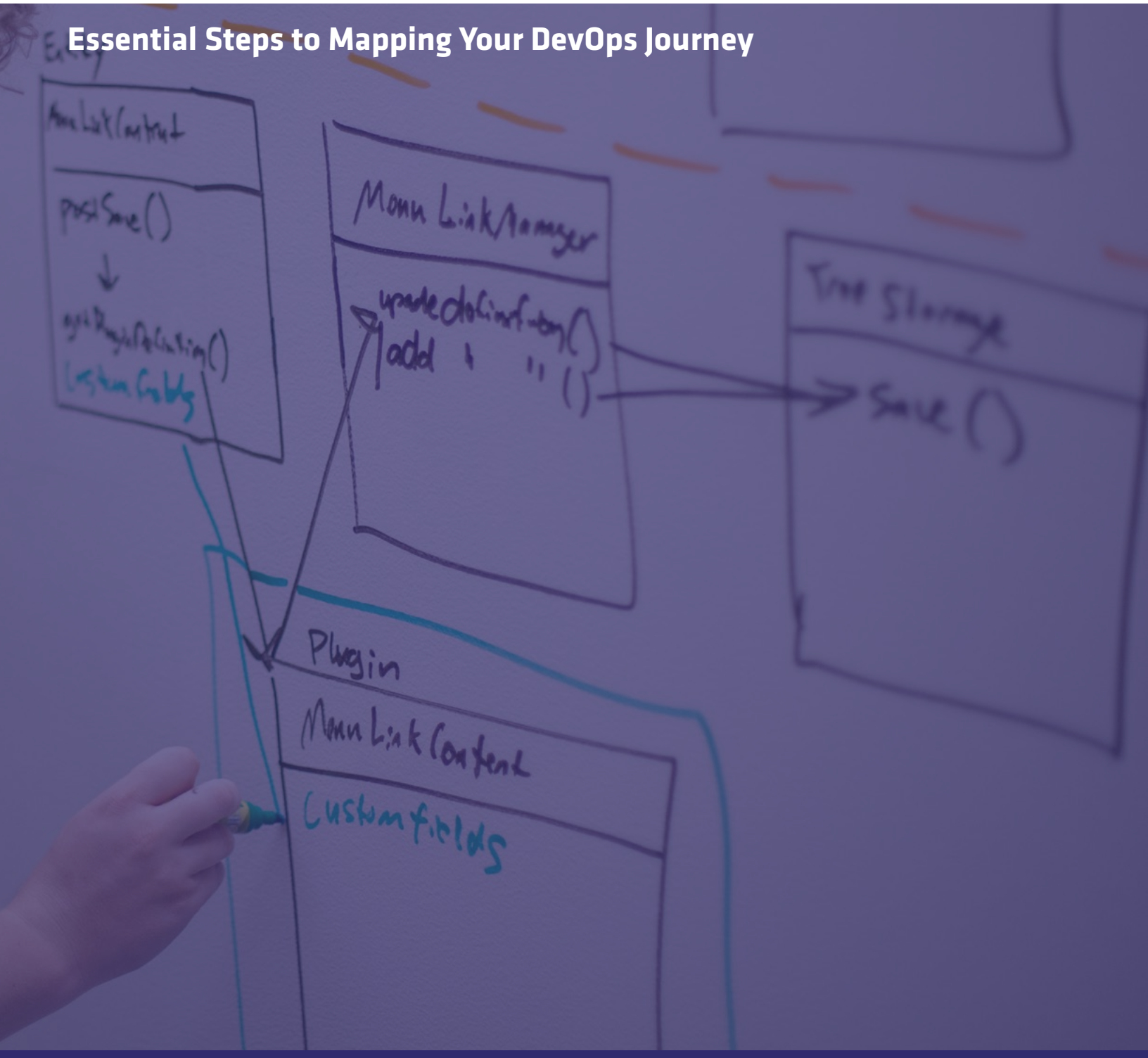
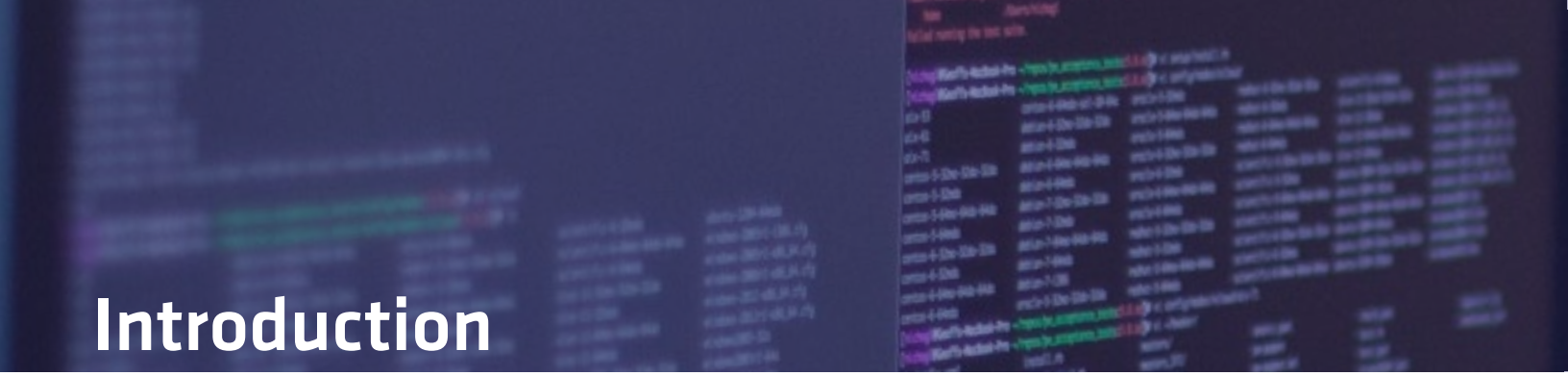
How to Build a High-Performing IT Team

Essential Steps to Mapping Your DevOps Journey



puppet labs

Table of Contents



Introduction

DevOps cuts deep and wide through industries, company sizes, and technology environments. Nevertheless, there's a common refrain from IT managers who are leading successful DevOps transformations within their organizations: DevOps is about continual learning and improvement rather than an end state.

