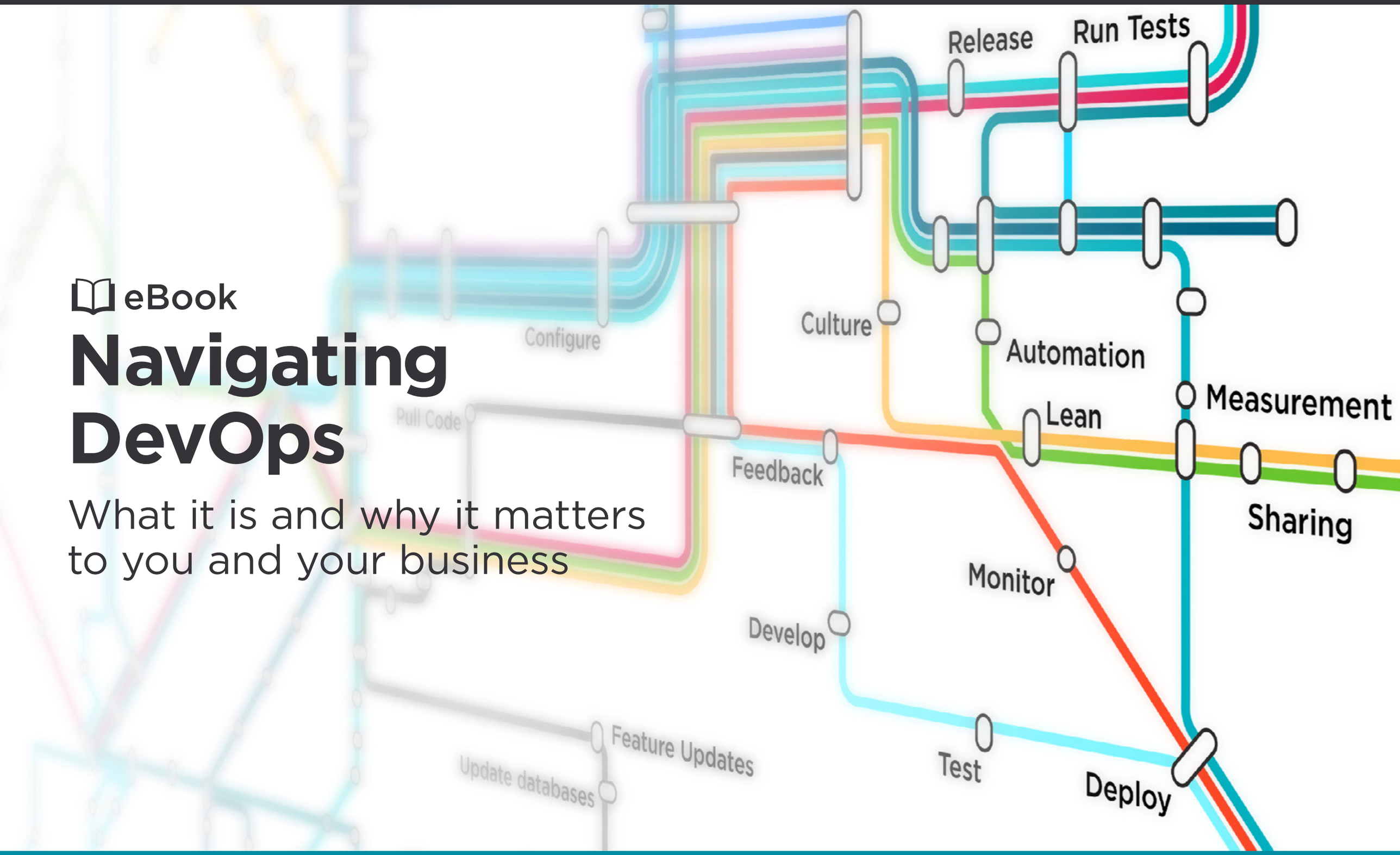
New Relic®

eBook

Navigating DevOps

What it is and why it matters to you and your business



 | US +888-643-8776 | www.newrelic.com | www.twitter.com/newrelic | blog.newrelic.com

Table of Contents

Introduction

There are as many opinions about DevOps as there are commentators, but one thing is undeniable: DevOps is real. In a few short years, headlines about DevOps have gone from "What the F--- Is DevOps?"[1] and "DevOps Is a Poorly Executed Scam"[2] to "The New Normal of DevOps"[3] and "Three Reasons Your Startup Will Suffer Without DevOps."[4] From tiny startups to giant Fortune 500 enterprises, the IT industry is adopting DevOps at an amazing rate.

