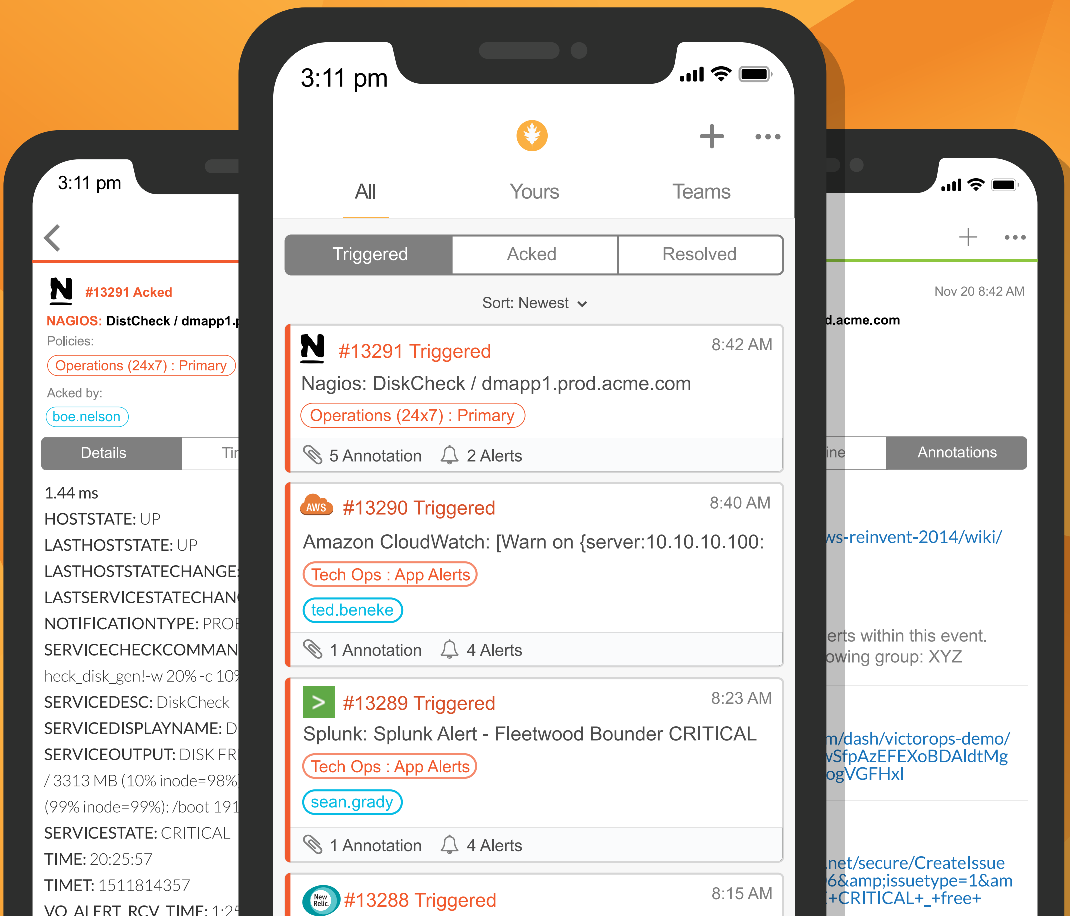
VictorOps

THE INCIDENT MANAGEMENT BUYERS GUIDE


3:11 pm

All Yours Teams

Triggered Acked Resolved

Sort: Newest

#13291 Triggered 8:42 AM
Nagios: DiskCheck / dmapp1.prod.acme.com
Operations (24x7) : Primary
5 Annotation 2 Alerts

#13290 Triggered 8:40 AM
Amazon CloudWatch: [Warn on {server:10.10.10.100:
Tech Ops : App Alerts
ted.beneke
1 Annotation 4 Alerts

#13289 Triggered 8:23 AM
Splunk: Splunk Alert - Fleetwood Bounder CRITICAL
Tech Ops : App Alerts
sean.grady
1 Annotation 4 Alerts

#13288 Triggered 8:15 AM

3:11 pm

#13291 Acked
NAGIOS: DistCheck / dmapp1.p
Policies:
Operations (24x7) : Primary
Acked by:
boe.nelson

Details Ti

1.44 ms
HOSTSTATE: UP
LASTHOSTSTATE: UP
LASTHOSTSTATECHANGE:
LASTSERVICESTATECHAN
NOTIFICATIONTYPE: PROE
SERVICECHECKCOMMAN
heck_disk_gen!-w 20% -c 10
SERVICEDESC: DiskCheck
SERVICEDISPLAYNAME: D
SERVICEOUTPUT: DISK FR
/ 3313 MB (10% inode=98%
(99% inode=99%): /boot 191
SERVICESTATE: CRITICAL
TIME: 20:25:57
TIMET: 1511814357
VO_ALERT_RCV_TIME: 1:2

Nov 20 8:42 AM
d.acme.com

ine Annotations

ws-reinvent-2014/wiki/

erts within this event.
owing group: XYZ

m/dash/victorops-demo/
vSfpAzEFEXoBDAldtMg
ogVGFHxl

net/secure/CreateIssue
6&amp;issuetype=1&am
+CRITICAL+_+free+

Your Guide to Incident Management

Incident: a problem, represented by an alert, that could negatively impact customers, your employees, and/or the stakeholders inside or outside of your organization.

In order to stay competitive in today's market, businesses are expected to innovate—quickly. Many engineering teams feel pressure to build, deploy, and operate services with increasing speed. High performing teams innovate faster and maintain their sanity because they're able to quickly recover from incidents.*

As we move from agile development to rapid deployment, teams need to think beyond a reactive operations center. That's why choosing the right incident management solution is more than just icing on the cake to a successful DevOps culture; it's the cornerstone to engaging high-performing engineering and ops teams who champion uptime and own on-call—instead of fear it. Ultimately, rethinking—and retooling—your approach to DevOps and incident management is imperative to delivering the world-class customer experiences that keep businesses relevant.