Your specific process will ultimately take its own shape in your organization as you learn from successes and failures along the way. Still, it's nice to be able to lean on outside wisdom and support from people and organizations that have been there before.

We're here to help: This handbook captures five essential phases for mapping out a DevOps journey. We draw on the success stories and lessons learned at organizations that are already driving improved performance and better business outcomes with their DevOps initiatives. We also draw from the key findings and research you'll find in our *2015 State of DevOps Report.*

Build the Business Case

Like many of today's IT leaders, you're asked to not only deliver more products and services than ever before, but to do so with greater speed and quality — and with no hits to reliability or security.

DevOps seems like it will help, but you're running into skepticism from your team before you've even really begun. You might even be a bit doubtful yourself: DevOps will require considerable change. How do you make a clear, convincing case for DevOps that reduces fear and converts skeptics into champions?

Asking that question is actually a great start. Building the business case is a crucial part of a successful DevOps transformation, especially in large organizations that have long relied upon rigid organizational structures, processes and technical practices. It will help you win support for DevOps while reducing the noise and fear that's inherent in this kind of transformation. Let's look at three pillars of a solid DevOps case.

1. Establish *why* you're doing DevOps with data.

In a famous *TED Talk,* Simon Sinek notes a common denominator of great leaders and catalysts for positive change: People don't buy what those leaders are doing but why they're doing it. The same idea holds true when building organizational buy-in for a DevOps transformation: Simply declaring "We're doing DevOps" is not going to get people on board. Instead, you need a compelling answer to the questions, "Why? And why now?"

All our customers want to move faster without compromising the reliability and stability of their systems — goals that directly conflict with each other in traditional organizations. Devs are tasked with getting new features into production as fast as possible; Ops, meanwhile, is measured on the uptime and performance of systems. So these teams become adversaries instead of allies. As a result, deployments to production are plagued by delays and errors, and they occur far less frequently than the business really needs.

This year's DevOps report again confirms that DevOps practices lead to better IT and organizational performance. High-performing IT departments achieve superior speed and reliability relative to lower-performing peers. The 2015 survey showed that **high-performing teams deploy code 30 times more often and with 200 times shorter lead times** than their peers. And they achieve this velocity and frequency without compromising reliability — in fact, they improve it. High-performing teams experience **60 times fewer failures**.

When incidents do occur, high-performing teams **restore service 168 times faster** than their peers. So DevOps practices really do make a positive difference to the business.

2. Emphasize benefits for different roles and teams.

The big picture case is great, but people want to know how DevOps impacts them at an individual contributor level, too. Show them how it will improve their jobs and address the things they care about most. As your DevOps journey gets underway, underscore these benefits with data demonstrating early wins.

Getting Dev on Board with DevOps

Faster deployments and feedback loops get to the heart of what devs want: Code gets from their laptops into users' hands much faster, and *continuous delivery* enables rapid iteration and improvement.

Tracking improvements to your **change lead time** during early pilots is a good place to begin: How quickly can code move from a dev's laptop to production? How does this stack up against your prior lead times? Tell a robust story that explains how you reduced that gap between commit and deploy: Did you automate more of the build process? Did you cut down the number of tickets needed to deploy?

As you get farther along on your DevOps journey, measure **deployment frequency** and compare it to the "before" picture. How often are you deploying now versus then? Just as important, have deployments becomes easier and faster?

Getting Ops on Board with DevOps

Ops benefits when devs work closely with them. It can be helpful to start by agreeing on a common toolchain and having the two groups work together to adopt the same tools used in development for integrating, testing and deploying infrastructure code. This brings devs more actively into deployments and troubleshooting, further breaking down old barriers while improving speed and reliability.

Tracking several metrics that Ops cares about will underscore the wins for the entire team — including Dev and QA:

- **Uptime/downtime:** Are you better able to meet your service-level requirements? Has downtime decreased?
- **Change fail rate:** Have failures decreased?
- **Mean time to recover:** Have you shrunk the time it takes to roll back to your last-known good state when a failure occurs?

How Everyone Wins

As you build success stories, it's important to connect the dots that show everyone how

they personally benefit from DevOps practices. QA will love that quality "moves to the left" in the software development process as a result of improvements to both speed and stability. Our 2015 State of DevOps survey found that's precisely what's happening in high-performing IT orgs: Quality checks and improvements are being built earlier and earlier into software development.

This "shift left" in the process benefits the entire organization. Errors are caught earlier, when the cost of fixing them is much lower. People gain confidence in the software delivery process itself and feel greater investment in improving it further. Adopting the practices of

lean management and continuous delivery can, in fact, improve not only the speed and reliability of software delivery, but the organizational culture as well. This in turn helps reduce burnout, increase retention and gets employees more engaged in innovation and improvement.

3. Start small and grow from early successes.

So how do you begin to measure these DevOps impacts and bolster your business case? Start small with specific tasks and projects.

This approach proved *highly effective* for Terri Potts, engineering fellow and software technical director at Raytheon. The defense contractor is a massive organization that has been around for decades, in an industry that is inherently averse to change and risk.

"You can't change the whole program, but you can start by getting a few of your sub teams going in the right direction," Terri says. "It can be helpful to bring in someone from the outside to automate a few tests or builds and give the team some examples to build on." Raytheon enabled one of its teams to shift from two integration procedures per month to running 27 of them in a single night as a result of automating its builds. That's a big win on a single initiative, and it became part of Potts' broader case for growing DevOps within the organization.