And yet, the majority of IT professionals either don't know anything about DevOps or just have a general understanding of the big picture. If you fall into either of those categories, this discussion is meant for you. In the following pages, we answer a number of basic questions—questions that probably sound much like your own:[5]

- What is DevOps?
- Where did it come from?
- What problems led to DevOps?
- How does DevOps "work?"
- How widely used is DevOps today?
- Why are people adopting DevOps?
- What are the benefits?

[1] James Turnbull, "What the F*** is DevOps?," blog post, https://coderwall.com/p/zhf8gq, May 2010.
[2] Ted Dziuba, "Devops is a Poorly Executed Scam," blog post, http://widgetsandshit.com/teddziuba/2011/03/devops-scam.html, March 2011.
[3] "The New Normal of DevOps," CA Technologies White Paper, http://www.ca.com/us/~/media/Files/whitepapers/the-new-normal-of-devops.pdf, December 2013.
[4] "Three Reasons Your Startup Will Suffer Without DevOps," Readwrite, http://readwrite.com/2014/01/01/three-reasons-your-startup-needs-devops-or-else.
[5] Damon Edwards, "The History of DevOps,"IT Revolution Press, http://itrevolution.com/the-history-of-devops.

CHAPTER 1

What Is DevOps?

What Is DevOps?

The word DevOps itself was coined in 2009 by Patrick Debois, who became one of its gurus. The term was formed by combining "development" and "operations," which provides a starting point for understanding exactly what people mean when they say "DevOps." Notably, DevOps isn't a process or a technology or a standard. Many devotees refer to DevOps as a "culture"—a term that New Relic favors. We also use the term "DevOps movement" when talking about topics such as adoption rates and trends for the future and "DevOps environment" to refer to an IT organization that has adopted DevOps culture.

This primer will have a great deal more to say about DevOps, but to get started, we need a serviceable definition:

> *DevOps (a portmanteau of development and operations) is a software development method that stresses communication, collaboration and integration between software developers and information technology (IT) operations professionals.*[6]

"Successful DevOps is mutual respect between development and operations. You deliver quality code and I'll deliver a quality platform to run that code. We can have differences of opinion about how to do it, but at the end of the day, we are together going to deliver software that performs as advertised and meets our customers' needs."[7]

—Mike Surma, DevOps Engineer, Rackspace

[6] "Big Data Drives Rapid Changes in Infrastructure and $232 Billion in IT Spending Through 2016," Gartner Research, October 2012.

[7] Interview, March 13, 2014

CHAPTER 2

Where Did DevOps Come From?

Where Did DevOps Come From?

Despite the mythical tone of some of the stories about its origins, DevOps was not created out of whole cloth. Rather, the seeds of DevOps were planted long ago and have been nurtured by forward-thinking IT experts in a number of disciplines. One commentator calls DevOps a "perfect storm,"[8] and there's plenty of evidence to back that up. The two primary antecedents of DevOps are:

- **Enterprise systems management (ESM).** Many of the people involved in the initial definition of DevOps were system administrators. These operations experts brought key ESM best practices to DevOps, including configuration management, system monitoring, automated provisioning, and the toolchain approach.

- **Agile development.** One observer calls DevOps "agile on steroids."[9] DevOps incorporates a number of agile principles, methods, and practices such as continuous delivery, continuous integration, and collaboration.[10] DevOps brings agile principles to system administration and ultimately to the full spectrum of IT operations. In the words of one DevOps pioneer with 15 years of system administration experience, shaking up the operations side of the house was long overdue:

System administrators have allowed themselves to lag in maturity behind what the state of the art is. These new technologies are finally causing us to be held to account to modernize the way we do things. And I think that's a welcome and healthy challenge.[11]

Why does this matter? It goes to the basic credibility of the movement. Far from being some outlandish manifesto of a few IT geeks, DevOps is actually quite mainstream in its origins. DevOps unites an established IT operations discipline with a proven development methodology—each half of the acronym represents the best practices of its craft. And that union mirrors the fact that DevOps integrates development and operations into a single-minded entity with common goals: high-quality software, faster releases, and improved customer satisfaction—again, totally mainstream enterprise concerns.