The purpose of this buyer's guide is to discuss why progressive, high-performing teams choose to invest in high-performance incident management software. From the challenges across the SDLC to specific incident management product features, we'll lay out everything you need to consider when choosing an incident management solution.

"Reliability is the single most important feature we provide."
- Dan Jones, CTO, VictorOps

The Challenges

Issues Commonly Faced by Organizations without an Established Incident Management Process

Alert Noise and Fatigue

Disorganized Communication

Poor Alert Flow from Disparate IT Systems

Siloed Communication

Wrong People Being Alerted

Unprepared for Crisis

Disconnected Workflows

Repeating Previous Mistakes

BUILDING A CULTURE OF URGENCY AND AVAILABILITY

High availability is essential to business success—an issue complicated by the increasing deployment demands of a highly competitive market. Accordingly, investing in processes to ensure near-zero downtime alongside rapid deployment is mission critical for the entire engineering and IT department.

Here, we break down how incident management is key to maintaining a culture of availability without slowing down the innovation process—and how DevOps is the essential piece for successfully executing this shift.

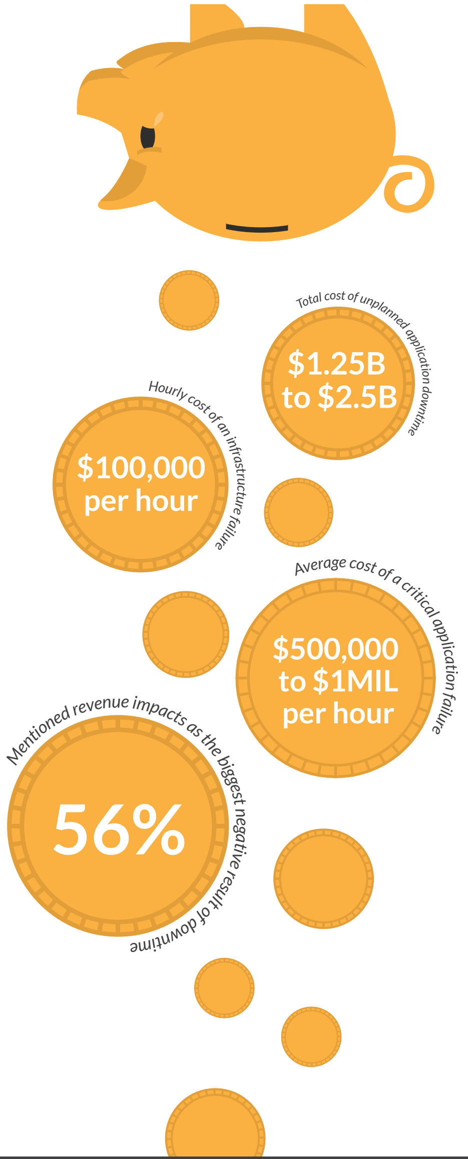
The Negative Economic Impact of Downtime

For the Fortune 1000, the average total cost of unplanned application downtime is $1.25B to $2.5B annually. The average hourly cost of an infrastructure failure is $100,000 per hour. The average cost of a critical application failure is $500,000 to $1 million per hour.*

These aren't outliers limited to the enterprise. Outages (and their subsequent costs) affect companies large and small. These types of errors are full of negative externalities, including branding and overall customer trust. For example, in 2017, GitLab lost a massive amount of customer data after an error (and subsequent failures of multiple redundant backup protocols). Customer projects, comments, and other data were all gone. While source code repositories were safeguarded, it was problematic for a company whose business involved data stewardship.**

In the VictorOps 2017 State of On-Call Report, we learned 56% of respondent mentioned revenue impacts as the biggest negative result of downtime in their business. Of course, downtime is more than just revenue, the repercussions of a major outage are felt throughout the business:

See content here


Total cost of unplanned application downtime
$1.25B to $2.5B
Hourly cost of an infrastructure failure
$100,000 per hour
Average cost of a critical application failure
$500,000 to $1MIL per hour
Mentioned revenue impacts as the biggest negative result of downtime
56%

* DevOps and the Cost of Downtime: Fortune 1000 Best Practice Metrics Quantified, IDC
**Postmortem of database outage on January 31, GitLab

Competitive Advantage of Minimal Downtime

More advanced companies use historical incident data to proactively prepare teams to resolve events faster, and to prevent those events in the first place. This in turn becomes a competitive advantage as highly functional "on-call" teams help protect revenue loss, maintain brand reputation, and drive customer satisfaction.

High-performing teams tend to fare far better than competitors when it comes to both throughput and stability.

Recent research demonstrates these high performers are deploying 46x more frequently, with 440x faster lead time from commit to deploy, all while maintaining a mean time to recover (MTTR) that's 96x faster. And change failure rate? It's 5x lower, so changes are ⅕ as likely to fail*

Shift from ITIL: DevOps and Modern IT

The traditional Information Technology Infrastructure Library (ITIL) model was developed in the late 1980s, a time when people were shipped physical disks for application updates. And while not every company then was in the business of selling software, almost every business now relies on running software and delivering online services. Software is disrupting every industry—entertainment, agriculture, finance...* This is where ITIL falls flat. ITIL separates duties and process approvals in an effort to support standardization and reduce duplication of work. This siloed and process-laden approach inherently slows down change. Nevertheless, many organizations still rely on this model, expecting to adhere to SLAs and maintain near-zero downtime despite incredibly rapid deployment demands.

In order to drive innovation, maintain uptime, and support employee growth, ITIL won't hold up in the always-on, 24/7 IT paradigm. Accordingly, we advocate for a DevOps model as a cornerstone of incident management.

DevOps is an approach to work where teams continuously look for methods to evaluate and improve technology, process, and people as they relate to building, deploying, operating, and supporting the value our organization provides. It's a broader shift in mindset that leads to addressing the needs of the business through the lens of the customer. We accomplish this through an increased focus on collaboration, measuring and improving processes, getting customer feedback, and improved transparency.