Start small with the cultural shift, too — don't expect to sell DevOps to everyone at once. In fact, by winning over smaller groups with specific projects, you'll create ambassadors who can help promote DevOps elsewhere in the organization, creating a multiplier effect.

As you build your business case and bolster it with early wins backed by data, you should also remain mindful of potential obstacles to long-term DevOps success. In the next section, we'll identify some of the most common challenges — and strategies for mitigating them.

Many organizations first explore DevOps driven by a common need: to deliver faster without sacrificing reliability. Our annual State of DevOps report has consistently debunked the myth that speed and stability are mutually exclusive. Our 2015 report underscored this yet again:

High-performing IT organizations experience **60 times fewer failures and recover from failure 168 times faster** than their lower-performing peers. They also **deploy 30 times more frequently with 200 times shorter lead times.**

Address the Biggest Challenges to Success

The team at Salesforce *once said*: "We're never going to get to the point where we're DevOps now." Rather, it's a continual process of learning and improvement. There will be challenges along the way. By identifying and anticipating them, however, you can plan how best to tackle those challenges in your organization so they don't waylay your DevOps initiative. Let's look at three common challenges we regularly see with our customers.

1. Conflicting incentives and measurement across functional teams

One of the biggest challenges to DevOps success is that teams aren't aligned around the same goals and incentives. Dev teams are rewarded for delivering new features fast; Ops is rewarded for ensuring stability of systems.

The battle between speed and stability seems irreconcilable, but we know that these two seemingly opposing forces can coexist: High-performing IT organizations are able to increase both throughput and stability. Our research has found that they deploy code 30 times more frequently with 60 times fewer failures. They also recover from failures 168 times faster.

Although ensuring code quality is technically the quality team's job, that doesn't mean they're the only ones responsible for testing code. In a high-performance organization, quality is built into the entire delivery system. Dev teams unit test their code and work with the quality team to write acceptance tests. Ops teams test infrastructure changes before deploying into production. These practices, and others like them, ensure better quality code and less chance of failure.

These are the practices of high-performing IT teams. They're all rooted in common ground: Resolving conflicting incentives to promote success for the entire team, rather than creating conditions that force different groups to compete with each other for control and resources.

In *Building a DevOps Culture,* Mandi Walls defines key pillars of organizational culture, including open communication, trust and respect. These are all good things, but a fourth pillar stands out as more immediately actionable: alignment of incentives and responsibilities. Walls notes the importance of empowering individuals and teams to take ownership of quality and customer value without worrying about how their work is measured.

Support from upper management is crucial here, because some failures are inevitable. People need to know they can learn from those early lessons without looking over their shoulder.

"Development isn't rewarded for writing lots of code. Operations isn't punished when all that code doesn't run as expected in production," Walls writes. "The team is rewarded when the product is awesome, and shares in the improvement process when the product could be more awesome."

2. Cultural stasis and mistrust

Aligning goals and incentives is an absolute must for tackling the next considerable challenge: Cultural *change*. This has been the biggest ongoing obstacle at defense contractor Raytheon, according to Terri Potts, engineering fellow and software technical director at the Fortune 500 firm. "The No. 1 hurdle is, 'That's not how we've always done it,'" *she said in an interview*. "It's hard to change the culture."

A significant transformation like DevOps can produce noticeable fear, especially in the absence of reassuring data. If a dev thinks DevOps means her phone will start ringing nonstop on nights and weekends, can you blame her for a being a bit skeptical? And if a sysadmin believes automation will eliminate his job, then fear is certainly a natural response.

Change isn't just hard for traditional enterprises. It's a challenge for a cloud leader like Salesforce, too. Unlike a bug in your code, there's no set test-troubleshoot-test methodology for cultural resistance. As a result, the team at Salesforce preaches patience on their *DevOps journey*: "You need to continuously socialize what you want out of DevOps," says Reena Mathew, principal architect. "The key thing is change has to happen across the board. It takes a while, but you need to continuously work on it."

Communicating and celebrating DevOps successes along the way is a crucial strategy for reducing fear and continuously building the business case, one both Potts and Mathew embrace in their organizations.

3. Outdated tools, legacy systems and entrenched practices

While it's true that legacy tools and systems may pose considerable challenges as you move forward, many IT teams in the most risk-averse in the most risk-averse industries turn to DevOps because they realize it's more expensive to maintain the status quo than to adopt new technologies and practices.

The good news is that tools, unlike the broader corporate culture, fall more directly under Ops' control. You don't have to overhaul everything at once. It's okay to start with one small thing, or at least one distinct thing. You might start, for example, by simply putting

all of your configuration scripts in version control, to ensure you always have a record so you can see which configurations were in force at any given point in the delivery cycle.

Or maybe your goal is to automate your deployments, so you start by standardizing your OS configurations. SaaS company Fulcrum Technologies was managing its 100-plus servers entirely with one-off scripts, which meant it had to provision two new nodes every time the company won a new customer. Doing this repeatable, routine task manually ate up much of the IT team's time. Automating the provisioning and configuration of new systems with Puppet freed up network and system administrator Bill Wellington to redirect his time toward more complex infrastructure issues and business problems. He was also able to spend more time helping his team invest in skills development, a boon for the individual team members and the organization.

