollback capabilities. These companies can then set higher expectations for customers and stakeholders, leading to impressive SLOs, SLAs and SLIs that drive greater business value. While the greater development and IT teams are in charge of maintaining a consistent release pipeline, SRE teams are tasked with maintaining the overall availability of those services once they're in production.

2. Performance:

As teams gain maturity in SRE and availability becomes less erratic, they can start to focus on improving service performance metrics like latency, page load speed and ETL.

- Which services or nodes are frequently failing?
- Are customers consistently experiencing page load or latency lag?

While overall availability may not be impacted by performance errors, customers who frequently encounter performance issues will experience fatigue and may be likely to stop using the service.

SRE teams should not only help application support and development teams fix bugs, but they should help proactively identify performance issues across the system. Small performance issues can build up over time and become larger customer-facing incidents. As overall service reliability improves, teams will open up more time to identify small performance issues and fix them.

3. Monitoring:

In order to identify performance errors and maintain service availability, SRE teams need to see what's going on in their systems. Naturally, the SRE team is assigned the great task of implementing monitoring solutions. Because of the way disparate services measure performance and uptime, deciding what to monitor and how to do so effectively is one of the hardest parts of being a site reliability engineer. At the end of the day, SREs need to think of monitoring as a way to surface a holistic view of a system's health. Anyone from any department in engineering or IT should be able to look at a single source to determine the overall performance and availability of the services they support.

The need for cross-service, cross-team visibility led to the creation of SRE's golden signals. The golden signals serve as a foundation for actionable monitoring and alerting for DevOps and IT teams. In a few pages, we'll go over SRE's four golden signals of monitoring and show why they're such a powerful foundation for service reliability.

4. Incident Response

Many SRE teams won't handle real-time incident response or take on-call responsibilities themselves. But, it's generally good to get the SRE team's input into incident response due to their collaborative efforts across all of development and operations – as well as their deep involvement in staging and production environments. The SRE team is in the best position of any in the organization for determining who should be on-call and when – as well as the people and teams who should be responsible for individual services.

SRE teams can ensure an efficient system for incident response. Through constant improvement of visibility and workflow transparency, SRE teams are able to surface the information on-call teams need to quickly fix problems. Site reliability engineers shouldn't only think of reliability in technical terms but also in human terms. Tightening the collaboration between people, processes and technology will lead to a proactive system of incident response and remediation.

5. Preparation

The continuous improvement of monitoring, incident response, and the optimization of service availability and performance will inherently lead to more resilient systems. At the end of the day, SRE teams build the foundation for a more prepared engineering and IT team. With the monitoring resources provided by the SRE team, the development and IT team can deploy new services quickly and respond to incidents in seconds.

A prepared team knows the health of their services and how to respond when there's a problem. When site reliability engineers are integrated into engineering and IT, developers are exposed to more of their production environment, and IT operations are involved earlier in the software development lifecycle. SRE teams serve the organization with the weapons and the transparency they need to combat reliability concerns. A reactive SRE team simply responds to issues and fixes them. But, a proactive SRE team puts the resilience of the system directly in the hands of individual team members.

Effective implementation of the core components of SRE requires visibility and transparency across all services and applications within a system. But, measuring the performance and availability of disparate services on a single scale can be convoluted. So, Google's SRE team developed the four golden signals as a way to consistently track service health across all applications and infrastructure.

Reviewing the Four Golden Signals of SRE

SRE's golden signals define what it means for the system to be "healthy." Establish benchmarks for each metric showing when the system is healthy – ensuring positive customer experiences and uptime. While a team could always monitor more metrics or logs across the system, the four golden signals are the basic, essential building blocks for any effective monitoring strategy.

Latency

(time taken to serve a request):

Define a benchmark for "good" latency rates. Then, monitor the latency of successful requests against failed requests to keep track of health. Tracking latency across the entire system can help identify which services are not performing well and allows teams to detect incidents faster. The latency of errors can help improve the speed at which teams identify an incident – meaning they can dive into incident response faster.