*2017 State of DevOps Report, Puppet

*Why Software Is Eating the World, Wall Street Journal, August 20, 2011

Bring DevOps Into Your Life

Benefits of DevOps + Modern Incident Management

Combining DevOps with a forward-thinking incident management tool means the end of a sh*t on-call experience.

For Devs: Owning Your Code

1. Empower Development Teams
2. Create More Stable Operating Environments
3. Spend Time Building and Innovating—Not Fixing and Maintaining
4. Improve Overall Quality of Your Code
5. Support Ownership and Accountability, Regardless of Role or Title

For Ops: On-Call That Doesn't Suck

1. Collaborate with Developers Behind the Code
2. Ditch the Shared Pager—Ack and Resolve from Your Own Mobile Device
3. Integrate across Your Toolchain (Monitoring & More) for Centralized Information
4. Access the Context You Need, Quickly—No Vague 2 a.m. Alerts
5. Move at the Speed of Development, without Sacrificing Safety or Efficiency

For the Business: Increase Efficiency and Boost the Customer Experience

1. Stay Ahead of the Competition
2. Limit Downtime & Improve Service Quality
3. Increase Productivity—and Happiness—of IT Staff
4. Drive Quality Communication across Teams
5. Increase Overall Organizational Velocity

DETECTION
RESPONSE
REMEDIATION
ANALYSIS
READINESS

Modern Incident Management Life Cycle

Today's teams must manage incidents across the entire lifecycle—folding in detection, response, remediation, analysis, and readiness. In this section, we'll dive into the five different phases of the incident management lifecycle.

For each stage, we'll cover the definition. Then, we'll discuss how they relate to the features and functionality you need in incident management software to do more than react to alerts.

Stage 1: Detection

Definition: Detection is the observation of a metric, at certain intervals, and the comparison of that observation against an expected value. Monitoring systems then trigger notifications and alerts based on the observation of those metrics.

How It Relates to Incident Management Software:

Simply put, detection is monitoring insights, looking for the signs and signals of an incident.

However, in organizations with legacy monitoring configurations, actually improving detection is tough. Environments are configured with broadly applied, arbitrarily set thresholds. The impact on on-call teams is measurable:

Too many flase alerts + Too many interruptions = Acute Alert Fatigue

For the above reasons, high-performing teams focus on two things in addition to the basics. The first is time series analysis in their monitoring and detection systems. For example, some progressive, in-market solutions offer a time-series database, enabling wide adoption in both new projects and within existing environments. Your incident management tool should be able to seamlessly integrate with advanced monitoring tools to improve measurement fidelity.

The second is an accurate feed of what's happening in your environment. In VictorOps, we call it the "Timeline." A timeline provides continuous data from across your ecosystem as alerts flow through the system, providing a broad, holistic picture for the size, scope, and urgency of any given alert at any given moment in time.

Stage 2: Response

Definition: The response phase is the delivery of a notification to an incident responder via any means and the first steps the responder takes to address the alert. Thus, a detection threshold is passed, an email/SMS/chat/phone call is sent (notification), and someone acknowledges receipt (response).

How It Relates to Incident Management Software:

There are a few key features to ensure the response happens effectively. You can think about these features as on-call essentials or, depending on how thin the feature set is, "basic alerting." Thus, the leading incident management tools in market will offer:

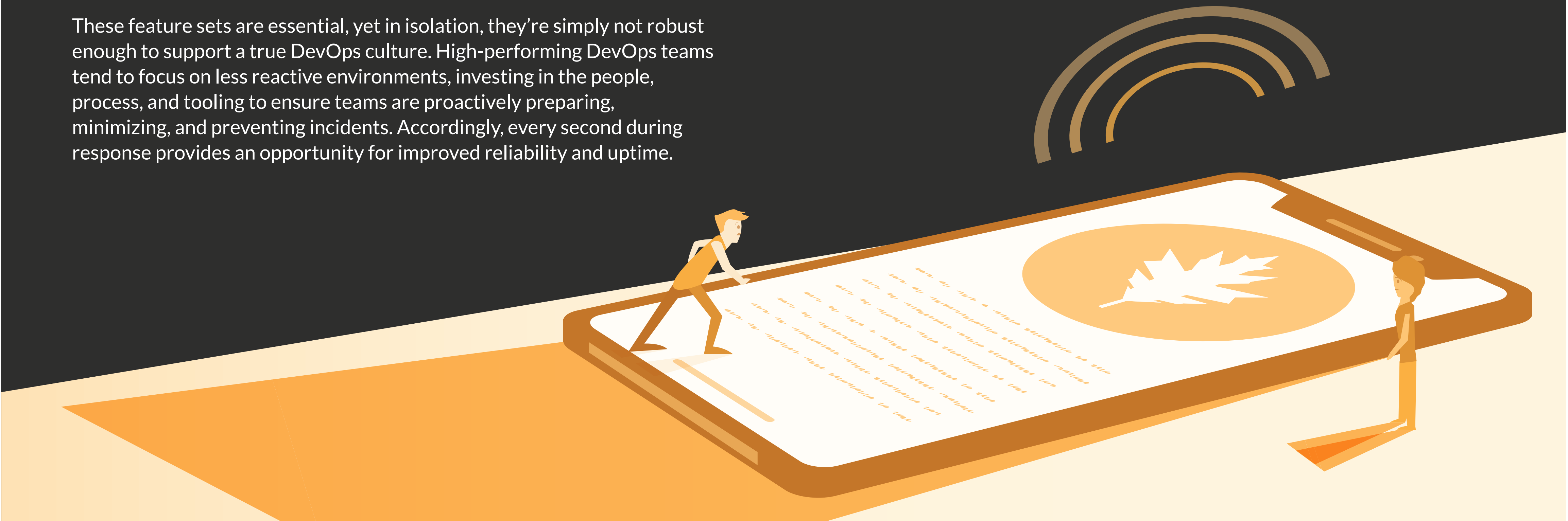
- Dynamic scheduling
- Team-specific rotations
- Automated escalation(s)
- Scheduled overrides

These feature sets are essential, yet in isolation, they're simply not robust enough to support a true DevOps culture. High-performing DevOps teams tend to focus on less reactive environments, investing in the people, process, and tooling to ensure teams are proactively preparing, minimizing, and preventing incidents. Accordingly, every second during response provides an opportunity for improved reliability and uptime.