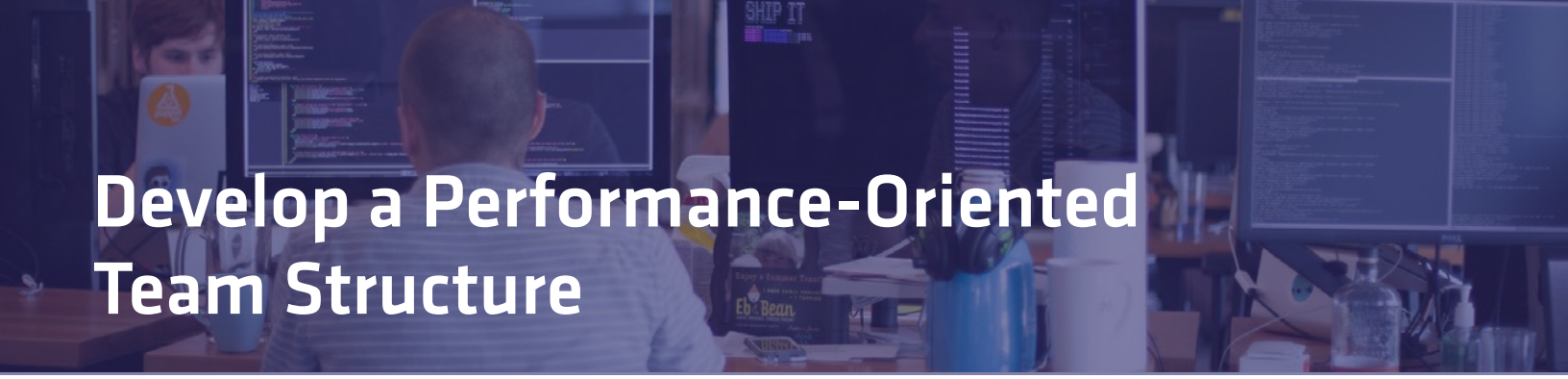
Managing infrastructure as code enabled Jez Miller, director of operations at Heartland Payment Systems, and his team, to scale the firm's virtualized infrastructure in *minutes instead of months,* while cutting deployment times from 10 hours to around 10 minutes.

Heartland's experience is a good reminder of why it's worth tackling challenges like these in the first place, even when it's not easy: The potential rewards are great for everyone. As you start to realize these rewards, fear shrinks and people see new opportunities appear.

"You stop doing those manual tasks at work that aren't fun, that aren't developing your skills, that aren't keeping you engaged as an employee. Now you are doing really cool stuff, solving a whole different set of problems in a whole new way," Miller says. "You start stacking up those resume lines, because you aren't just doing system patching for a living. You can solve much bigger problems."

To get to where they are today, Jez Miller and Heartland needed to address incentives that put teams in conflict rather than in cohesion. In the next section, we'll learn about the importance of team structures — and how organizations like Heartland have fostered real collaboration and empathy between functional units.

A leading cause of DevOps skepticism is the belief that brownfield environments aren't suitable for best practices such as continuous delivery. Our 2015 State of DevOps Report found this isn't the case: Continuous delivery can be applied to any system provided it's architected with testability and deployability as priorities. Moreover, there's a clear link between continuous delivery practices and higher IT performance, lower change failure rates, and decreased deployment pain.



Develop a Performance-Oriented Team Structure

We often get asked what the ideal organizational structure is for implementing DevOps practices. The answer to this question depends on a range of variables, such as the flexibility of your organization's current structure, available skills and resources, relationships between teams and team leaders, and corporate culture and politics. In this post, we'll examine the three patterns we see most commonly.

Conflicting incentive structures created by misaligned goals and siloed teams do not produce the best outcomes. At *Heartland Payment Systems,* for example, these misaligned incentives once turned software releases into painful, months-long cycles. "Ops was meant to keep things secure and stable, but Dev wanted to move very quickly," says Jez Miller, Heartland's director of operations. That's why Heartland and many other companies have realigned their teams around common goals, processes and tools as a necessary step in their successful DevOps transformations.

Type 1: Close-Knit Collaboration Between Dev & Ops

This structure takes once-siloed development and operations functions and redeploys them as tightly woven, highly collaborative teams working side by side. This often involves considerable change from the previous way of doing things. Consider this advice from Sam Eaton, director of engineering operations at Yelp, on how to build trust and convince people that this is a positive change : Begin with a willingness to experience someone else's pain points first hand. "It helps if you've done what they do, so you should take the on-call rotation, take the alerts, and walk a mile in the other person's shoes before you speak up about what needs to be changed," says Sam.

For Jez's team at Heartland, aligning development and operations on a common toolchain boosted collaboration while dramatically speeding up deployments and reducing errors. In fact, the company no longer tracks downtime because it occurs so rarely.

"Conversations with development went from, 'oh that's tough, we don't know that, we aren't experts'...to 'we can change that easily because we know we can run those changes through QA, we can validate them, and treat the infrastructure as a code base as well,'" Jez says. "And that fundamentally changed what we were able to do as a company, and the speed with which we were able to deliver."

This approach also maximizes the blend of software engineering skills and deep systems knowledge most organizations need. That combination can be tough to find in a single person, which poses problems for the "NoOps" approach. "NoOps" is typically seen in startups or smaller companies where the devs have enough systems and infrastructure knowledge to get the app up and running, but lack the deep knowledge required to scale or to succeed in a large enterprise setting. Similarly, the completely embedded ops team, in which systems people can code and devs have extensive operational expertise, is relatively rare, and often only found in technically mature, highly innovative web companies like Netflix or Facebook.

Other organizations, then, are better served building upon the individual skills and expertise they already have in house. Make sure dev and ops teams are collaborating and sharing responsibilities throughout the software delivery lifecycle — from initial planning all the way to managing the production environment. Many organizations employ a "You build it, you run it approach" meaning that developers are responsible for deploying their applications — and being on-call if anything goes awry. This builds empathy and weans everyone off the habit of thinking: "That's someone else's job."

CenturyLink's vice president of product, Richard Seroter, writes *on his blog* about the tight partnership between development and operations in his own cloud organization: "There's no 'us versus them' tolerated, and issues that come up between the teams — and of course they do — are resolved quickly and decisively."

