Integrate and Automate

An excerpt from *Lessons from 29 DevOps Experts On The Best Way to Make The Transition to Continuous Delivery*

Download the Full eBook Now!

INTEGRATE AND AUTOMATE

In *Integrate and Automate*, Stelligent CEO Paul M. Duvall reminds us what software development is all about. He tells the story of a software application he helped create that was ultimately deployed at a hospital logistics center. It was a thrilling moment—a reminder of what motivates him as a developer.

But there was a nagging problem. That release took two years to deliver. "The primary reason for this delay," Duvall writes, "was system complexity: a 70-person development team, software that had to be manually installed at the logistics centers, and so on. So, the work got batched and delayed and delayed." It was, he writes, a maddening process.

Continuous delivery means integrating the entire software system with every code commit. It reduces the time it takes for end users to get their software, and it provides fast, actionable feedback to production teams. Automation is its secret sauce. This mini-e-book, Integrate and Automate, is the third in a series of six sponsored by Zend. It collects the wisdom of seven of the world's most brilliant DevOps minds.

Web developer and blogger Scott Hanselman makes the point emphatically. "The most powerful tool we have as developers is automation: I can't stress this point enough," he writes. Humans, after all, do repetitive tasks poorly, while computers are designed for them. Seek ways to automate processes down to a single button press, he writes. That will breed confidence, both among end users and inside your organization.

None of the advice to be found in *Integrate and Automate* is more cogent than that offered by Michael Dehaan, CTO at Ansible and creator of several DevOps automation tools. Successfully automating processes, he says, is about creating a culture of automated testing, both at the developer and quality assurance levels. Achieving that might take time, but ultimately, he writes, you will be able to "deploy a dozen times an hour with confidence."

Andi Gutmans, CEO and co-founder of Zend, recalls stumbling onto DevOps practices before any such thing existed. He was working on jet avionics simulators in a complex production environment comprising more than 100 binaries and various hardware drivers created in several languages. There was no easy debug for that. But Gutmans could draw from automation experience he had derived as a systems administrator. He used that experience to deliver a one-click build environment for the avionics simulators.

The same thing would not be nearly so difficult now. "Today, with the methodology and all the great tools and best practices that exist to support DevOps and continuous delivery, there's no excuse not to embrace it fully," Gutmans writes. Automation, in effect, does all the heavy lifting for you.

Integrate and Automate is one chapter of a larger e-book, ***Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery.*** The publication provides best practices and advice from the top DevOps industry leaders. For those wanting to learning more about implementing continuous delivery, this book covers each step: getting started in continuous development, integrating and automating the process, getting the team on board, changing the culture, and best practices for the future. **Download the full e-book now** to take advantage of these expert insights and determine whether continuous delivery is right for your business.

- Kevin Featherly

FOREWORD

Exploring Continuous Delivery

Innovation has changed. Gone are the days when a solitary genius holed up in a garage conceived a big idea, and then painstakingly perfected and brought it to market years later. Today, innovation is fluid, fast moving, and collaborative. Innovation is the engine for growth and value creation in the modern world, and software is the fuel.

The ability to create new, high-quality software applications and bring them to market more quickly is the "X factor" that defines industry leaders, and these leaders all have one thing in common: their IT organizations are leaving traditional approaches behind in favor of new, agile, collaborative approaches to the design, development, and delivery of applications.

At Zend, we are committed to helping companies deliver innovation more quickly. We've seen the dramatic results of this trend in working with Fiat, Hearst Corporation, BNP Paribas, Newell Rubbermaid, Prada, and other customers that are achieving faster and more frequent releases of more reliable software and, as a result, improving their business growth and profitability. Like other companies around the world, their success stems from the adoption of Continuous Delivery methodologies and best practices.