This is an important point: Developers will not positively respond to (read: adopt) a highly-reactive on-call management tool. The tool needs to offer context, collaboration, and visibility.

Many high-performing teams have found success through ChatOps tooling and workflows that centralize communication andsetup the first responder for success. While receiving a basic notification in Slack/Stride/Mattermost is great, a contextual alert with a visual indication of the current state, plus links to relevant runbooks or dashboards, saves the responder valuable time digging into the error.

When purchasing an incident management tool, buyers should look not only for bidirectional chat integrations and ChatOps functionality, but also the ability to configure alerts to fit team needs—any information present in the alert payload can be used to provide additional details to the on-call responder. Straightforward contextual details attached to each alert will reduce the stress of on-call and provide a next-level technique for resolving incidents faster.



Stage 3: Remediation

Definition: Remediation is the true "firefighting" stage of incident management, where teams aim to quickly diagnose and solve the problem.

How It Relates to Incident Management Software:

A variety of factors impact the length of the remediation stage, often a combination of severity and unknowns. However, the severity of the incident is, of course, often the most direct correlation to MTTR. This "severity" factor may leave teams feeling like the overall time to repair is outside their control; however, there are a variety of ways the combination of incident management software, processes, and team can put the control back in their hands.

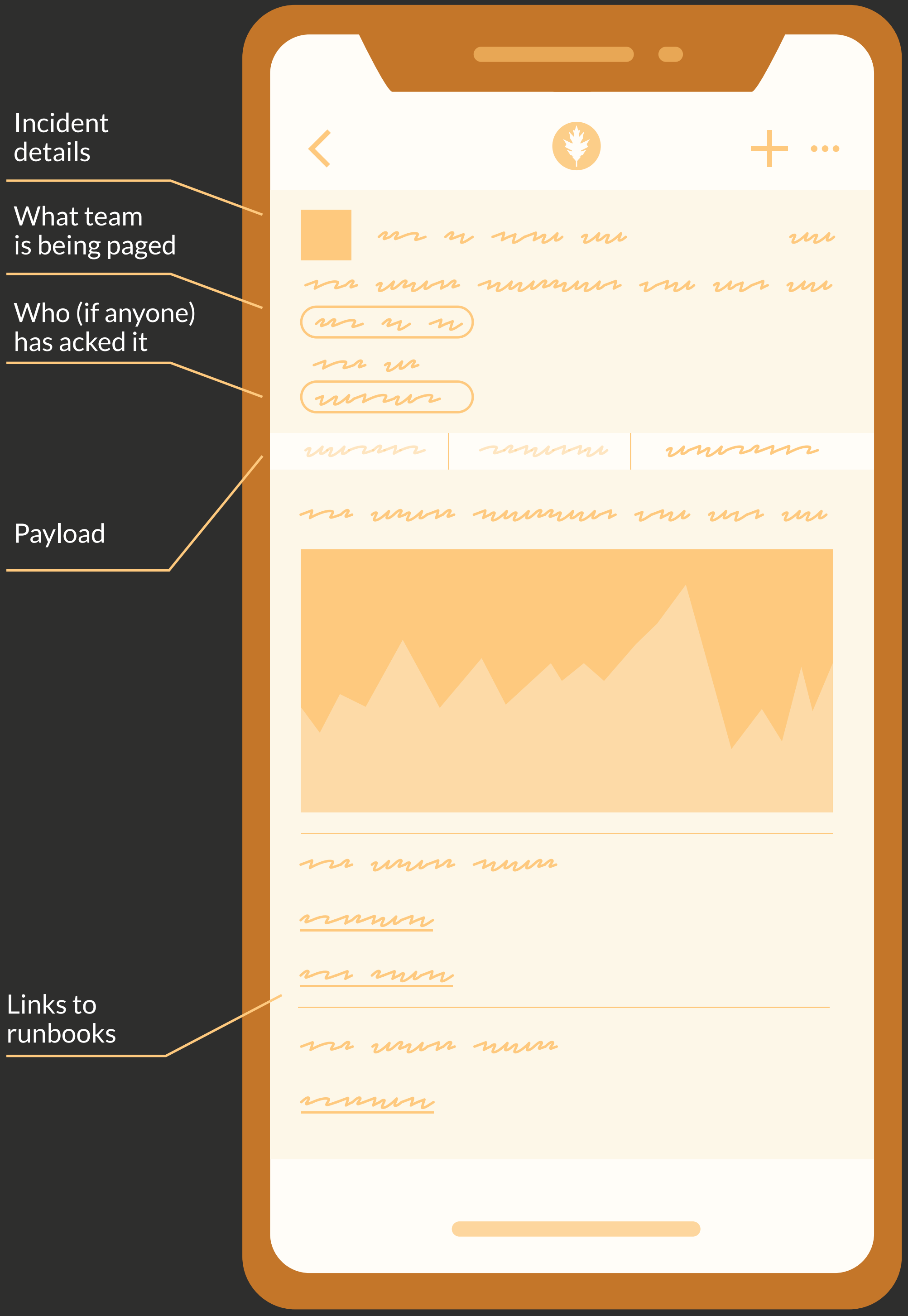
The first piece depends on contextual alerts: what data does the team have access to and, perhaps more importantly, do they have the ability to understand the real-life implications of the data. Contextualization of data allows teams to turn metrics into actionable insights that provide a higher fidelity picture of the incident.

Incident management software can act as a black box for time-series systems (e.g., InfluxDB), log analytics systems (e.g., Splunk), and changes to production (e.g., Jenkins, GitHub).