Traffic

(the stress from demand on the system):

How much stress is the system taking at a given time from users or transactions running through the service? Depending on the business, what you define as traffic could be vastly different from another organization. Is traffic measured as the number of people coming to the site or as the number of requests happening at a given time? By monitoring real-user interactions and traffic in the application or service, SRE teams can see exactly how customers experience the product while also seeing how the system holds up to changes in demand.

Errors

(rate of requests that are failing):

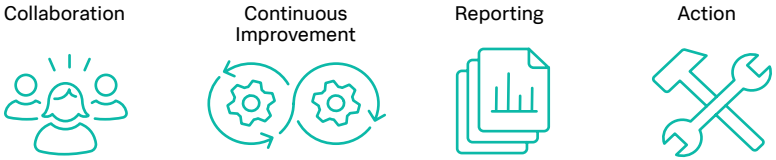
SRE teams need to monitor the rate of errors happening across the entire system but also at the individual service level. Whether those errors are based on manually defined logic or they're explicit errors such as failed HTTP requests, SRE teams need to monitor them. It's also important to define which errors are critical and which ones are less dangerous. This can help teams identify the true health of a service in the eyes of a customer and take rapid action to fix frequent errors.

Saturation

(the overall capacity of the service):

The saturation is a high-level overview of the utilization of the system. How much more capacity does the service have? When is the service maxed out? Because most systems begin to degrade before utilization hits 100%, SRE teams also need to determine a benchmark for a "healthy" percentage of utilization. What level of saturation ensures service performance and availability for customers?

The Foundation of SRE at VictorOps



> "SRE shines a light on the unknowns in a running system, it eliminates fear associated with the unknown. SRE brings awareness to the maintainers of a system and builds confidence toward handling unknown issues."
>
> - Jonathan Schwietert, SRE Team Lead

Nearly three years ago, as the VictorOps engineering team started to grow, Jonathan Schwietert began pitching the idea of building an SRE process. The team needed an avenue to channel the hunger for reliability and discuss ways to improve the resilience of their services. So, the engineering team assessed their desired goals from site reliability engineering and how it would fit into the business.

The Goals of Implementing SRE at VictorOps

Vision:

Provide a unified vision for product reliability at VictorOps

Culture:

Coach engineering to embrace and own reliability in their respective areas

Process:

Provide a first-class development workflow to address reliability

Tools:

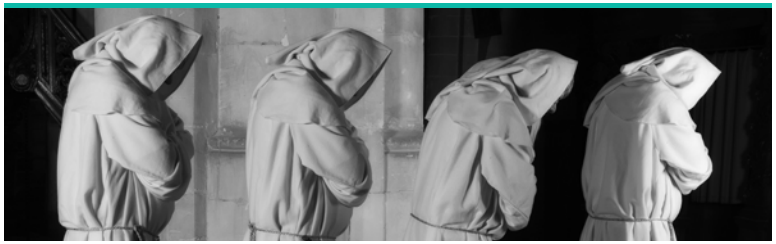
Direct the team to powerful toolsets for instrumenting the system

System:

Maintain and improve reliability in customer experience to achieve a balance between high reliability and deployment speed

For VictorOps, it was important that SRE be tightly integrated with development and operations workflows – increasing the reliability of services without hindering development speed. After looking at the long-term goals for SRE at VictorOps, the team decided to institute the SRE council – a group of engineers from cross-functional engineering teams dedicated to site reliability engineering.

The SRE Council



The SRE council met weekly to discuss any incidents or general concerns from the prior week. Council members would look at the frequency of alerts coming into the system and produce action items to help reduce alert fatigue while ensuring monitoring and incident response processes were put in place. Accordingly, the council broke down their key objectives in order to track their effectiveness over time.