Type 2: Dedicated DevOps Team

Dedicated DevOps teams are often made up of experienced operations people with a mix of skills and experience with the tools and processes of continuous delivery, including version control, managing infrastructure as code, and working closely with dev teams . These purpose-built DevOps teams typically start by addressing the things that are most painful, such as deployment automation, and if they're successful, can evolve to providing shared services for the rest of the organization.

As *Matthew Skelton* notes in his great *blog post* on team typologies, however, you do need to watch out for a pitfall of the dedicated team: Even if it's temporary, it can become just as much of a silo as the functional teams that preceded it.

Despite that risk, there's evidence for the rapid growth of dedicated DevOps teams. In our *2015 State of DevOps Report*, 19 percent of respondents identified themselves as working specifically in a DevOps department, a noticeable rise from 16 percent in our 2014 survey.

Type 3: Interdisciplinary teams

Interdisciplinary team consist of representatives from all functions responsible for developing and deploying a service (business analysts, developers, quality engineers, ops, security, etc.). These teams are fully empowered and self-sufficient — they write it, test it, and deploy it.

Gareth Rushgrove, senior software engineer here at Puppet Labs, helped build that kind of team inside the Government Digital Service in the UK. Here's what Gareth had to say about it: "Rather than a traditional [structure] — the operations people over there, the developers are over there, designers are over there, everyone in separate teams by discipline — our structure was very much teams around a product, around a thing."

In some cases, these teams are temporary, in the sense that they're created by drawing people with the right mix of skills and experience from different areas of the company — and external resources as needed — to work on fast-paced, time-boxed projects.

GE Capital's *DevOps journey*, for instance, has relied on interdisciplinary teams pulled together for *six-month projects*, with shorter-duration sprints baked into that timeframe. There are a variety of names for these kinds of teams: "agile" appears quite a bit; *"tiger team"* is another.

The combination of being self-sufficient and also having shared incentives is powerful: Cross-functional teams work more efficiently, with fewer hand-offs to other teams and more opportunities for knowledge sharing.

There is No "I" in Team, or Is There?

DevOps is definitely a team sport, but it's still important to clearly define individual roles and responsibilities for success within the team. Here are some recommendations:

- **IT manager:** Build trust with counterparts on other teams; create a climate of learning and continuous improvement; delegate authority to team members.
- **Dev manager:** Build trust with Ops counterpart; create a climate of learning and continuous improvement; bring Ops into the planning process early.
- **Systems engineer:** Automate first the things that are most painful for Ops.
- **Quality engineer:** Provide input into scale and performance; provide feedback on staging environments.
- **Devs:** Plan for deployment as you're planning new features; get feedback from Ops and work with them on deployment process.

Regardless of your organizational structure, DevOps requires collaboration across multiple functions and shared responsibility to ensure the best possible quality.

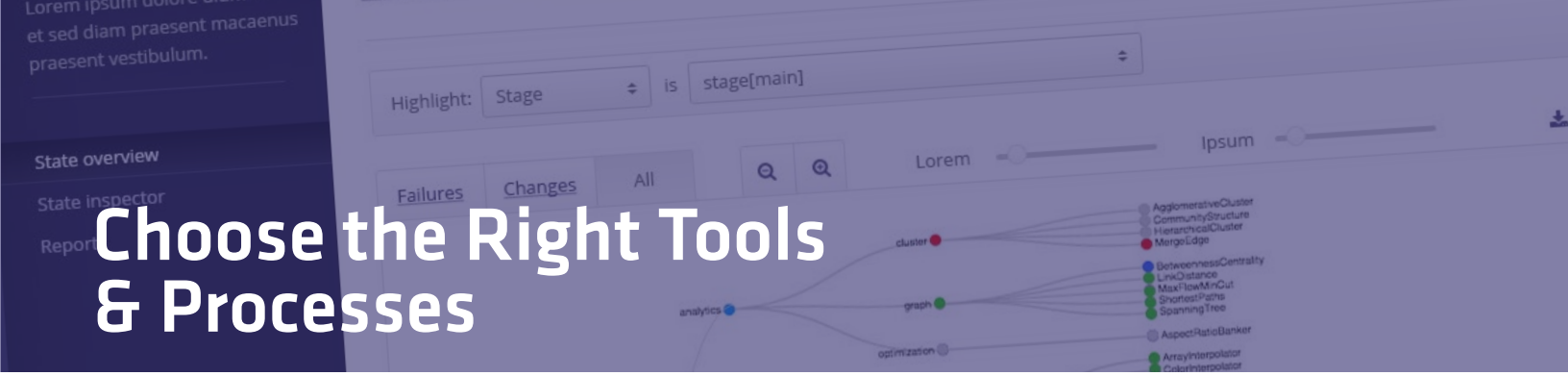
And the rewards can be tremendous, as Jez at Heartland Payment Systems discovered: "Stop the firefighting, keep consistent platforms, let people sleep at night, give people their lives back, and really make their development lives and their operational lives easier, and make them successful within their businesses."

It's clear that DevOps success relies heavily on people and culture. The right tools and processes are also essential, and can help drive the necessary organizational changes in the early stages of your DevOps journey.

Disney's DevOps transformation included an apparently small change: the Ops group changed its name from "Web Operations" to "Systems Engineering." According to Jason Cox, the team's director, it was much more than a superficial org chart change:

"The software engineers now saw us as kindred engineers," Jason said. "We got more collaboration, and more sense of what the software engineers did. We saw more of their space and their pains, and what it takes to write apps. We were able to tell them about resiliency, and what it takes to run apps."

The name change signified a deep cultural change, and the new-found empathy between people in different technical roles engendered the next big leap in Disney's DevOps journey: embedding the systems engineers in cross-functional, product-driven teams. That led to success stories for many of those teams, such as reducing the time needed to stand up new instances from days to five to 10 minutes, and significant decreases in manual steps and human-caused errors.



