splunk> + VictorOps

WHY DEVOPS MATTERS:

Collaborative Transparency in
Incident Management

VictorOps is purpose-built incident management software for DevOps-focused teams. Sign up for a 14-day free trial to start making on-call suck less.

TABLE OF CONTENTS

THE ORIGINS OF DEVOPS

In an attempt to standardize and streamline IT service management (ITSM), the first version of the IT Infrastructure Library (ITIL) was published in the 1980s[1]—shortly after the first personal computer was invented.[2] Then, in 1990, the world wide web was created.[3] And, over the years, ITIL has been updated to keep up with technological changes, most recently in 2011.[4]

However, the rate at which technology is created and adopted has increased exponentially over the past 50 years.[5] With increasingly complex architecture and market demands, along with constant end-user demand for consistent uptime and availability, the need for reliable, battle-ready infrastructure and software grows. Customers expect not only reliability, but they also expect innovation. So, it stands to reason that our approach to software development, security, and IT operations would need to grow in tandem, accommodating reliability in parallel with consistent development of new features and updates.

Enter the DevOps movement.

DevOps: The combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market." (AWS)

The idea of building an agile, collaborative relationship between developers and operations teams who support CI/CD and system reliability existed before DevOps became a formalized term. In fact, one of the first references to DevOps ideals comes in 2008. A software developer named Andrew Shafer was going to hold a session about "Agile Infrastructure," but thinking there was no interest in his talk, he didn't attend his own session. However, one man named Patrick Debois was interested so he found Andrew Shafer in the hallway where they had a discussion. Later, Debois and Shafer founded the Agile Systems Administration Group.[6]

But then, early DevOps methodology was truly popularized when it popped up in a talk from John Allspaw and Paul Hammond, Flickr's SVP of Technical Operations and Director of Engineering respectively, about deepening dev and ops collaboration. Then, inspired by the talk, Patrick Debois held the first-ever DevOpsDays event in Ghent, Belgium in 2009.[7] From there, DevOps became an official term and continued to expand in popularity across the globe.

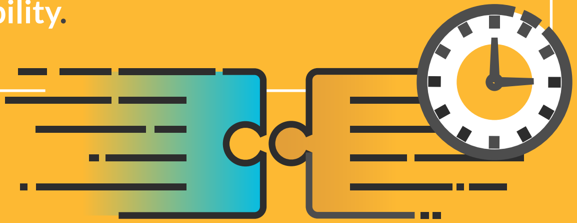
[1] https://en.wikipedia.org/wiki/ITIL#History
[2] http://lowendmac.com/2014/personal-computer-history-the-first-25-years/
[3] https://www.history.com/news/who-invented-the-internet
[4] https://en.wikipedia.org/wiki/ITIL#History
[5] https://ourworldindata.org/technological-progress
[6] https://blog.newrelic.com/engineering/devops-name/
[7] https://devops.com/the-origins-of-devops-whats-in-a-name/

WHY DEVOPS MATTERS

Every year since DevOps was cemented as a distinct term, people have continued to adopt DevOps principles, iterated on those principles, and helped the community grow. But due to the infancy of DevOps in practice, teams can easily fall into a trap of improper implementation and misuse of DevOps practices and values. So, we created *Why DevOps Matters* as a guide for facilitating a DevOps organizational change dedicated to collaboration and transparency.

Why DevOps Matters defines what DevOps actually means, demonstrates how leveraging DevOps improves organizational cooperation and visibility, and ultimately, why the effective usage of DevOps allows you to build reliable systems faster.

Read our own story of how we, VictorOps, adopted a culture dedicated to collaboration and transparency, leading to a balance of delivery speed and maximum uptime in Build the Resilient Future Faster: Creating a Culture of Reliability.



SO WHAT IS DEVOPS REALLY?

At its core, DevOps refers to an agile, collaborative relationship between development and IT operations teams.

So, DevOps is really a mindset, not a defined process or role. If you start asking around, you'll find a number of teams with DevOps engineers or a dedicated DevOps team. However, an organizational structure such as this has trouble living up to the potential of a truly DevOps-centric culture. DevOps is really about finding ways to deepen cross-functional transparency and collaboration—and with continuous improvement at the core, you're never truly done.