Regardless of your specific approach to these metrics, your incident management ought to support a holistic picture of your systems and data. Robust integrations, contextual alerts, and runbooks attached to alerts serve as a collective knowledge base for dealing with a variety of issues, no matter your role or tenure.

Learn More About Remediation Here!

A good alert should have the following:



Stage 4: Analysis

Definition: The analysis phase, often referred to as postmortem or post-incident review, is the learning process after an incident is resolved. While the historic approach to this phase has relied heavily on Root Cause Analysis (RCA), increasingly complex systems have led progressive teams away from relying only on single causal entity analysis. Instead, teams are increasingly looking towards models that address system complexities, e.g. Cynefin, to better understand the wholistic, multi-faceted cause of an incident.

How It Relates to Incident Management Software:

When we discuss analysis, there are a few key pieces necessary for Incident Management Software to support a healthy Post-Incident Review (PIR).

The first is the the Incident Dashboard or Timeline, which is helpful for providing a quick view of misbehaving systems before and during the incident; who shipped something to production; who was taking action; what actions was that individual taking; and what communication was happening throughout the incident. All of these pieces serve as critical data for an effective PIR.

Close readers may notice some nuances to words we've chosen (or avoided) as we discuss incident analysis, namely "Post-Incident Review" and "root-cause analysis" (RCA).

Post-Inicident Review is our replacement for post-mortems. You can learn more about our approach to the Post-Incident Review, including why it's so essential for DevOps teams—*here*. The decision to not use RCA mirrors this sentiment based on the current complexity of people and systems. You can read more about our root-cause analysis philosophy in *this blog post*.

The second is also reporting related: Mean time to acknowledge (MTTA) and mean time to resolve (MTTR). MTTA/MTTR reporting allow your teams to visualize and uncover the underlying trends regarding a team's ability to respond to and resolve incidents. By wholistically analyzing the impact of incident volume—and your teams use of the incident management software—you can determine levers to lower MTTA/MTTR specifically and minimize the cost of downtime.

The third is a Post-Incident Review—different than the actual process of an internal PIR, this PIR is a tangible report where individuals, including Leadership, can quickly pull a timeframe of data (no more manual aggregation of emails, Slack, SMS, and monitoring systems) for key learnings. This report facilitates a PIR, or "retrospective", and documents long-term action items. Out-of-the-box PIR reporting allows your team to quickly and easily access monitoring data, system actions, and human remediation to better understand the who, what, when, where, and why of an incident. All of this analysis is essential for the preparedness and readiness required for teams to not only quickly resolve incidents in production, but also improve the reliability of systems to proactively address issues before they occur.

Stage 5: Readiness

Definition: Readiness, the next logical step, is the phase where teams take action to enact improvements to people, process, and technology in order to prepare and, as much as possible, prevent future incidents. Actions taken during this phase vary from architecture and application changes, creating and updating runbooks, or Game Days.

How It Relates to Incident Management Software:

Readiness is the full package of incident management software. As you review the various facets of your team, from systems to processes, does your software enable your team to proactively, collaboratively, and seamlessly address incidents to lower MTTA/MTTR—and minimize the cost of downtime?

In practice, this stage can be the most difficult. Despite a team's best efforts, action items are often left unanswered and day-to-day work supersedes suggetions and improvements. While management often expects full prevention of problems, high-priority projects somehow take the place of supporting these fragile systems.

Of course, one of the best ways to be prepared is to integrate readiness into the software delivery lifecycle (SDLC). Creating a culture where ownership doesn't end when something is shipped into production is an essential piece of minimizing downtime. After all, what's the point of DevOps if the dev team gets to ship something into production at 5pm on a Friday only to leave a Ops team firefighting all weekend long? While the two aren't always complete causational (let's avoid RCA), software releases are the single biggest factor contributing to downtime.*

Teams must find a way to incorporate reliability into releasing, and while you need the right people and process in place, tooling can help. Look for an incident management solution that provides some visibility into the SDLC via developer tooling integrations (e.g., Github, Jenkins). With this visibility, developers and ops alike have a holsitic view of what's happening across systems—including shipments to production.

Additionally, you should take time to optimize your alert stucture, configuring alerts to meet a teams and organizations needs. A noisy alert system or "paging" system can leave teams fatigued and unaware of which alerts actually require action. At VictorOps, our Transmogrifier is our unique alert rules engine, empowering teams to set up a few processes essential to readiness in the face of the most important alerts. Here are a couple key configurations:

1. Alert Rules: Match behavior to fields in alert payloads and create cascading logic to meet often demanding automation needs.

2. Noise Suppression: Using suppression and classification (either critical, warning, or info), unactionable alerts will be visible in Timeline and Reporting but won't disturb users. Alert aggregation further reduces noise by bucketing related alerts into a single incident, adding even more intelligence to your input stream.

3. Alert Annotations: Link alerts to relevant and helpful instructions, images, graphics, data, notes, or wiki-based runbooks to help responders have everything they need to quickly investigate and resolve the incident.

4. Routing: Set up unique escalation policies in line with team needs and fine-tune. Kick off escalations based on rules you choose and escalate to a secondary policy within your team, different team, or individuals.

To learn more about how you can build a culture of reliability in your organization, without hiring a full-time SRE staff, check out this *eBook*.