[8] Ernest Mueller, "What is DevOps?" the agile admin, http://theagileadmin.com/what-is-devops (December 2011).
[9] Aniket Deshpande, "DevOps" an Extension of Agile Methodology – How It will Impact QA?," Software Testing Help, http://www.softwaretestinghelp.com/devops-and-software-testing.
[10] Jack Crews,"Agile Values, Principles and Practices," online presentation, http://www.slideshare.net/jackcrews/agile-values-principles-and-practices.
[11] Ernest Mueller, "Q&A: Ernest Mueller on Bringing Agile to Operations," http://dev2ops.org/2010/04/qa-ernest-mueller-on-bringing-agile-to-operations, 2010.

CHAPTER 3

What Problems Led to the Creation of DevOps?

What Problems Led to the Creation of DevOps?

Developers and system administrators don't see eye to eye on a lot of things, but they do agree that their customers on the business side of the house frequently pull them in two different directions. On the one hand, business users demand change—new features, new services, new revenue streams—as fast as possible. At the same time, they also want a system that is stable and free from interruptions. That creates a problem:

> *The problem with the traditional software delivery process (or the lack thereof) is that it is not well adapted to support these two requirements simultaneously. So companies have to choose between either delivering changes fast and ending up with a messy production environment or keeping a stable but outdated environment.*[12]

Not surprisingly, neither choice is acceptable to enterprise executives. And more importantly, neither allows a business to provide the best solutions it can to it's customers.

Developers are all too willing to push out software faster and faster—after all, that's what they are typically hired to accomplish. Operations, on the other hand, knows that rapid-fire changes without proper safeguards threaten to destabilize the system, which goes directly against their charter.

DevOps was created to resolve this dilemma by integrating everyone associated with software development and deployment—business users, developers, test engineers, system administrators—into a single, highly automated workflow with a laser focus: Rapid delivery of high-quality software that meets all user requirements while maintaining the integrity and stability of the entire system.

How do these disparate groups join forces? By subscribing to a common set of principles that transcends traditional discipline boundaries and roles, for example:

- Ensure all teams are working towards the same goal and are being measured by the same business metrics.
- Maintain short development cycles that enable the business to pivot quickly with changing requirements.
- Utilize feature flags and progressive deployment strategies that make it easy to enable or disable new features in production without re-deployments.
- Create extremely fast feedback loops that allow for almost immediate problem identification and remediation by the appropriate teams.
- Reflect on how to become more effective as a team, then tune and adjust your behavior accordingly.[13]

[12] Niek Bartholomeus, "My experience with introducing DevOps in a traditional enterprise," http://niek.bartholomeus.be/2013/01/28/introducing-a-devops-culture-in-a-traditional-enterprise, January 28, 2013.

[13] Ernest Mueller, "A DevOps Manifesto," the agile admin, http://theagileadmin.com/2010/10/15/a-devops-manifesto, October 15, 2010.

CHAPTER 4

How Does DevOps "Work?"

How Does DevOps "Work?"

Like all cultures, DevOps has many variations on the theme. However, most observers would agree that the following capabilities are common to virtually all DevOps cultures: collaboration, automation, continuous integration, continuous delivery, continuous testing, continuous monitoring, and rapid remediation.

Collaboration

Instead of pointing fingers at each other, development and IT operations work together (no, really). While the disconnect between these two groups created the impetus for its creation, DevOps extends far beyond the IT organization, because the need for collaboration extends to everyone with a stake in the delivery of software (not just between Dev and Ops, but all teams, including test, product management, and executives):

> *Successful DevOps requires business, development, QA, and operations organizations to coordinate and play significant roles at different phases of the application lifecycle. It may be difficult, even impossible, to eliminate silos, but collaboration is essential.*[14]

Automation

DevOps relies heavily on automation—and that means you need tools. Tools you build. Tools you buy. Open source tools. Proprietary tools. And those tools are not just scattered around the lab willy-nilly: DevOps relies on toolchains to automate large parts of the end-to-end software development and deployment process.