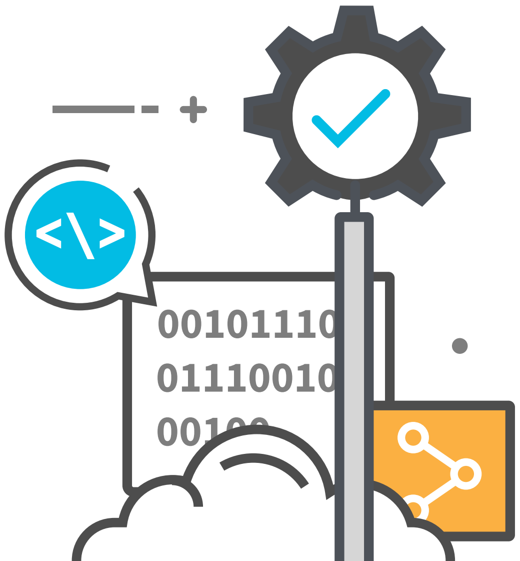
DevOps is an ideology dedicated to continuous improvement, collaboration, and transparency–values required by every person in your company. In DevOps, developers are held accountable for the code they write and are responsible for taking on-call duties. When DevOps is a cross-functional practice—a culture adopted across teams—it has the power to transform the way code is shipped, uptime is maintained, and incidents are managed.

MAINTAINING THE CODE YOU WRITE

A DevOps culture starts with multiple teams made up of members from multiple disciplines continuously building and deploying new features and services. Everyone from product managers to QA engineers to mobile developers to operations engineers can be on one sub-team of a DevOps-minded team.

In a DevOps culture, engineers are responsible for maintaining the code they write, meaning they'll resolve incidents more efficiently, and care more about baking reliability into everything they build.

For instance, if there's an error with a mobile app feature in production, who's more equipped to remediate that problem, the mobile engineer who built the feature or a siloed operations teammate who's never worked on this particular issue before? Even further, if that particular mobile engineer is unavailable, with multiple sets of eyes on the release of the feature, it's likely another colleague involved in the feature release would have insight into a solution for the error.



AVOIDING A SILOED DEVOPS TEAM

There's no "correct" organizational structure for DevOps. But, in order for DevOps to function effectively, every person in the organization needs to buy into the concept. If you create a separate DevOps team of DevOps engineers, you're simply creating another siloed team.

In this setup, your developers and operations teams not only have to collaborate with one another, but they also need to find ways to collaborate with the DevOps team. If your DevOps team isn't involved in either the creation or the deployment of code, they're just as blind as a siloed ops team would be.

While this is a great first step in the right direction and shows you're starting to think about DevOps, this structure won't tighten collaboration and offer more visibility into current operations.

Building smaller teams within the greater team, containing end-to-end skill sets, allows you to build faster and hold teams accountable for the services they produce. This type of DevOps promotes a culture of code ownership, where developers are on-call, providing additional exposure to a system in production and establishing a deeper level of system understanding; all this transparency, in turn, leads to faster incident resolution.

HIRING FOR DEVOPS

Because DevOps is philosophy, not a specific role, you need to look for a few additional things as your teams grow and move toward a culture of DevOps.

Of course, you need to make sure the person you're interviewing has the specific skill set required for their role (e.g. mobile development, data engineering, platform engineering, etc.). But, you also need to make sure they'll fit into a culture of collaborative transparency and won't shy away from the accountability associated with that culture.

Specific traits to look for when building a DevOps-focused team include:

- Interviewee shows buy-in to your team and culture
- Interviewee shows buy-in to your product
- Interviewee shows technical excellence
- Interviewee shows traits of a T-shaped person (i.e. has a wide scope of understanding in multiple concepts, but highly focused technical skills in one area as well)
- Interviewee shows a hunger to continuously improve, learn new programming languages, and try different techniques

A DevOps culture is built over time through intelligent hiring of open-minded people dedicated to learning new things, continuously improving, and taking accountability for the services they create.

- DevOps starts with the company culture you build
- DevOps blossoms organically when you hire collaborative individuals focused on continuous improvement and a desire for transparency
- DevOps-oriented team members will embrace on-call responsibilities, incident response, and the accountability of maintaining the code they write

Incident management is only one small piece of DevOps, but it's an important one. On the next page, we'll cover some incident management basics and later discuss how DevOps fits into the full software development lifecycle while creating a holistic incident management process.