* "How to Measure Release-related Downtime," Plutora

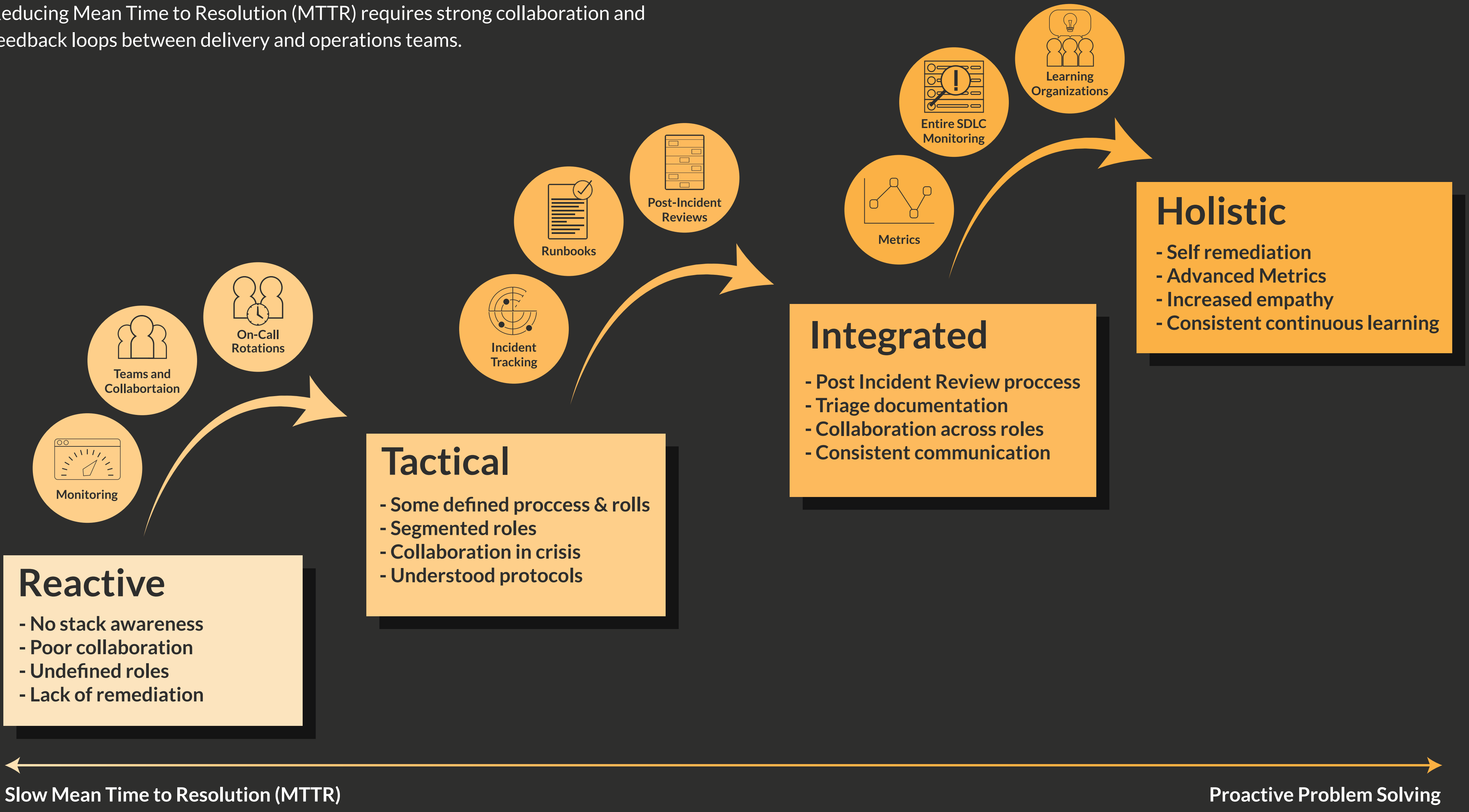
Incident Management Maturity

Beyond the stages of an incident, from readiness to resolution,there is a continuum of maturity for organizations and their overall approach to incident management.

Reducing Mean Time to Resolution (MTTR) requires strong collaboration and feedback loops between delivery and operations teams.

This culture of learning is fundamental to modern incident management and excellent DevOps practices.

Check out the chart below to do a quick assessment on the current stage of your incident management maturity. The pop-out arrows should cue you in on key tactics to guide your team from one stage to the next.



Reactive

- No stack awareness
- Poor collaboration
- Undefined roles
- Lack of remediation

Tactical

- Some defined proccess & rolls
- Segmented roles
- Collaboration in crisis
- Understood protocols

Integrated

- Post Incident Review proccess
- Triage documentation
- Collaboration across roles
- Consistent communication

Holistic

- Self remediation
- Advanced Metrics
- Increased empathy
- Consistent continuous learning

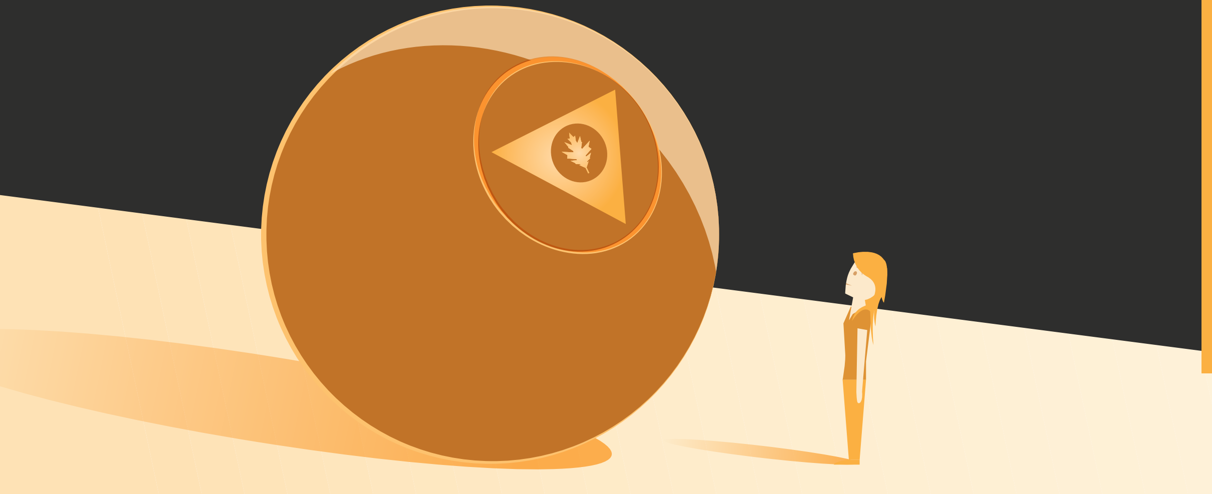
Questions to Ask Before Purchasing a Solution

Here's the thing: The majority of incident management tools on the market address the basics of "alerting." These basic feature sets, i.e., enriched alerting, on-call scheduling, broad integrations, and varied notification methods are all standard features.