Choose the Right Tools & Processes

Contrary to what some vendors might tell you, there's no single, one-size-fits-all DevOps tool. Rather, the most effective results come from standardizing on a toolchain that maps directly to best practices such as version control, peer review and continuous delivery — all built on a foundation of managing *infrastructure as code.*

The right tools are also a huge help in establishing the early successes that convince people to embrace change rather than fear it. In this way, proper tooling becomes a catalyst for the grassroots transformation — backed by strong leadership — that we see behind many DevOps success stories.

One IT services firm, for instance, spent countless hours using manual processes and one-off scripts to manage its Linux and Windows environments. After the Linux admins began automating much of their routine updating and patching work with Puppet, the Windows team took notice and followed suit. The company has subsequently saved between 60 and 80 percent on the cost of routine Windows management. Plus all admins, both Windows and Linux, now work regular office hours, with rare exceptions.

Automating painful, manual processes is just a first step — though a significant one — in establishing the DevOps toolchain. We see five key categories for tools that drive best DevOps practices.

5 Pillars of the DevOps Toolchain

- **Version control** (GitHub, Mercurial, Perforce, Subversion, Team Foundation Server)
- **Configuration management** (Puppet and others)
- **Continuous integration** (Atlassian Bamboo, Go, Jenkins, TeamCity, Travis CI)
- **Deployment** (Capistrano, MCollective [part of Puppet Enterprise])
- **Monitoring** (New Relic, Nagios, Splunk, AppDynamics, Loggly, Elastic)

Make sure you adopt and standardize on tools that are suitable for your organization in each of these categories, and take into consideration how each tool you select fits in with the rest of the toolchain. It's also wise, of course, to make sure someone owns the issue of overall tool compatibility — and is empowered to drive decisions around tooling.

Managing Cultural Resistance to New Tools

DevOps success grows from everyone on the team working with the same tools and processes. But we all tend to get a little dogmatic about the tools we use, so standardizing on a toolchain requires much more than simply declaring: "Here's a new tool; now we're using it!"

Devs should take an active role in collaborating with their Ops colleagues on how to use the tools they employ for software development. You can schedule lunch-and-learns and encourage pairing to help admins understand how these tools can benefit both the individual and the overall organization. Similarly, Ops can standardize on critical policies (base OS config, NTP, firewall) and provide self-service options, or a pool of pre-configured resources. QA architects and the testing team should likewise be brought into the early tooling discussions, as writing and configuring tests in a standard way is important to every stage of the development and release cycle.

Some resistance is normal, so focus on how each tool will help people do their jobs better while addressing issues they care about.

At Hiscox, that resistance led to a valuable lesson during the early stages of its DevOps implementation: You can't just tell people what the problem is and what the solution is. Instead, Hiscox's technology and platform lead Jonathan Fletcher found that creating empathy between functional teams, bolstered by the rights tools and processes, was critical for success. Fletcher details one of the *key DevOps learnings at Hiscox:*

"Work with people, have them help define the problem statement and then lead them to the same conclusion. That helps overcome the resistance people feel and helps accelerate the change curve. People feel bought in to the change instead of feeling, 'Who's this telling me how to do my job?'"

How Tools Drive Processes That Fuel Performance

In the early stages of a DevOps transformation, we see the most success when starting with version control and configuration management tools. One reason is because they naturally connect to some of key processes and metrics that correlate with higher IT performance. These tools will help you attain the early successes that bolster your DevOps case. If you're frustrated by a high number of errors with each deployment to production, for example, the use of a standard version control tool by Dev, Ops, and QA can slash error rates while reducing mean time to recover when issues do occur.

The same applies to any pain points or repetitive processes you can automate. *Fulcrum Technologies* has reduced both the amount of time it spends building systems and the amount of time it takes to find and implement a fix when production issues occur.

"In the past, we experienced issues because a patch or multi-step change was not applied uniformly," says Bill Wellington, who's a network and system administrator at Fulcrum Technologies, and in charge of DevOps. "If I have a consistent state — even one that is broken — then coming up with a fix is a much easier undertaking."

As you build your toolchain, it's important to understand how each tool amplifies the benefits of the others. We're big believers in managing infrastructure as code, and once your infrastructure code is in a version control system, you're able to apply the best practices of agile development. These practices include peer code review, continuous integration, automated testing, and continuous delivery, all of which help you increase deployment frequency, while simultaneously increasing the quality of your code. And version control not only enables faster deployment with fewer failures, it also lets you roll back easily to the last good known state when failures do occur, reducing downtime for your services.

Peer code review is a great example of how increased collaboration on a common toolchain improves results. It sounds a little scary if you're used to relying on traditional change approval processes. But our research has shown that peer code review correlates with higher IT performance, while there's no clear link between external change approval processes (such as a change approval board) and system stability.

Similarly, *continuous delivery* exemplifies the symbiotic relationship between the right tools and technical best practices. It allows software teams to deploy much faster and more frequently by enabling smaller changes delivered with greater frequency. This in turn promotes faster feedback loops, and the ability to make incremental improvements more quickly and frequently.

With automated testing, you help transform quality assurance from a pre-production bottleneck to something infused into the earliest stages of software development. Jonathan at Hiscox has seen the results of this change first hand:

"We now have testers working with developers and business analysts from the early requirements stages. There has been a big change, from testing being something we throw offshore to testing being a first-class citizen," he says. "The test code is as important as the application code. Great testing sets your development free."

It also saved the Hiscox team time and money while ensuring quality: Automated testing has reduced multiple days of effort to an overnight, hands-free process, and the company lowered the cost of deploying one of its most important applications by 97 percent.

The right toolchain and processes don't just improve the work; they improve how you

work by increasing collaboration and feedback loops, and enabling rapid iteration and innovation, all while improving stability. Your work culture becomes one of constant learning and improvement, driving higher IT performance that produces visible business results.