AN INTRO TO INCIDENT MANAGEMENT

Incident Management: The method of using people, processes, and tools to identify, diagnose, respond to, and remediate problems in both hardware and software across applications and IT infrastructure.

It's naive to assume you'll never encounter an incident or experience downtime. Teams always need to have incident management processes and tools that help people fix issues faster. And because Fortune 1000 companies pay between $1.25 and $2.5 billion in annual unplanned application downtime costs , resolving incidents quickly is imperative.

Let's take a look at some key functionality in incident management tools that makes on-call suck less, driving collaboration and improved system visibility:

Functionality in Incident Management Tools

- On-call scheduling
- Intelligent alert routing and transformation
- Automated escalations
- Scheduled overrides
- Centralized incident history
- Incident analytics
- Seamless integration with monitoring and communication tools

An incident management tool should centralize incident data and offer a platform for communication. An incident management solution with this functionality increases collaboration and transparency across teams and systems.

As teams embrace a culture of DevOps, they're able to become more proactive about incident management and build more robust applications and services. Later, we'll cover the specific ways that DevOps can help shorten the incident lifecycle.

Earlier we touched on structuring DevOps in your organization, but now we should dive into some more specific core values and common misconceptions of DevOps.

DEVOPS CORE VALUES

As we'll start to explore, you'll see how closely DevOps core values relate to an efficient incident management process. In a DevOps culture, collaborative teams and transparency of information are key components. Information should be pertinent and available to nearly anyone on the team who needs it, and teams need to have tools and processes in place to help them work cohesively around this data. With access to necessary data and a culture of open collaboration, teams can move quickly, shorten feedback loops, and focus on larger projects.

With that said, let's dive deeper and explore all of the essential core values of DevOps:

- **Exposure** - DevOps helps teams work collaboratively throughout the software delivery and incident lifecycles, giving teammates more visibility into the entire CI/CD pipeline. Not only will engineers have a deeper understanding of the services they're building as they write the code, but everyone in the organization will have better historical knowledge of the way the system is built. This deep exposure allows the team to deploy new features faster and remediate incidents efficiently.
- **Accountability** - Because everyone is responsible for the code they write in DevOps, there's increased accountability for the reliability and ownership of systems. A deeper accountability and understanding of traditional operations responsibilities help developers build transparent systems with failure in mind.
- **Automation** - In a DevOps culture, anything that can be automated should be automated. But it's important to mention that automation should be focused on improving human workflows. Before automating a task, teams should consider: How can teams leverage automation to make people more successful? Automation is especially useful when it's used in-line with human workflows and historical incident data, bringing context into one location, making the most of people operations, and optimizing future processes.
- **Collaboration** - Open conversations need to be held cross-functionally within any DevOps-focused organization. The more involvement you can get from additional teams, the better. Ask questions and cultivate a culture of collaboration between different disciplines and people. Deeper collaboration between development, security, and IT operations will expose vulnerabilities and pain points in your infrastructure. Armed with this information, teams can implement more resilient engineering practices and add reliability to their systems.
- **Transparency** - A centralized source of truth for all communication and incident history helps less-informed teammates to understand a situation if something comes up while they're on-call. By building transparency into workflows, opening up visibility into cross-functional communication, and establishing a culture of open information, team members can get what they need when they need it. With a transparent activity history in one place, you can make positive organizational changes.
- **Continuous Improvement** - Last but not least is continuous improvement. Every single person on a DevOps-minded team needs a desire to continuously improve–whether they're improving processes, people operations, or tooling. In association with the other DevOps values, teams can constantly iterate on the way they do things, and search for better alternatives.

DEVOPS MISCONCEPTIONS

You've seen the values of DevOps, now let's look at what DevOps is not.

Many people like to discuss the idea of DevOps best practices. But, the idea of best practices contradicts everything about DevOps. With so many factors at play (e.g. organizational structure, company size, infrastructure and tech stack, development speed, company's maturity, etc.), DevOps best practices simply can't apply to everyone. DevOps is more of a mindset–a constant drive to improve the technology, processes, and people operations associated with the services you build.