Caveat: Because DevOps tools are so amazingly awesome, there's a tendency to see DevOps as just a collection of tools. While it's true that DevOps relies on tools, DevOps is much more than that.

Continuous Integration

You usually find continuous integration in DevOps cultures because DevOps emerged from agile culture, and continuous integration is a fundamental tenet of the agile approach:

> *Continuous integration (CI) is a software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base. The goal of CI is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible…the usual rule is for each team member to submit work on a daily (or more frequent) basis and for a build to be conducted with each significant change.*[15]

The continuous integration principle of agile development has a cultural implication for the development group. Forcing developers to integrate their work with other developers frequently—at least daily—exposes integration issues and conflicts much earlier than is the case

[14] Laurie Wurster et al, "Emerging Technology Analysis: DevOps a Culture Shift, Not a Technology," Gartner report, August 2013.

[15] Margaret Rouse, "Continuous Integration (CI)," SearchSoftwareQuality, http://searchsoftwarequality.techtarget.com/definition/continuous-integration, July 2008.

with waterfall development. However, to achieve this benefit, developers have to communicate with each other much more frequently—something that runs counter to the image of the solitary genius coder working for weeks or months on a module before she is "ready" to send it out in the world. That seed of open, frequent communication blooms in DevOps.

Continuous Testing

The testing piece of DevOps is easy to overlook—until you get burned. As one industry expert puts it, "The cost of quality is the cost of failure."[16] While continuous integration and delivery get the lion's share of the coverage, continuous testing is quietly finding its place as an equally critical piece of DevOps.

Continuous testing is not just a QA function, in fact, it starts in the development environment. The days are over when developers could simply throw the code over the wall to QA and say, "Have at it." In a DevOps environment, everyone is involved in testing. Developers make sure that, along with delivering error-free code, they provide test data sets. They also help test engineers configure the testing environment to be as close to the production environment as possible.[17]

On the QA side, the big need is speed. After all, if the QA cycle takes days and weeks, you're right back into a long, drawn out waterfall kind of schedule. Test engineers meet the challenge of quick turnaround by not only automating much of the test process but also redefining test methodologies:

> *Rather than making test a separate and lengthy sequence in the larger deployment process, continuous delivery practitioners roll out small upgrades almost constantly, measure their performance, and quickly roll them back as needed.*[18]

Although it may come as a surprise, the operations function has an important role to play in testing and QA:

> *Operations has access to production usage and load patterns. These patterns are essential to the QA team for creating a load test that properly exercises the application.*[19]

Operations can also ensure that monitoring tools are in place and test environments are properly configured. They can participate in functional, load, stress, and leak tests and offer analysis based on their experience with similar applications running in production.

The payoff from continuous testing is well worth the effort. The test function in a DevOps environment helps developers to balance quality and speed. Using automated tools reduces the cost of testing and allows test engineers to leverage their time more effectively. Most importantly, continuous testing shortens test cycles by allowing integration testing earlier in the process.

[16] Wayne Ariola, SYS-CON.tv interview at the 13th International Cloud Expo®, http://devopssummit.sys-con.com/node/2912573, Nov 4-7, 2013.
[17] Sanjeev Sharma, "Understanding DevOps – Part 4: Continuous Testing and Continuous Monitoring," blog post, http://sdarchitect.wordpress.com/2012/10/30/understanding-devops-part-4-continuous-testing-and-continuous-monitoring , October 2012.
[18] http://www.pwc.com/us/en/technology-forecast/2013/issue2/features/devops-continuous-delivery.jhtml.
[19] Jim Hirschauer, "DevOps Scares Me – Part 4: Dev and Ops Collaborate Across the Lifecycle," DZone, http://java.dzone.com/articles/devops-scares-me-part-4-dev, August 2013.

Continuous testing also eliminates testing bottlenecks through virtualized dependent services, and it simplifies the creation of virtualized test environments that can be easily deployed, shared, and updated as systems change. These capabilities reduce the cost of provisioning and maintaining test environments, and they shorten test cycle times by allowing integration testing earlier in life cycle.[20]

Continuous Delivery

In the words of one commentator, "continuous delivery is nothing but taking this concept of continuous integration to the next step."[21] Instead of ending at the door of the development lab, continuous integration in DevOps extends to the entire release chain: including QA and operations. The result is that individual releases are far less complex and come out much more frequently.