Our *2015 State of DevOps Report* found that's precisely what's happening in many organizations: These technical best practices, enabled by modern tools, continue to correlate with greater IT performance, as measured by both speed and reliability.

In fact, this year we found that high-performing orgs continue to deploy 200 times faster and 30 times more frequently than lower-performing peers, while quality and stability actually increased: High-performing IT teams experience 60 times fewer failures and recover from failure 168 times faster than lower-performing organizations. As we wrote in our report: "High performers are able to achieve higher levels of both throughput and stability through the use of DevOps practices — a key reason the movement has gained so much traction."

We've learned about the changes to people, processes, and tools that DevOps typically requires. But what does the path to continual learning and improvement really look like? In the next section, we'll examine common phases of a DevOps journey en route to greater IT performance.

Hiscox's DevOps story highlights the fact developing the right teams, processes, and tools as part of a DevOps transformation isn't about change for change's sake: It's about achieving greater performance and business results. In Hiscox's case, those were dramatic.

The company slashed the cost to deploy a single application by 97 percent, a direct result of reducing the time to release by 89 percent and staff required to release by 75 percent. Jonathan Fletcher, enterprise architect and lead for technology, platform, and DevOps at Hiscox, asked an equally exciting question: "How much is un-siloed thinking actually worth?"

We've wondered about similar questions: How do DevOps practices impact things that are harder to quantify, like organizational culture? And how does culture impact overall performance? It turns out those impacts are tangible and significant. Our 2015 State of DevOps Report digs into why a generative culture, investment in DevOps, and IT performance are the top three predictors of overall organizational performance. *Read more.*

Plan Your Key Implementation Phases

There are many paths to DevOps success, and they all share something in common: The transformation of people, processes, and tools occurs in incremental phases over time. There's no end point for DevOps; it's about achieving a state of continual learning and improvement.

Many DevOps journeys begin because of a pending project that drives Ops to seek better processes and tools. This might be a greenfield application; a new platform as a service (PaaS) or infrastructure as a service (IaaS) offering for internal users; a continuous delivery initiative to improve application throughput and quality; or maybe even a regulatory compliance mandate.

If you don't have an immediate project to drive your DevOps initiative, consider this great advice from Kate Matsudaira of *Popforms:* Make one up. Give it a name, time-box it, and involve as many teams and levels of the organization as possible. Most importantly, share the results. By doing so, you create a sense of purpose and urgency, which is so critical to the success of any project. This will give you a strong foundation of early learning and success to continue building upon when those major time-sensitive projects do come in.

No matter the project, there are many incremental steps in any DevOps journey, and we can group them into three distinct implementation phases

Phase 1: Automate the Things That Are Painful

One of the questions we get asked frequently is "Where do we start?" Our answer is "Start small and automate the things that are causing the most pain." It sounds simple, but for most organizations that are caught in a vicious cycle of constant fire fighting, it's by no means easy. In the 2015 State of DevOps Report, we found a strong correlation between IT performance and deployment pain:

The more painful code deployments are, the poorer the IT performance, organizational performance and culture. The data also tells us that painful deployments result in higher change fail rates.

We advise IT managers to ask their teams two questions:

1. How painful are your deployments?
2. What is causing the most pain?

The answer to the first question will tell you how your team is performing. The answer to the second question will tell you what to prioritize. As we often say, automation transforms the vicious cycle of constant fire fighting into a virtuous cycle of continuous improvement and learning, which is the goal of DevOps in the first place.

Phase 2: Standardize Processes and Tools

Now you've had a taste of how automation can give you time back in your day to work on more strategic initiatives. The next step is to start standardizing processes and tools to reduce complexity. Many of our large enterprise customers have hundreds of applications, and all of them are running on different technology stacks. Some app dev teams may be doing continuous delivery while others are just getting started with continuous integration. At this point, it's useful to do an audit of all of these applications to understand what tools and processes are in place. We know of one company that did an audit like this and discovered they had 60 internal private clouds!

Consider the gains in productivity attained in manufacturing when the assembly line was introduced and ***parts became interchangeable.*** Throughput increased dramatically, the quality of the end product improved, and costs were reduced. You can think of the software delivery process as the assembly line, and components of the technology stack as its interchangeable parts.

We've found that there are two critical practices that significantly increase the reliability and stability of the system, while also improving throughput: The use of version control for all production artifacts and infrastructure as code. Standardizing the tools used for these practices across the organization further increases productivity and efficiency.

1. Version Control

If you have a lot of custom scripts, the simplest thing you can do is move them to a version control system. In the event that a server goes down and you don't have a backup, you'll still have all of your scripts safely stored in a single place. Better still, this gives you an on-the-job way to learn the tool (and later teach it to others) as you expand your use cases to include application and infrastructure code.

2. Infrastructure as Code

The next step is to eliminate the need for so many custom scripts and manual processes in the first place. In Chapter 2, we heard about how this labor-intensive effort was hampering Bill Wellington and the team and Fulcrum Technologies. Managing 100 servers

with unrepeatable scripts wasn't a scalable solution for the company's rapidly growing SaaS business. So Fulcrum automated its configuration management and did away with most of the manual processes and one-off scripts. Managing infrastructure with standardized, repeatable code then freed the team up for other projects to improve operations and move the business forward.

"I was able to concentrate on the problem of monitoring our complex infrastructure in the time I would have spent building machines," Bill says. He was also able to invest more time in ongoing skills development for his team members, which will continue paying dividends as Fulcrum grows.

This might seem overwhelming if you're heavily reliant on custom scripts and manual configurations today. Don't worry about trying to automate everything at once. Start with just one of your most painful, time-consuming tasks. Maybe that means standardizing the base operating configurations across your machines, a common early step we see our customers take. From there, you can expand and automate other *common configuration tasks*, such as NTP, DNS or SSH.