DevOps may be implemented in a number of ways, but there are some common misconceptions you may hear. Many of these misconceptions can render DevOps far less effective than it should be. Check out some common misunderstandings and how these ideas could negatively affect your DevOps culture if you let them:

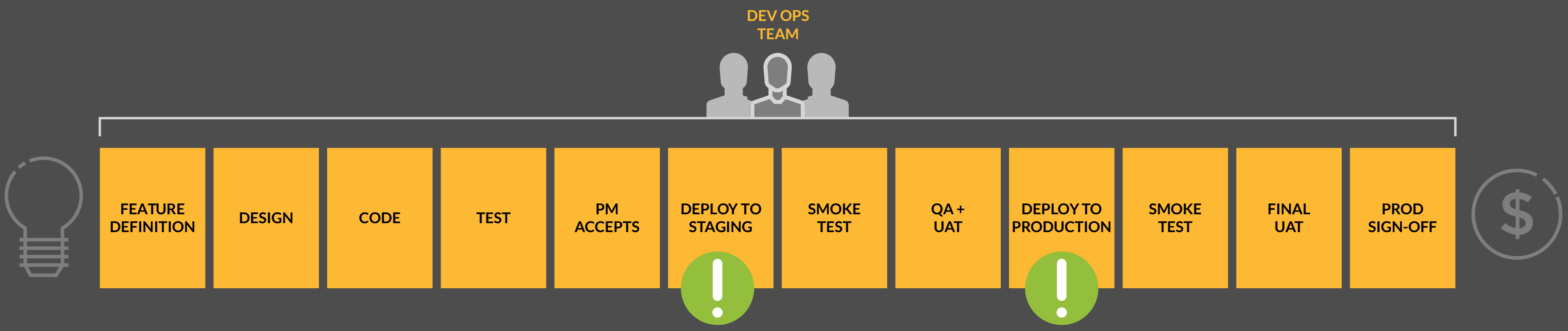
- **DevOps is a specific role** - As we've mentioned previously, DevOps is not one specific role. A team dedicated to DevOps is made up of a number of engineers from different disciplines working cross-functionally to build reliable systems quickly and efficiently. You can't expect one engineer to have all the skills necessary to effectively work on any issue across your entire infrastructure. That's why well-structured, collaborative teams of people with complementary skills and experience tend to be far more effective than organizations with siloed "DevOps teams."
- **DevOps is only for smaller organizations** - When properly implemented, DevOps works across companies of any size. Through deeper collaboration and transparency of workflows, organizations can identify problems as they scale and adapt team structures. Every team grows differently, so there isn't one way to structure teams for DevOps. Just remember to focus on continuous improvement by testing out different structures and optimizing workflows based on prior successes.
- **There's one specific process for DevOps** - There's no one-size-fits-all approach to using DevOps. No single process will work for every team. DevOps is about creating visibility into how things are working, collaborating across multiple teams, and improving the process from there. No matter how well you think your process is working, there's always room for improvement.
- **DevOps is an implementation of a process or tool** - DevOps is a belief that improved collaboration, transparency, and a desire to be better will drive success. Don't ever believe some golden process or tool will lead you to success. As long as your company follows the core DevOps values and personalizes them to fit the needs of your teams, you're on the right path.
- **Everyone in DevOps has to be a SysAdmin and a Developer** - There's a common belief that engineers on a DevOps-minded team need to be able to do everything. This is a fundamental flaw in the way teams are structured with DevOps Engineers as a specific role. One person can't effectively handle all the tasks of a traditional system administrator as well as the responsibilities of a developer. And even if you could find one person who could do it all, what do you do if they're on vacation or they leave the company? Collaborative teams bridge the gap, allowing for consistent development speed and system reliability.

DEVOPS IN THE SOFTWARE DEVELOPMENT LIFECYCLE (SDLC)

Traditionally, product managers would determine the product roadmap. They'd pass the tasks over to the development team. The engineers would work on their specific portion of the app or service and then pass it on to the security and IT operations teams. Without any context, the operations team would deploy the code, fix anything that goes wrong, and then maintain the service going forward.