Council

- Bring concerns from the teams
- Vet concerns in order to build SRE backlog
- Present before and after improvements once a concern is addressed
- Provide input to SRE roadmap

Day-to-Day

- Encourage impromptu reliability conversations
- Team point of contact for SRE stories

Agile Ceremonies

- Help break down high-level work into detailed story-level representations
- Act as a representative for SRE stories during backlog refinement and sprint planning
- Present before and after SRE improvements during quarterly sprint planning

With more meetings came more discovery – making the SRE council more productive over time. The members of the council then take their learnings back to their respective development and IT teams – helping spread systemic knowledge and build deeper reliability into the product.

Implementing the Four Golden Signals at VictorOps

Fast forward to today. VictorOps organized an SRE team and process that works. And now, VictorOps is implementing SRE's four golden signals, well, three signals, across the entire system. By starting with the tracking of latency, traffic and error rate across all of our services, we're able to add visibility to system health and provide more context to all of engineering.

The ultimate goal is to track all four golden signals. But, in order to track saturation, you'll need to monitor the other three signals first. Latency, error rate and traffic can help define what it means for a system to be full. The goal of implementing the golden signals is to consistently track health across all services, allowing businesses to scale products and operations, helping on-call responders fix problems for code they didn't write.

What is SRE to VictorOps?

SRE is one of the most important disciplines of our entire business. So, when diving into SRE operations, we brought in site reliability engineers from Netflix, Twitter and Bluprint to discuss how they approach SRE and how their principles can be applied at VictorOps. The SRE council is made up of developers who are writing the next features – allowing us to think concurrently about reliability while building new features. With those teachings and our own experiences over the past three years, we've grown a lot in our SRE maturity.

So, we're including three interviews with members of our SRE council to show how they think about SRE and why it's important for business:

Jonathan Schwietert

(Platform Engineer & SRE Team Lead)

Q: What do you like about the SRE council?
A: It prevents throwing reliability over the wall and protects our DevOps mentality. The people on the SRE council are developers who'll be writing the next feature, so it's important that we're thinking concurrently about reliability while building features.

Q: So what does SRE mean to you, personally?
A: SRE is a way to improve cross-functional collaboration and visibility to help add reliability into everything we build. SRE provides an avenue to direct the hunger that we have for reliability.

Q: What is the ideal SRE team structure to you?
A: There's no ideal structure. Implementing an SRE structure is dependent on a large number of factors including the size of the organization, the product, the people, and the culture of the organization. SRE will look vastly different if you have a young, fairly unstable product vs. that of a mature, more stable product.

If your product is fairly unreliable, you'll have a more retroactive SRE structure, whereas a more reliable product will be able to have a more proactive SRE approach. There's always a ratio to balance between proactive and retroactive SRE, dependent on where you're at in your product's development and life cycle. Because of this, there simply isn't a one-size-fits-all approach to SRE, absolutely not.

Q: What's your favorite part of SRE?
A: SRE shines a light on the unknowns in a running system. It eliminates fear associated with the unknown. SRE brings awareness to the maintainers of a system and builds confidence toward handling unknown issues.

Q: How do you measure SRE efforts and effectiveness?
A: SRE is a tough thing to measure. But, I'd measure it as the stability of a system over time, the frequency of system incidents and severity of incidents. Also, if you think of retroactive SRE as the bottom of a scale, and fully proactive SRE as the top of a scale, where you fall along that scale is a good measurement of SRE's effectiveness. You can also measure SRE based off lower-level monitoring metrics and SLOs.

Q: Do you think chaos engineering and game days are a necessary part of SRE?
A: Yes–it's the proactive side of SRE. It's important to get exposure to multiple parts of the system before a real incident happens. Everyone on the team is more prepared for an issue this way.

Q: Any additional comments?
A: I just want to reiterate that what we've done for SRE at VictorOps is what fits our DevOps culture best. This system works for us, but that doesn't necessarily mean it will work for everyone else. And, of course, we're continuously improving and adapting our SRE practices as we learn and grow.

DeAndre Carroll

(Software Engineer - Middle Tier Team)

Q: How long have you worked in SRE? Is VictorOps your first time?