The actual release frequency varies greatly depending on the company's legacy and goals. For example, one Fortune 100 company improved its release cycle from once a year to once a quarter—a release rate that seems glacial compared to the hundreds of releases an hour achieved by Amazon.

Exactly what gets released varies as well. In some organizations, QA and operations triage potential releases: many go directly to users, some go back to development, and a few simply are not deployed at all. Other companies—Flickr is a notable example—push everything that comes from developers out to users and count on real-time monitoring and rapid remediation to minimize the impact of the rare failure.

Continuous Monitoring

Given the sheer number of releases, there's no way to implement the kind of rigorous pre-release testing that characterizes waterfall development. Therefore, in a DevOps environment, failures must be found and fixed in real time. How do you do that? A big part is continuous monitoring.

According to one pundit, the goals of continuous monitoring are to quickly determine when a service is unavailable, understand the underlying causes, and most importantly, apply these learnings to anticipate problems before they occur.[22] In fact, some monitoring experts advocate that the definition of a service must include monitoring—they see it as integral to service delivery.

Like testing, monitoring starts in development. The same tools that monitor the production environment can be employed in development to spot performance problems before they hit production.

Two kinds of monitoring are required for DevOps: server monitoring and application performance monitoring. Monitoring discussions quickly get down to tools discussions, because there is no effective monitoring without the proper tools. For a list of DevOps tools (and more DevOps-related content), visit New Relic's DevOps Hub.

[20] "Enterprise testing capability for continuous software delivery," http://www.ibm.com/ibm/devops/us/en/build/test/.
[21] Sanjeev Sharma, "Understanding DevOps – Part 2: Continuous Integration and Continuous Delivery," blog post, http://sdarchitect.wordpress.com/2012/09/25/understanding-devops-part-2-continuous-integration-and-continuous-delivery, September 2012.
[22] Julien Pivotto, "The devops approach to monitoring," Open World Forum presentation, http://www.slideshare.net/roidelapluie/devops-andmonitoringowf13, October 2013.

CHAPTER 5

Who's Adopting DevOps?

Who's Adopting DevOps?

DevOps adoption is accelerating. A December 2012 study of more than 4,000 IT professionals found that 63% of respondents had implemented DevOps practices, an astonishing 26% increase over the previous year.[23] While not yet a mainstream trend, DevOps is clearly gaining momentum.[24]

What kinds of companies are—or should be—embracing DevOps? It depends on who you ask. One school of thought sees DevOps as the right answer for everybody, from small startups to tech giants: "DevOps is now a way of life and ignoring it can make or break a company of any size.[25]

Others contend that size matters—a lot. According to this view, DevOps is a piece of cake for smaller companies, but the culture of the large corporation may be a significant problem: "If good collaboration is missing from the DNA of the company, it will take more time and effort to introduce DevOps successfully."[26]

The plain fact is that DevOps is evolving so rapidly that it's difficult even to characterize the level of adoption in any meaningful way. However, there's plenty of evidence showing that size by itself is no predictor of DevOps success.

Take the big guys first. Ask a dozen DevOps converts to name the biggest success stories in DevOps and most will give the same four or five names: Google, Twitter, Amazon, Netflix, and Facebook. Their combined market capitalization of more than USD $717 billion in 2014 exceeds the GDP of all but 19 countries in the world. Many industry watchers believe that DevOps—or something that looks a lot like DevOps—has been an essential component in their meteoric growth. "These are the breakout successes that could only be possible with a new operating philosophy...Their success in many ways is a roadmap for how DevOps can succeed at all different sizes of organizations."[27]

On the other end of the spectrum are the "hip, slick, and cool" startups and medium-sized businesses—Tumblr, Evernote, Etsy, and Github, to name a few—who play with the big boys in large part because DevOps helps them leverage small teams into outsized revenues: "Traditional IT processes and product development methodologies weren't going to help these small companies punch above their weight."[28]