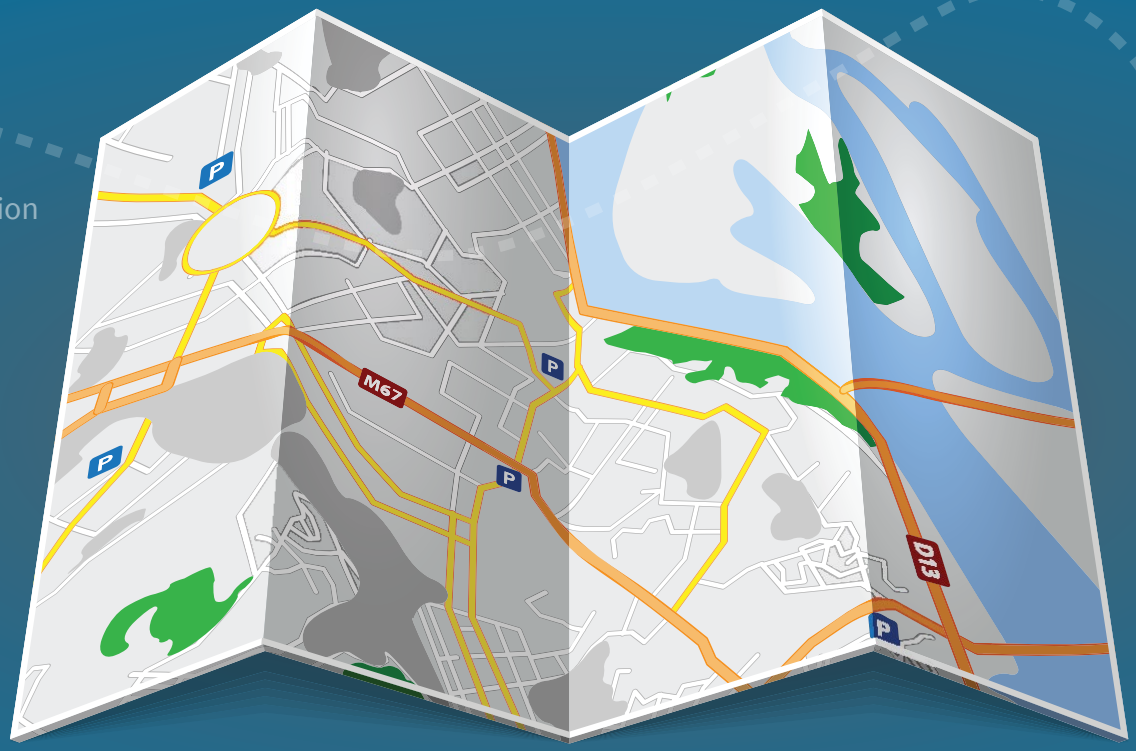
This e-book has been created for companies at virtually any stage of the journey toward Continuous Delivery. In the following pages, you'll find essays from software industry leaders whose experiences, insights, and solutions can make it a lot easier to get started, progress smoothly, and finish strong.

Wishing you the best success,
Andi Gutmans
CEO, Zend

zend

Zend helps businesses deliver innovation more quickly, on a larger scale, and across more channels than ever before. More than 40,000 companies rely on our solutions, including Zend Server, the integrated application platform for mobile and web apps. Zend Server provides superior tools for creating high-quality code, best-in-class infrastructure for moving applications from source control through deployment, and the best back-end platform for performance at Web scale. Zend helped establish PHP, which today powers more than 240 million applications and websites around the world. Visit us at www.zend.com.

Continuous Delivery is a Journey



Get buy-in

Integrate & Automate

Adopt best practices

ROI validation

Build the business case

Get started

We'll meet you wherever you are,
with the resources you need to succeed.

Learn More

> The ability to create new, high-quality software applications and bring them to market more quickly is the "X factor" that defines industry leaders.
>
> **Andi Gutmans**, CEO & Co-founder, Zend

zend

INTRODUCTION

Continuous Delivery isn't just a technical shift, it's a cultural one. Even though it takes hard work to make the transition, the benefits can't be ignored. Faster time to market, better quality product, competitive advantage, higher customer satisfaction and reduced cost of development are just a few of the benefits driving CD to become the new norm.

With the support of Zend, we reached out to 29 top DevOps professionals and asked them the following question:

Your friend has been tasked with transitioning her company's software development efforts to Continuous Delivery. She's extremely capable, but she's nervous about leading the transition. Please share a story from your own experience that will provide her with a critical piece of advice that will help her to be more successful.