A: I've worked in SRE before VictorOps, but not in a formalized setting. Part of my previous responsibilities involved hack-testing, trying to figure out how to exploit the system in order to use it for how it was designed. I was working to stop automation and exploitation from taking advantage of our platform. So, I spent some time looking at the site outside of the rules to check for robustness.

Q: How does your role specifically influence reliability in the system?

A: Well, middle tier is the first point of contact for a lot of external systems, we're the gateway to a lot of systems–APIs, integrations, etc. Middle tier needs to build reliability in an inherently chaotic part of the system. Any server side change that modifies the user experience and comes through the web client or mobile client goes through us. We're influencing reliability on both the server-side and the client-side. Middle tier is always working to transfer data reliably, implement proper server-side checks, and optimize the client-side for good UI and experience.

Q: If you could, what would you like to do or change about SRE at VictorOps?

A: Getting more buy-in would be great. That's why we're adding rotations to our SRE Council in order to give more people exposure to SRE. If the SRE Council becomes static for too long, it can become a silo – which is not conducive to our DevOps nature.

Q: In addition to reliability, do you believe SRE improves collaboration and development speed?

A: Collaboration? Yes. Development speed? Maybe. Over time I believe that SRE will improve development speed, but it's a highly front-loaded process. It takes time to set up monitoring tools, alerts and processes to make SRE more proactive. But once you're in a good place, it allows for more time to be spent on development.

Q: What's your favorite part of SRE?

A: My favorite part is being able to share information across all our teams and improve visibility across the entire system.

Q: Do you think chaos engineering and game days are a necessary part of SRE?

A: Yes. Having formalized chaos engineering processes lets teams break away from everything else they're doing and focus in on making the system robust. You can get real results in real time.

Q: Any favorite tools for SRE?

A: Jmeter. Jmeter is a great automated testing and load testing tool.

Q: So, what does the term SRE mean to you?

A: SRE means challenging ourselves to make the system as reliable as we can. SRE is about getting out of your comfort zone, changing your thinking, and approaching problems from a different angle to make the system more reliable and robust. By finding the strengths and weaknesses of your system, SRE helps you make it more reliable for customers and easier to manage for the team.

Andrew Fager
(Data Engineering Team Lead)

Q: So, what's your role on the SRE Council?
A: I represent the data team. I work on building persistence, reliability, and resilience within our data engineering pipelines.

Q: How do you think your role on the SRE Council specifically influences reliability in the overall system?
A: Well, customers rely on our data to remediate incidents. My team is different from other teams in the sense that we need to ensure that our data is in a reliable and persistent place so that our customers can analyze it after the fact. We can work to find ways to break the data pipeline, but ensure that the data is retained. This way, we can build more robust data pipelines for ourselves and our customers.

Q: What do you like about the SRE Council?
A: Different perspectives. Everyone on the Council has a different expertise and way of approaching issues. It's interesting to learn how our whole system is interconnected, how we can improve cross-functional communication, and the way that the reliability of one team can affect another.

Q: What is the ideal SRE team structure to you?
A: Blue team, red team type game days are great. Your team will learn a lot through simulated chaos on the system. Malicious actors on one team can take things down while the other team responds to the issue in a realistic on-call scenario.

Q: How do you measure SRE effectiveness?
A: Well, with the data engineering team, there are usually more lagging metrics and less actionable ways to measure SRE's effectiveness. We can essentially measure the effectiveness of SRE based on the speed and consistency of data in ETL. If ETL lags, how can we bring it up to speed quickly? How can we build a more resilient ETL pipeline?

Q: What's your favorite part of SRE?
A: I love the collaboration it brings. SRE has great data-driven aspects to it. You need visibility into the system and its performance metrics in order to allow people to look at the data and take action on it. Also, game days are a fun part of SRE that can give cross-functional knowledge to team members so incident resolution becomes easier.

Q: So what does the term SRE mean to you?
A: Whole system reliability. A lot of it is about working to mitigate and prepare for unknown unknowns. There's no way you can know everything that could happen on your site, so it's important to be able to react quickly in a non-fatal way.