Sounds like there could be a lot of disconnect, right? Due to the increasing pace of CI/CD and innovation, regardless of any disconnect, this process for building and maintaining services simply doesn't work anymore. Why are operations teams bearing the brunt of the responsibility without any input into development? DevOps allows operations teams to offer input into development, while developers also take on-call responsibilities and gain exposure to systems in production, helping everyone gain a more holistic understanding of the way their system functions.

Let's take a look at how DevOps plays into the SDLC:



THE FULL DEVOPS LIFECYCLE

Just as DevOps bolsters transparency and collaboration throughout the entire software delivery lifecycle, it also applies to incident management. While developer insight into IT operations allows for faster development, IT operations insight into development helps shorten the incident lifecycle. Let's look at the five steps of the incident lifecycle to better understand why DevOps is beneficial:

Many teams simply deploy code to production, set up basic monitoring tools, and walk away–leaving someone else to handle the maintenance of the service. For the DevOps lifecycle to truly work, teams must take ownership of the code they write and the day-to-day upkeep of the service. Monitoring, as a concept, ought to include a notion where people actively maintain the services they've created.

By preparing actionable steps and system visibility for potential consequences tied to deployment in highly integrated, complex architectures, DevOps improves the speed of reliable service delivery. Teams require systems for collaboration and system visibility to lead a more proactive approach to the SDLC and incident lifecycle.

DEVOPS IN THE INCIDENT LIFECYCLE

Step 1: Detection - With both developers and operations engineers on-call, the team can better fine-tune monitoring tools and thresholds to detect incidents faster. Through collaboration and more visibility into incident management and software delivery, DevOps helps people to proactively add reliability into their services. Through deeper exposure and more visibility into a system's inner workings, teams become capable of detecting incidents more quickly–sometimes before they even happen.

Step 2: Response - Once you've fine-tuned your methods for detection, you can start improving the incident response process. Oddly enough, incident response is often overlooked–even though it accounts for 73% of the incident lifecycle on average.[9] Also, incident response involves the largest time commitment for humans over any of the other steps of the incident lifecycle. DevOps allows people to be more efficient in getting contextual alerts to the right person at the right time. Constantly search for ways to make incident response easier for your teammates and work to make on-call suck less. DevOps-focused teams are already shortening incident response times with VictorOps–sign up for a 14-day free trial today.

[9] https://victorops.com/ebooks/state-of-on-call-2015/

Step 3: Remediation - The actual fix of an incident should be a very small portion of the incident lifecycle. If detection and response are efficient, the responding engineer will have the information or assistance they need to quickly remediate the problem. Centralizing detection, response, and remediation in one location will make the final two steps of the incident lifecycle much easier.

Step 4: Analysis - With your centralized data, you can analyze exactly what happened during an incident. You can review the way people responded to the incident, how long it took to detect the issue, and what steps were taken to resolve the issue. Based on this data, you can conduct detailed post-incident reviews and walk away with action items. These action items can range from silencing unactionable alerts, preventing an incident from occurring again, or providing resources for faster remediation the next time around–or all of the above.

Step 5: Readiness - With a strong incident analysis, your team will be more prepared for future incidents. Processes can be adjusted and on-call engineers should be given better resources fo r incident detection, response, and remediation. Being prepared for failure is the best way to ensure higher levels of uptime and provide service reliability to your customers.

Read the Guide: ***How DevOps Plays Into the Incident Lifecycle***

END-TO-END COLLABORATIVE TRANSPARENCY THROUGH CENTRALIZATION

In order to optimize the effects of DevOps within your company, you need to centralize workflows. If software delivery, incident response, and chat take place in three different tools, understanding how your team works can become difficult. If you don't know how your team is currently doing things, improving workflows and implementing helpful tools becomes nearly impossible.

So, let's look at the three main things you need to centralize in order to make the most of your DevOps culture:

- Software delivery lifecycle data
- Incident lifecycle information
- Real-time communication and chat history