[23] "2013 State of DevOps Report," Puppet Labs and IT Revolution Press, https://puppetlabs.com/wp-content/uploads/2013/03/2013-state-of-devops-report.pdf.
[24] Kyt Dotson, "DevOps Interview: Raja Bhargava, CEO of JumpCloud http://devopsangle.com/2014/01/07/devops-interview-rajat-bhargava-ceo-of-jumpcloud/.
[25] Adam Duro, "Three Reasons Your Startup Will Suffer Without DevOps," readwrite, http://readwrite.com/2014/01/01/three-reasons-your-startup-needs-devops-or-else, January 01, 2014.
[26] "The New Normal of DevOps," CA Technologies White Paper, http://www.ca.com/us/~/media/Files/whitepapers/the-new-normal-of-devops.pdf, December 2013.
[27] James B. Brown, "5 Reasons Why DevOps is Hitting Its Stride," Innovation Insights, http://insights.wired.com/profiles/blogs/5-reasons-why-devops-is-hitting-its-stride#ixzz2x6O8QnIA, March.
[28] Ibid.

CHAPTER 6

Why Are Your Peers Embracing DevOps?

Why Are Your Peers Embracing DevOps?

DevOps has something for everyone in the software chain: developers, operations, and testing. Furthermore, DevOps even touches the business side of the house: managers who monetize the software and executives who worry about the bottom line. Here are some of the benefits cited by each group.

Developers

Automated provisioning is a big win for programmers, because they can stand up a development environment themselves with no paperwork, no lengthy approval cycles, no waiting for IT to provision a server—no lost time. When developers can provision a working environment in 15 minutes, with all the right resources—compute power, storage, network, applications—it changes the way that they work. They can be far more creative and innovative. It's much easier to try multiple options, run different scenarios, and test their code more thoroughly.

When developers first begin to work in a DevOps world, one real eye-opener for many is understanding just what goes on inside that black box labeled "Operations." That knowledge helps developers work effectively with operations in a joint problem-solving mode. Problems are resolved faster and cause fewer distractions. Best of all, the frantic late-night phone call becomes a thing of the past—and that leads directly to greater job satisfaction and better quality of life for developers.

Operations

There's a widespread belief that system administrators constantly obsess about system stability—and in fact, it's true. Their nightmare scenario is a software release that takes down the system within seconds of production deployment, developers who shrug off responsibility ("It's your code now!"), users in various degrees of outrage—and no clear path to a quick, effective resolution.

Early adopters of DevOps methods have found that the increased involvement by developers actually improves system stability. Automation also helps by eliminating human errors common in manual operations, and has the added benefit of reducing the amount of time spent on routine tasks. There's a quality of life issue for system administrators as well, in the form of skill building, career opportunities, and a great deal more uninterrupted sleep and personal time. In a DevOps environment, operations rely on tools to a much greater extent than in traditional environment, often building their own tools and writing scripts that automate portions of the deployment process.

Test Engineers

The impact that DevOps has had on the testing side of the house can be summed up in two words: Chaos Monkey. Netflix developed this remarkable, innovative tool to break its own software![29]

> *Chaos Monkey works on the principle that the best way to avoid major failures is to fail constantly. The software simulates failures of instances of services…by shutting down one or more of the virtual machines…In this way, it's possible to prepare for major unexpected errors rather than just waiting for catastrophe to strike and seeing how well you can manage.[30]*

DevOps requires new ways to test software, which challenges test engineers to innovate on their side of the house—exactly the impetus that led to the creation of Chaos Monkey. With automated provisioning, test engineers can provision a test environment that is virtually identical to the production environment, resulting in more accurate testing and better ability to predict the performance of new releases. As with other groups, test engineer productivity increases thanks to automation and collaboration.

Product Managers

Technically, DevOps is just about the IT function of the enterprise. However, those who have made the transition will tell you that DevOps changes everything:

> *An effective DevOps strategy allows an organization to analyze, for the first time and in real-time, Web analytics data, machine data, and existing structured data to achieve a 360-degree view of how customer-facing systems are and are not delivering business value. DevOps allows organizations to understand the behavior of individual customers, what actions they take and how their behavior compares to that of other customers. It allows an organization to perform what-if analyses of changes to their user interface or product offers. Based on the results of these predictive analyses, and of actual field experience, the Agile "development" side of DevOps then allows organizations to respond to market needs more quickly than ever before.[31]*

Let's break that down a bit. In a DevOps environment, business stakeholders have greater influence on the development process. Thanks to the collaborative spirit of DevOps, developers actually care about business requirements and foster relationships with product managers. DevOps also gives product managers immediate feedback about the impact of new pricing, features, and product bundles, which allows them to test variations and gauge their effectiveness.