During the evaluation process, buyer's should think about the next level of feature sets aside from basic functionality—essentially, you want to invest in a platform that continues to advance beyond alerting, building features that support a culture of high availability (reduced alert noise, improved uptime and SLAs, a culture of near-zero downtime) as well as DevOps standardization.

Perhaps most importantly, you want to look for software that treats you like a human being, i.e., being on-call shouldn't crush your soul. In today's connected workplace, most people don't work 9-5 anymore. For employees that have to answer an on-call page in the middle of the night, you'd like to know the software (and the people behind the software) have your backs. Take a look at the support team for the incident management solution you're evaluating and determine if they have a progessive, user-first mindset. Do they build features for the user or the CEO? Do they care about your experience waking up to an outage at 2:00 AM?

Your software needs to do more than check the boxes, it should make your on-call life not suck while simultaneously growing and scaling along with the organization.



These are the most important questions to ask of your solution:

Questions for On-Call Management

1. Will I find contextual alerts with abundant information for resolution?
2. Does the tool have built-in automation to reduce noise and alert responders only during critical incidents?
3. Does the tool support collaboration with bidirectional group chat integrations?
4. I have an international team. Does this software support international notifications?
5. Does this tool support/integrate with my existing critical toolchain components?
6. Can I access a variety of reports, including MTTA/MTTR and overall incident frequency?
7. Is there a native mobile app that supports on-the-go on-call?
8. How easy is it to conduct a thorough post-incident review? How hard is it to access historical data?
9. How can I configure alerts?
10. Are there varied levels of user permissions?
11. Bonus: Do I have SDLC visibility to see when things are shipped to production?

Questions for DevOps Teams

1. How likely is it that my development team would use this tool?
2. Would they find value in alerting? Or, would they simply be inundated with noisy alerts that make on-call miserable?
3. Does this tool prepare me for continuous learning or continuous improvement?
4. Can I access out-of-the-box performance metrics to report on SLAs and uptime?
5. How easy is it to conduct a thorough post-incident review?
6. Does this tool surface when new code is pushed into production?
7. Is this tool built for DevOps standardization? Or would we need to migrate to a new tool as our team progresses?

Migration & Success

Want to make a switch? Migrating to a new tool doesn't have to be a pain—at least, when it comes to VictorOps. The VictorOps support and success teams guides teams, enterprise and SMB, through the migration process to ensure everything goes smoothly. Teams will have all comparable basics within minutes—plus the features and functionalities to minimize downtime without slowing down delivery speed. Our dedicated customer success team will continue to support you throughout the VictorOps journey.

Build vs. Buy

Do you want your expensive developers spending time building, maintaining, supporting, troubleshooting, and updating an in-house solution? Consider the total cost of ownership, for which development itself is often just a small fraction. Wouldn't you rather developers focus on customer-facing work?

5 reasons homegrown solutions aren't recommended:

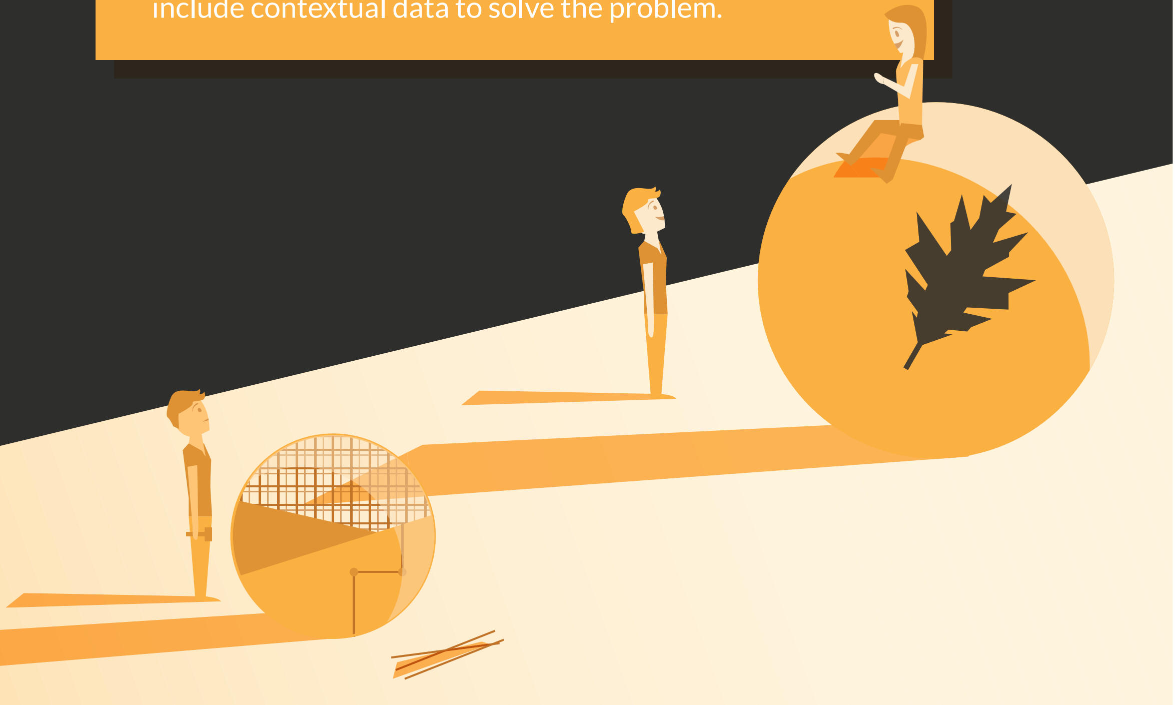
Lack of visibility—Many in-house solutions rely on email, SMS, and a collection of cobbled-together tools as a means of instilling urgency and showing transparency across all team members. The signal gets lost in the noise.