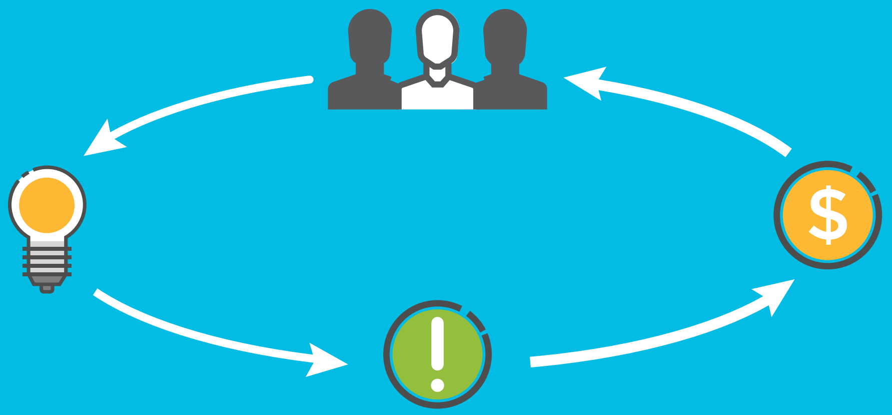
If you can find this information in one place, you can accurately assess the efficiency of your teams over time. This leads to a faster, more reliable CI/CD pipeline as well as improved incident response and remediation.

From the seedling of an idea to an actual feature in production, DevOps creates a culture that drives collaboration and transparency between development and IT operations teams. And these values don't stop with the engineering team. Increased transparency on the engineering side allows for deeper collaboration with business teams. Armed with this transparency, sales, marketing, and customer support teams can improve the way they interact with customers and prospects, driving increased revenue.

The best part is–this collaboration flows both ways. Engineering teams get more input from business teams into the features and services prospects and customers want. Additional collaboration between business and engineering teams helps prioritize the product roadmap, allows you to build features faster, and helps separate you from the competition.

SO WHAT DOES THE DEVOPS TEAM DO?

Well, everything! A collaborative DevOps-focused team, armed with highly transparent tools and workflows can better manage software delivery and incident management. Also, improved collaboration and transparency helps your team scale as you grow. When new teammates start, getting them up-to-speed quickly on how your system is built and some weaknesses of the infrastructure is imperative to maintaining CI/CD and overall system reliability.



Then, when engineers take their first on-call shift and encounter an incident, they don't feel as if they've been stranded on an island. So, let's also look at some of the ways that DevOps can help translate into an efficient on-call incident management solution.

END-TO-END HOLISTIC INCIDENT MANAGEMENT

A system or process for incident management can only get you so far. When your team embodies the DevOps values, incident management will move to the next level. By combining a centralized platform for incident data and communication within a DevOps culture, teams will have deeper transparency across all workflows from beginning to end.

A tool is only as good as your process, which is only as good as your people. People are the core driver of incident response and remediation, so you need to empower them. Don't force people into workflows they won't use; bring the tools and processes to the places you're already working.

Essential DevOps Tools

- **Monitoring & Alerting:** Monitoring and alerting is just the beginning. Everything should be built with monitoring and alerting built in, but the monitoring and alerting setup shouldn't be reactive. Not only should your monitoring data and log analytics be detailed, but you need to find ways for improving visibility into architecture and service health, then make sure incident resolution steps are readily available. Most monitoring setups are made up of a combination of APM and system monitoring tools like New Relic, AppDynamics, Splunk, and Nagios.
- **Communication:** DevOps relies heavily on collaboration and transparency, making ease of communication across the organization a key component of efficient teams. By allowing people to communicate through multiple channels of their choice, you ensure communication

is transparent cross-functionally. Open communication leads to better idea sharing, deeper collaboration throughout the SDLC, and faster incident response when an alert comes in.

- **CI/CD:** As CI/CD is an essential element of DevOps, you need tools like Jenkins or Travis CI to help orchestrate, automate, and manage the continuous integration and deployment of your services. You can leverage associated processes with these tools to help you manage and automate a number of tasks related to changes and testing for projects going into production.
- **Configuration Management:** Tools like Chef or Puppet can be used to ensure consistency of assets across your entire operational environment, track interdependencies between services in complex infrastructures, and govern changes made to your system. Configuration management tools improve visibility and agility to service delivery, allowing teams to move faster and build more reliable systems.
- **Security:** While a number of organizations don't always consider security a specific part of DevOps, security should always be integrated across the entire DevOps lifecycle. Security tools can help you monitor for vulnerabilities, insider threats, external attacks, data loss, and much more. Every DevOps-centric organization needs security tools such as Signal Sciences or Threat Stack mixed in with their software delivery and incident management toolchain.