Automating your configuration management tasks and managing your infrastructure as code, along with using a standardized version control tool, can also simplify the compliance challenges many IT teams face, by creating a complete, accessible audit trail. In fact, easier, more efficient compliance reporting in itself often drives new DevOps initiatives.

Phase 3: Process Improvement with Focus on Internal Customers

When you've automated away pain points and standardized tools and processes, you can expand your DevOps implementation to areas like application deployment and systems monitoring. In doing so, you extend the speed and reliability gains you've achieved within the operations team to internal customers such as Dev and QA that rely upon you for infrastructure.

If your application deployments are currently all-hands-on-deck, war room affairs, this would be the point to consider a move toward continuous delivery in your organization. Continuous delivery is an enormous goal, and not every organization will achieve it quickly, but you can still make significant progress in areas like deployment frequency, deployment lead time, and change fail rate. Our annual State of DevOps Report has found continuous delivery practices consistently correlate to higher IT performance.
When Ops is able to be more strategic and responsive to internal customer needs — while improving reliability and reducing stress — you'll likely see a decrease in help-desk tickets and be on your way to dismantling silos between operations, network, and storage teams, among other benefits.

Phase 4: Organizational Realignment with a Focus on Continual Learning & Improvement

There may not be a single end point for DevOps, but there is an overarching goal: Become a learning organization that's always striving to improve and deliver more value to customers.

Salesforce is a great example: They're never "done" with their *DevOps journey*, but they have reached the point where everyone on the team works in alignment within that process of continual learning and improvement.

Reena Mathew, principal architect on Salesforce's Quality Engineering team, once described it like this: "We all benefit from working together more efficiently. Everyone is interested in making our services better, and everyone is thinking bigger scale." Better still, this benefits the team and the company today while preparing it for what's next around the corner, a must in a highly competitive industry: "We're encouraging people to ask the right questions to understand what we need to deliver for the future," Mathew says.

Salesforce didn't get to that point overnight. Rather, it's benefitting from the DevOps equivalent of compounding interest: The longer you do it, the greater the rewards. There's no "kick up your feet and relax" phase. Instead, the results of your hard work will grow and grow.

This is borne out by our annual State of DevOps survey data. Our research has shown a consistent correlation between DevOps practices and higher IT performance; it has also found that performance keeps improving with time. Our 2015 survey once again debunks the myth that speed and reliability are in conflict. Better yet, reliability continues to improve over time with DevOps practices. The report states: "The fact that stability increased in our high-performing group suggests that quality is shifting to the left — that is, it's being built into the software earlier in the development process."

In other words: **The best keep getting better.**

Appendix

Additional DevOps Resources

There are plenty of other resources available for anyone who wants to adopt DevOps practices in the workplace. Here are a few that we recommend.

Resources

2015 State of DevOps Report. Download the full report, with insights on IT performance, lean management and the role IT leaders play in a DevOps transformation.

Infrastructure as Code: Why it Matters. In short, infrastructure as code enables speed, innovation and stability.

The Phoenix Project by Gene Kim

Tools

Version control
GitHub, Mercurial, Perforce, Subversion, Team Foundation Server

Configuration management
Puppet Enterprise

Continuous integration
Atlassian Bamboo, Go, Jenkins, TeamCity, Travis CI

Deployment
Capistrano, MCollective [part of Puppet Enterprise]

Monitoring
New Relic, Nagios, Splunk, AppDynamics, Loggly, Elastic

Puppet Modules

You'll find more than 3,000 modules on the Puppet Forge, written by Puppet Labs employees and Puppet community members, to help you automate just about anything. A few are highlighted here.

puppetlabs/vcsrepo Provides a type to manage repositories from various version control systems.

puppetlabs/ntp. Installs, configures and manages the NTP service.

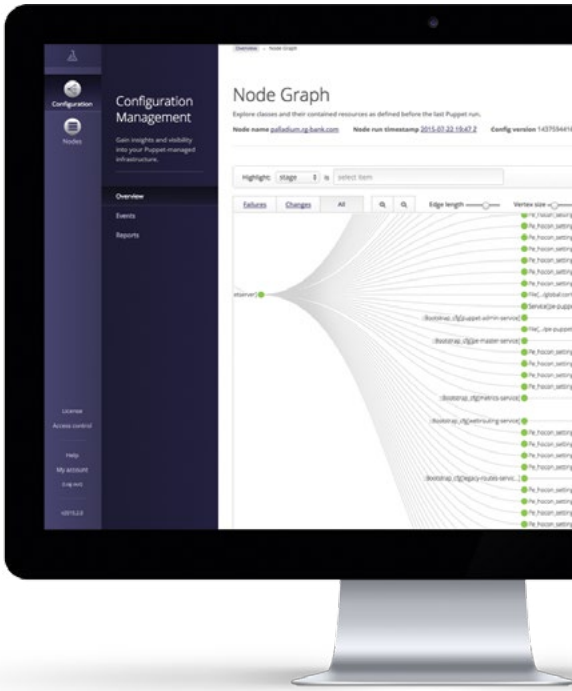
elasticsearch/elasticsearch. Manage and configure Elasticsearch nodes.

rtyler/jenkins. Manage the Jenkins continuous integration service with Puppet.

sensu/sensu. A module to install the Sensu monitoring framework.

Puppet Enterprise:
Automate and Get Your Life Back

Puppet Enterprise gives you the power to easily automate repetitive tasks, quickly deploy critical applications, and proactively manage infrastructure, both on-premise and in the cloud.



Give Puppet Enterprise a try:

- Download the Learning VM, which comes pre-installed with Puppet Enterprise and guides you through a series of fun quests to learn the fundamentals.
- Explore our online and instructor-led courses.
- Try Puppet Enterprise for free on up to 10 nodes.

Questions?

Stop by ask.puppetlabs.com or contact our sales team.