Line of business (LOB) managers love DevOps because software gets to market faster—giving them the competitive edge that they crave. Because DevOps improves system stability, customers experience fewer outages and are therefore more loyal—the perfect cure for high churn rates.

[29] http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html.
[30] Margaret Rouse, "Chaos Monkey," WhatIs.com, http://whatis.techtarget.com/definition/Chaos-Monkey, August 2013.
[31] Kapil Apshankar, "How DevOps Drives Real-Time Business Growth," Cognizant white paper, http://www.cognizant.com/InsightsWhitepapers/How-DevOps-Drives-Real-Time-Business-Growth.pdf , 2013.

Executives

When Patrick Debois and other IT wizards started the DevOps movement, they surely weren't concerned with how it would be received in the corporate boardroom. Just five years later, DevOps is a hot topic in those same boardrooms.

What do executives like about DevOps? For one thing, it helps the organization deliver high-quality products and get them to market much faster than competitors with traditional methods of software development—actions that impact the bottom line and build brand value. Another reason is the ability to attract and retain top talent: high-quality developers, system administrators, and test engineers want to work on the latest and greatest. Finally, when developers, operations, and QA work together, top executives rarely get pulled into inter-departmental disputes, leaving them more time to craft the focused business goals that everyone is now pulling together to reach successfully.

"My mindset was to slow down the release process. I wanted to have as few releases as possible so that I could be sure that the system would be stable. After working for several years in a DevOps environment, I've done a complete 180 degree turn. Now my philosophy is, the more frequently we deploy, the better."[32]

—Aaron Bento, Site Reliability Engineer, New Relic

[32] Interview, February 6, 2014

CHAPTER 7

How Will I Benefit from DevOps?

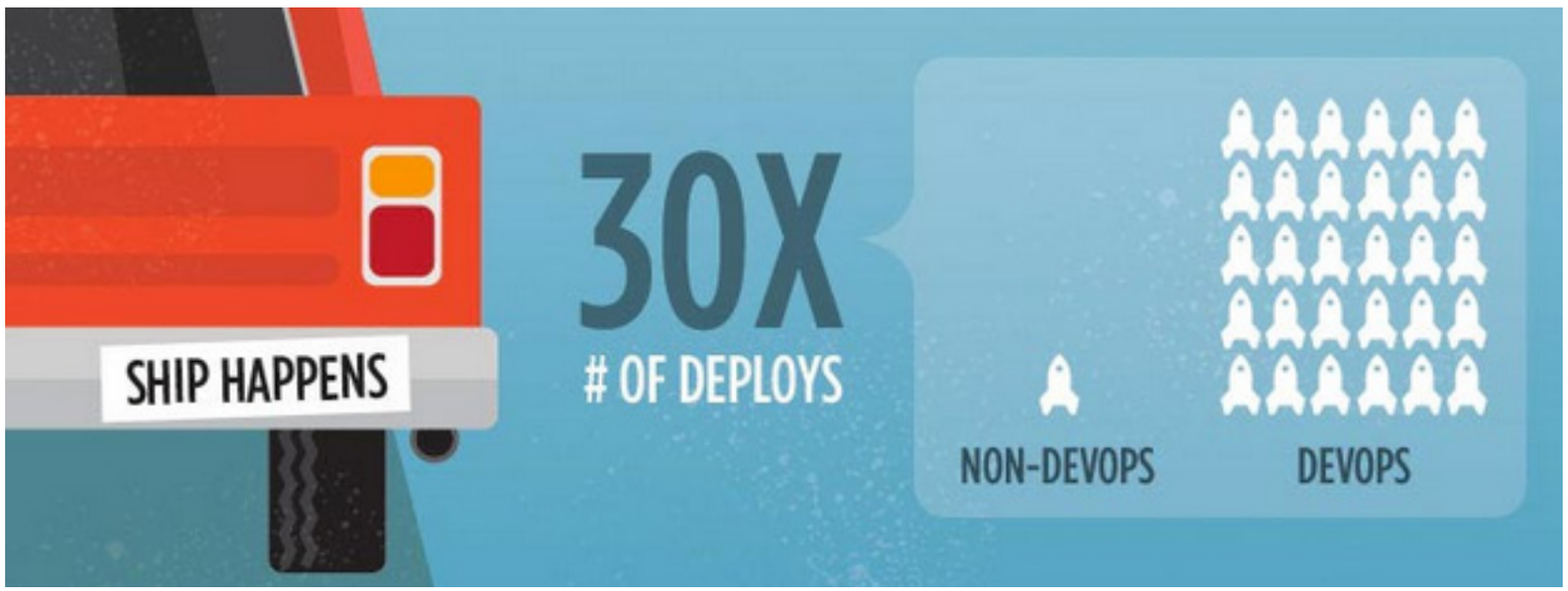
How Will I Benefit from DevOps?