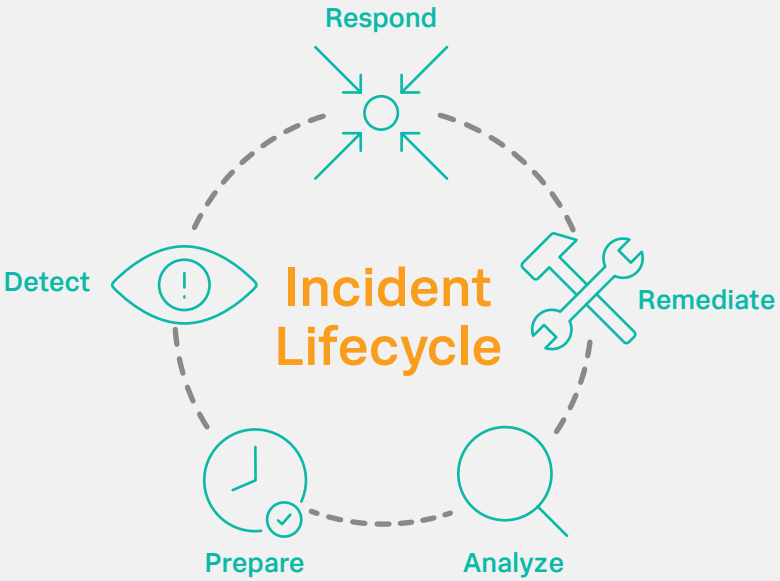
Taking Action with Your Monitoring

A well-adjusted monitoring model gives you a consistent view of how your architecture is behaving. However, visibility into performance and availability is only the first step for site reliability engineering. The monitoring data needs to provide context to engineering and IT teams, allowing them to quickly see what's wrong and loop in the proper teams for incident remediation. A proactive SRE team will set up incident response plans and a collaborative alerting strategy to get the right information to the right people at the right time.

SRE's Four Golden Signals in the Incident Management Lifecycle

The four golden signals serve as an excellent jumping-off point for actionable monitoring. Tracking the latency, traffic, errors and saturation for all services in near real-time will help all teams identify issues faster. The golden signals also give teams a single pane of glass view into the health of all services – whether they're maintained by that specific team or not. Instead of disparate monitoring across every feature or service, you can roll all monitoring metrics and logs into a single location.

Effective monitoring will not only lead to improved incident detection but it will improve the entire incident lifecycle over time.



Using SRE to Facilitate a DevOps Mindset

There isn't one specific method for taking on site reliability engineering. But, whatever you do, SRE should improve visibility and collaboration among people, processes and technology. By dedicating software engineers to application and infrastructure concerns, you're proactively fixing problems. Moreover, with SREs spending so much time in production, they can pass knowledge to the rest of the development team.

Site reliability engineers expose themselves to many aspects of the system, inherently improving the collaboration between developers and IT operations teams. Facilitating a DevOps mindset through SRE leads to breakthroughs in your team's productivity and your system's resilience. When an incident occurs, instead of passing blame between development and IT, SRE opens transparent discussions about how they can improve. SREs are the gatekeepers for efficient, reliable software development practices that don't force all production responsibilities to IT teams.

Implementing SRE and the four golden signals of monitoring will improve cross-functional visibility and collaboration, bringing IT operations and developers together. Join the teams that are already using SRE's four golden signals to promote a positive engineering culture and drive better customer experiences.

Make the Most of SRE and the Golden Signals

Enter VictorOps

Centralize on-call rotations, alert automation and collaborative incident response in one single solution. SRE teams can track deployments and system changes in real-time across all of engineering while consistently monitoring and alerting on SLIs across the entire ecosystem. Add visibility into developer and IT operations workflows, improve the way SREs manage monitoring and alerting, and ultimately build more reliable systems with VictorOps.

Ready to start using a comprehensive alerting and on-call incident response solution? Sign up for a personalized demo with one of our product experts or go at it yourself in a 14-day free trial.

Make on-call suck less.

GET A DEMO

FREE TRIAL


splunk>
+
VictorOps