Lack of accountability and ownership—Email responses often become overlapped and communication splinters across other tools like HipChat, Skype, or Slack. The result is slower time-to-resolution and a massive cleanup exercise for postmortem analysis.

Inflexibility to schedule changes—On-call schedules are always created with the best intentions, but homegrown solutions often can't handle the inevitable—a child's first violin concert or an illness. Homegrown solutions often leave last-minute schedule changes in the hands of individual team members.

Lack of contextual documentation—56% of on-call professionals say that their biggest challenge is sharing contextual information (Source: State of On Call Report 2014). The top reported issue for problem remediation was effective surfacing of correct internal wikis.

Alert fatigue—Homegrown and alert-only solutions are known for multiple factors that contribute to alert fatigue: sending multiple messages without appropriate escalation paths, sending alerts to multiple individuals, and failing to include contextual data to solve the problem.



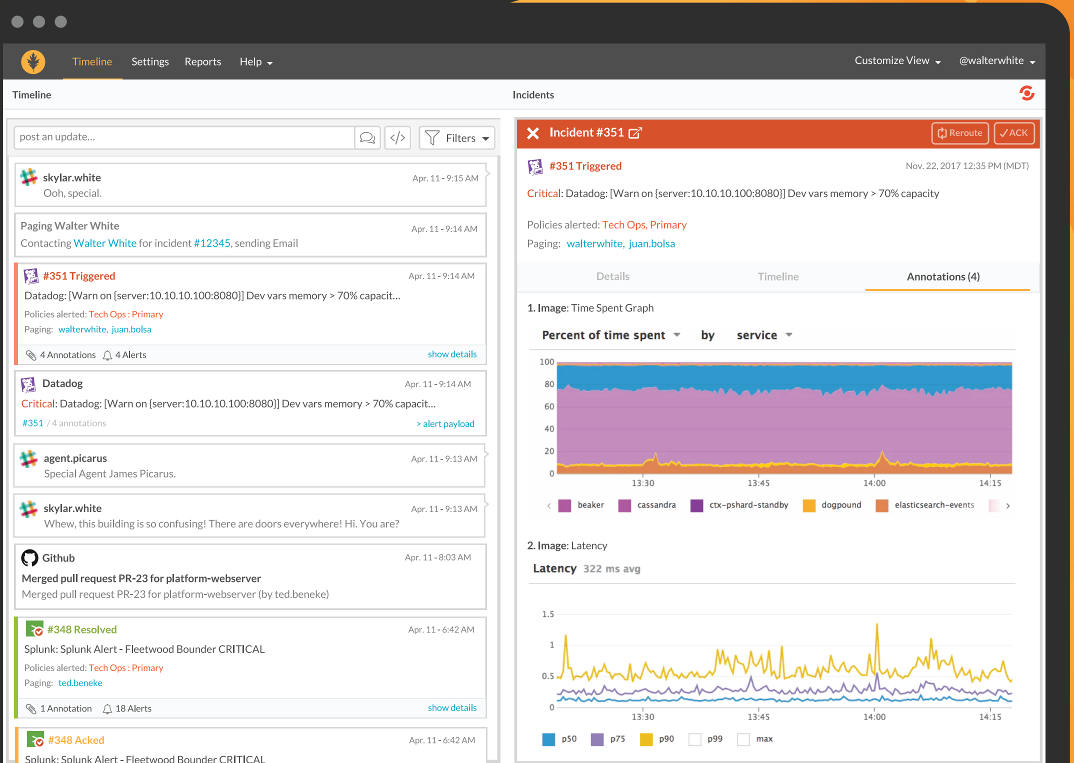
Why VictorOps

VictorOps supports a full-stack approach to incident management. Unlike our competitors, our system leans into the progressive vision of DevOps—providing broad visibility, from deployments to production, to even the noisiest systems.

We centralize user activity for next-level event transparency, so your team can lean into the speed of DevOps.

Ready to see VictorOps end-to-end incident management in action? Sign up for a personalized walkthrough with one of our product experts.

Get A Demo



skylar.white Apr. 11 - 9:15 AM
Ooh, special.

Paging Walter White Apr. 11 - 9:14 AM
Contacting Walter White for incident #12345, sending Email

#351 Triggered Apr. 11 - 9:14 AM
Datadog: [Warn on {server:10.10.10.100:8080}] Dev vars memory > 70% capacit...
Policies alerted: Tech Ops : Primary
Paging: walterwhite, juan.bolsa
4 Annotations 4 Alerts show details

Datadog Apr. 11 - 9:14 AM
Critical: Datadog: [Warn on {server:10.10.10.100:8080}] Dev vars memory > 70% capacit...
#351 / 4 annotations > alert payload

agent.picarus Apr. 11 - 9:13 AM
Special Agent James Picarus.

skylar.white Apr. 11 - 9:13 AM
Whew, this building is so confusing! There are doors everywhere! Hi. You are?

Github Apr. 11 - 8:03 AM
Merged pull request PR-23 for platform-webserver
Merged pull request PR-23 for platform-webserver (by ted.beneke)

#348 Resolved Apr. 11 - 6:42 AM
Splunk: Splunk Alert - Fleetwood Bounder CRITICAL
Policies alerted: Tech Ops : Primary
Paging: ted.beneke
1 Annotation 18 Alerts show details

#348 Acked Apr. 11 - 6:42 AM
Splunk: Splunk Alert - Fleetwood Bounder CRITICAL

Incident #351

#351 Triggered Nov. 22, 2017 12:35 PM (MDT)
Critical: Datadog: [Warn on {server:10.10.10.100:8080}] Dev vars memory > 70% capacity

Policies alerted: Tech Ops, Primary
Paging: walterwhite, juan.bolsa

1. Image: Time Spent Graph

2. Image: Latency