Credible sources report some pretty remarkable benefits achieved with DevOps. However, caution is in order. Suppose you overheard someone saying, "I'm getting 30 miles to the gallon." What car? What kind of driving? If he's talking about an F-150 truck driven off-road, the number is so high that you simply wouldn't believe it. On the other hand, for a new Prius driven exclusively on the highway, 30 mpg would indicate big problems. Context matters. So whenever you encounter claims about improvements related to DevOps, be aware that your results may vary.

That said, a survey by Puppet Labs[33] found that DevOps adopters release software 30 times faster than their peers. The quality of the software products is higher, too, as shown by the finding that DevOps applications have half the failures of competitors. Finally, the net effect on system stability is positive: when the platform does go down, DevOps groups restore service 12 times faster than peers.

One thing is obvious: IT professionals who have adopted DevOps tend to be raving fans. It's not hard to see why, given the improvements cited in the same study:

- **Infrastructure stability:** 83 percent of respondents report either "some improvement" or "significant improvement."
- **App deployment speed:** 83 percent report either "some improvement" or "significant improvement."
- **Security:** 45 percent expect DevOps to improve security, while only 7 percent think that systems will be less secure thanks to DevOps.



[33] PuppetLabs survey, op cit.

Conclusion

Five years into the great DevOps experiment, the data is clear: DevOps is here to stay—and for some very good reasons. Many thought it impossible, but DevOps has succeeded in integrating business users, developers, test engineers, and system administrators into a single workflow focused on meeting customer requirements. Why would they willingly do so? Because there's something in it for everyone. Developers and system administrators stop arguing and start supporting each other, lowering blood pressures all around. Business managers are happy because they actually get the software products that they need to sell products and services. Executives watch their beloved dashboard metrics—revenue, customer satisfaction, system reliability—heading steadily north. And everyone is able to deliver the best results and overall experience possible to the customer.

Gains like these, however, don't come easily. In order to successfully deploy code more frequently while keeping your systems humming, you need the ability to accurately monitor all the changes going on in your environment. New Relic provides the data you need to measure and monitor the new features the Dev team delivers, while ensuring the stability that the Ops team requires. To learn more about DevOps and how New Relic can help your organization successfully make the transition, visit: http://newrelic.com/devops.

"One of the best messages that DevOps has for all of us is, if you don't like the way someone's doing something, talk to them and find out why. You'll probably learn that there were great reasons for everything, including 'I didn't know how to do it.'"[34]

—Sascha Bates, Consultant, Chef

[34] Sascha Bates, "Shenanigans," http://blog.brattyredhead.com/, May 17th, 2013.

About New Relic

New Relic is a software analytics company that makes sense of billions of metrics about millions of applications in real time. Our comprehensive SaaS-based solution provides one powerful interface for web and native mobile applications and consolidates the performance monitoring data for any chosen technology in your environment. Our 90,000 customers use our cloud solution every day to optimize more than 200 billion metrics for 3 million applications. When your brand and customer experience depend on the performance of modern software, New Relic provides insight into your overall environment. Learn more at newrelic.com.

New Relic, San Francisco HQ

188 Spear Street, Suite 1200

San Francisco, CA 94105

New Relic, Portland

111 SW 5th Avenue, Suite 2800

Portland, OR 97204

Tel: +1.888.643.8776

support@newrelic.com

www.newrelic.com

New Relic, Seattle

2101 4th Avenue, 19th Floor

Seattle, WA 98121

New Relic, Dublin

34-39 Nassau Street, 3rd Floor

Dublin 2, Ireland