The response was fantastic. Not only did we receive insightful essays, but the expert advice came from the very people who have been leading this revolution – people like Gene Kim, Andi Gutmans, Rebecca Parsons, Scott Hanselman and Andrew Yochum. The essays in this book roughly break down into six categories that range from understanding the business case for CD through actually making the journey. We hope the collective wisdom and hard-learned lessons contained in these pages will inspire you and help you take your own development efforts to a higher level.

All the best,
David Rogelberg
Editor

Integrate and Automate

PAUL M. DUVALL
CEO at Stelligent

Paul M. Duvall is the CEO of Stelligent, which offers cloud delivery solutions in Amazon Web Services. Winner of the 2008 Jolt Award, he is the principal author of *Continuous Integration: Improving Software Quality and Reducing Risk* (Addison-Wesley, 2007), *DevOps in the Cloud* (Addison-Wesley, 2012) and two IBM developerWorks series on automation, DevOps, and cloud computing. He is passionate about software delivery and the cloud and actively blogs at www.stelligent.com.

Twitter | Website | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

KEY LESSONS

1. **RELEASING AND RECEIVING SOFTWARE IS WHAT MOTIVATES CREATORS AND USERS, RESPECTIVELY.**
2. **INTEGRATION IS THE KEY TO CONTINUOUS DELIVERY.**

It's about releasing software to users. When starting my career, one of my first jobs was as a programmer, but I didn't just write application code. I was involved in writing systems code and configuration across the entire software systems stack, including security, systems administration, diagnostics and monitoring, user role management, and later an application framework that more than 70 software developers used. I was writing code in many languages—C++, PowerBuilder, and several others I've since forgotten. In this role, I learned how the entire system was constructed across team and system silos.

The other lesson I learned is what motivates us. Releasing and receiving software motivates both the creators and the users, not silly Hawaiian shirt days or contrived "team-building" exercises. It's about releasing software.

After two long years on this project, I was thrilled when the software I helped create was used at a hospital logistics center. I immediately knew that this was what motivated my work. I think if you asked most people who develop and deliver software systems, and then talk with users, you'd get near universal agreement. Yet, we waited two years to release the software system to users. The primary reason for this delay was system complexity: a 70-person development team, software that had to be manually installed at the logistics centers, and so on. So, the work got batched and delayed and delayed. I found it maddening that it would take us days to get just the application portion (not the environment or databases) of the software system built and deployed within a simple shared integration environment.

It was also while I was on this project that I saw a single-page industry magazine advertisement with a picture of a keyboard button and the word Integrate on it. To me, it symbolized the means to an end. The more we could integrate the complete software system with recent changes, the better the chance that we could regularly release software to users while collectively motivating the software users and the software creators. This is what motivates me every day and why we create continuous delivery systems for our customers today. With continuous delivery, you're integrating the entire software system (application code, configuration, infrastructure, and data) with every code commit. In doing so, you're reducing the time it takes to release software to users while providing rapid and actionable feedback to team members, getting both the users and the creators closer to the collective goal.

> ***"I found it maddening that it would take us days to get just the application portion (not the environment or databases) of the software system built and deployed within a simple shared integration environment."***

WHAT YOU ARE CAPABLE OF WITH AUTOMATION

SCOTT HANSELMAN

Principal Program Manager at Scott Hanselman

Scott Hanselman is a web developer who has been blogging at hanselman.com for more than a decade. He works in open source on Microsoft ASP.NET and the Microsoft Azure Cloud out of his home office in Portland, Oregon. He has written many books and spoken in person to almost half a million developers worldwide.

Twitter | Website | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

If you do it twice, automate it. I'm consistently shocked when I find large companies opening two file windows and deploying mission-critical sites from Joe's laptop. The reasons usually come down to, "We didn't have the time" or "We lacked organizational will."

The most powerful tool we have as developers is automation: I can't stress this point enough. Humans are lousy at doing repetitive work, but computers excel at it. Look hard for the opportunity to automate processes down to a single button push. Even better, can you remove the button entirely? When you automate something, see if you can automate even further.