Core Functionality of DevOps-Centric Incident Management Tools

- **Automation:** Everything that can effectively be automated should be automated. Set alert rules that route alerts and notify the proper on-call engineer or team. You should also work automation into native chat through ChatOps functionality. This way, you can work manually with team members, escalate incidents through chat, or run commands directly from your chat application. Automation can be used across your entire incident management stack to streamline workflows and make on-call suck less.
- **Centralization:** Centralized monitoring, alerting, and chat data creates transparent, collaborative workflows for everyone on the team. Incidents can be discussed in-line with alert context, allowing anyone on the team to get involved. Centralization of workflows also improves visibility for the broader team into escalations and on-call schedule changes.
- **Customization:** Systems are built over time, potentially resulting in a mishmash of homegrown solutions and integrated third-party services. Also, DevOps team structure can look different. Accordingly, an incident management solution needs to be highly customizable. Does it have flexible APIs, webhooks, and endpoints that can work with a number of services? Does it provide functionality to help developers and operations engineers do their job better?
- **Analytics:** In a centralized platform, you can track important incident management KPIs over time. This helps you refine the process and determine ways you can improve. Metrics such as time spent on-call, MTTA/MTTR (mean time to acknowledge/resolve), and incident frequency can show you what's working and what's not.
- **Collaboration:** If incident management solutions aren't focused on improving human collaboration, they aren't fit for a DevOps culture. A DevOps-focused incident management solution will provide an avenue for communication and visibility across the entire incident lifecycle. The team needs to be able to access information faster and work around the appropriate data with colleagues.

INHERENT SRE

Site Reliability Engineering (SRE):

“Fundamentally, it’s what happens when you ask a software engineer to design an operations function.”

- Ben Treynor, VP of Engineering at Google[10]

SRE is an essential function of every engineering and IT operations team. But, you may notice this structure looks an awful lot like a siloed team of DevOps engineers. When a culture of DevOps is implemented at your company, SRE will become an inherent trait of everything you build.

With a fully collaborative DevOps-oriented team broken into sub-teams, transparency and collaboration improves across the entire SDLC and incident lifecycle. DevOps exposes engineers to systems in production, improving overall system understanding, workflows, and communication–leading to faster feature delivery, incident response, and deeper system reliability. In this culture, a group of engineers are constantly exposed to systems in production and collaborate around feature releases, from product roadmap to deployment. With DevOps, SRE doesn’t need to be a specific team or role within your organization, but it always needs to be part of your way of thinking.

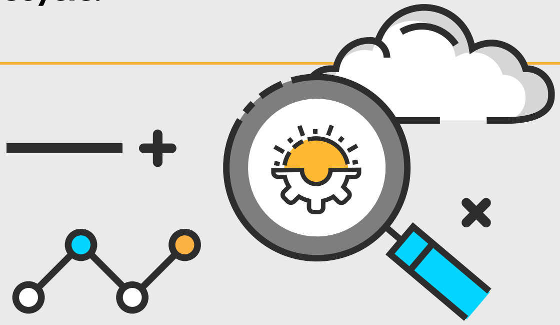
A true DevOps culture will view SRE as an inherent piece of development and incident management workflows.

[10]https://landing.google.com/sre/interview/ben-treynor.html

How We Do SRE

Instead of a solely dedicated SRE team at VictorOps, we’ve implemented an SRE council. The SRE council is formed to expose more engineers to the duties of SRE, helping disperse specific SRE-related tasks across the organization, while also spreading the importance of building reliable systems. By creating a council of diverse viewpoints and disciplines, then cycling members every six months, the emphasis on SRE spreads across the entire organization.

Read more about our approach to SRE and how we made it an integral part of our SDLC and incident lifecycle.



COLLABORATIVE TRANSPARENCY LEADS TO CONTINUOUS IMPROVEMENT

If you only take one thing from this eBook, remember that DevOps is all about continuous improvement. A dedication to building a culture of DevOps, including collaboration, code ownership, and transparency, will help your team continuously improve and build reliable software—faster.

Continuous improvement happens through deep analysis of your people, processes, and tooling. By gathering data, centralizing this information, and taking time to actually use the collective information, you can iterate and learn to become a better team.

A few important practices to help you continuously improve:

- **Centralized Timeline:** Consolidating alert data with chat history, inside of a blameless DevOps culture, leads to more effective real-time incident response. The nature of centralized real-time collaboration and alerting tools allows anyone on the team to find the alert data they need when they need it. Centralized system information and communication allows you to almost immediately turn data into insights and insights into incident remediation actions. Then, over time, the entire team can continuously improve workflows and processes based on historical data.
- **ChatOps and Chat History:** A detailed, integrated chat history will show you exactly how your team is working. You should consolidate communication through SMS, email, chat apps, and maybe even phone call logs/video chat recordings, into one single communication history. This will help you understand how people are working together and what you might be able to do to improve the process. Also, when digging into specific incidents, you can see exactly how your team reacted, not just what was going on with the technology.
- **Open Ecosystem:** An open ecosystem allows for a free flow of new ideas. Teams need to work cross-functionally, share insights, and provide constructive feedback. Only through open communication and diverse viewpoints coming from multiple areas of the business can an organization grow.
- **Establishing a Blameless Culture:** The nature of CI/CD and Agile software development means you can't avoid incidents. But, creating a culture of blame and finger-pointing when something goes won't exactly speed innovation. By establishing a blameless culture, you can really dig into your incident management and focus on treating the problem, not the symptom.
- **Post-Incident Review (PIR) vs Postmortem:** Whenever an incident occurs, members of the greater team need to actually sit down and take time to understand the problem. Anyone looped into the resolution of the incident must attend a PIR, but it should be an open forum to anyone with suggestions. What steps can be taken to shorten remediation time? What caused the incident to occur? Was it even an actionable alert? Post-incident reviews are one of the single most important ways to iterate on your processes and continuously improve.

Start conducting your own actionable post-incident reviews by downloading our free post-incident review template.

DEVOPS MATTERS. A LOT.

DevOps isn't a specific role or a defined set of processes. DevOps is a culture and method for continuous improvement, collaboration, and transparency to drive success. Success not only in building reliable software and Agile engineering practices but also success in driving revenue and operational efficiency company-wide.

Don't buy into the belief that there's one way to "do DevOps." DevOps will take many forms across every organization, but it will always adhere to the core values. A deep understanding of common DevOps misconceptions and core values will allow you to implement DevOps at your company.

Enter VictorOps

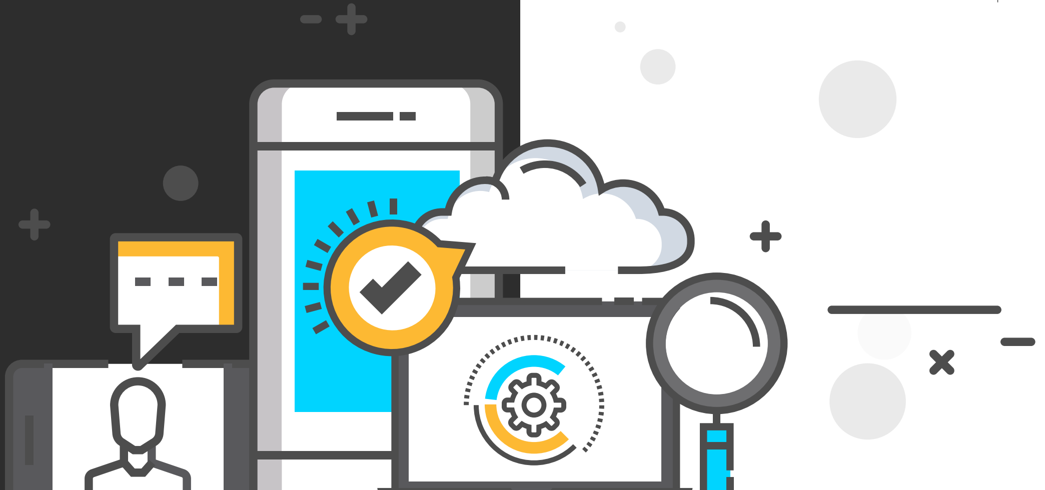
Get an incident management tool that encourages a culture of DevOps collaboration and transparency. Surface software delivery and incident details across your entire organization with a centralized platform for on-call schedules, intelligent alert routing, and chat to make on-call suck less.

Ready to start using a holistic end-to-end incident management 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