Years ago, I worked on a large banking system. We were proud to have automated the build and test, and then eventually got to the point where we "deployed" the binaries into a folder. However, we realized quickly that we don't ship binaries: we ship complete solutions.

We changed our definition of *deployment* and expanded the continuous deployment system to generate a fresh virtual machine (VM) for the sales department after each build. Sales loved demoing new features but never had the time to build new demo banks. Of course not: that was our job!

We automated the build system to pop completely configured, fresh VMs out the other end. The added benefit, of course, was that we now had a build product that looked exactly like the product we sold.

Our rapidly maturing continuous deployment expanded to include better unit tests, better integration tests, and better custom acceptance tests. In fact, everything gets better when you're no longer afraid of your deployment system. Imagine the confidence we felt in our code! We made a code change and a bank appeared an hour later downstream. If a bank came out that we didn't trust, we updated our tests and deployments and tried again.

Good continuous deployment systems breed confidence, both in the product and in the organization. All that time we didn't think we had? We gained time when we got this process down. All that organizational will we thought we lacked? We gained it and more when we realized what we were capable of.

> ***"The most powerful tool we have as developers is automation."***

KEY LESSONS

1. **EVERYTHING GETS BETTER WHEN YOU'RE NO LONGER AFRAID OF YOUR DEPLOYMENT SYSTEM.**
2. **GOOD CONTINUOUS DEPLOYMENT SYSTEMS BREED CONFIDENCE, BOTH IN THE PRODUCT AND IN THE ORGANIZATION.**

AUTOMATE QA FOR SUCCESSFUL CONTINUOUS DELIVERY

MICHAEL DEHAAN

CTO at Ansible

Michael DeHaan is the creator of popular DevOps automation tools like Cobbler and Ansible and co-creator of Func. He serves as CTO of Ansible, Inc., and lives in Raleigh, North Carolina.

Twitter | Website | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

One of the events that led to me creating Ansible was working at a hosted web application company that did infrequent releases and the challenges that doing so caused. Each release involved five or six people stuck in a conference room, manually pulling servers out of monitoring systems and load balancers, manually kicking off updates, and putting them back into load balancing and monitoring.

This process took hours, and invariably something went wrong. Update steps were skipped. Although there was manual quality assurance (QA), regression testing was difficult.

The secret to continuous deployment is automating the process. More importantly, it's about building up a culture of automated testing at both the developer and QA level:

- Deploy development environments with automation that is the same as that used in production to ensure that developers run code in an environment that simulates production.
- When developers check in source code, the continuous deployment system should run unit tests. Coverage and requiring unit tests for each piece of functionality are critically important.
- If the code passes those unit tests, deployments go to a staging environment that is a close mirror of production. Automated integration tests are then run against the staging environment.
- If those tests succeed, run the same automation against production. Embed tests in the rolling update process to ensure that servers that fail are not added back to a load-balanced pool.

Although it takes some time to get there, automated QA coupled with an automation system designed for embedded tests and rolling updates can then let you deploy a dozen times an hour with confidence. In addition, join your local DevOps group! You'll find many like-minded individuals seeking many of the same solutions.

KEY LESSONS

1. **AUTOMATE THE DEPLOYMENT PROCESS.**
2. **BUILD A CULTURE OF AUTOMATED TESTING.**
3. **JOIN A LOCAL DEVOPS GROUP.**

"The secret to continuous deployment is automating the process. More importantly, it's about building up a culture of automated testing at both the developer and QA level."

PACKING APPS FOR EASY DEPLOYMENT

KRIS BUYTAERT

CTO of Inuits

Kris Buytaert is a long-time Linux and open source consultant. In fact, he's one of the instigators of the DevOps movement. Currently with Inuits, Kris spends most of his time working to bridge the gap between developers and operations, with a strong focus on high availability, scalability, virtualization, and large infrastructure management projects. His goal is to build infrastructures that can survive the "10th-floor test," better known today as the *cloud,* while actively promoting DevOps concepts.

Twitter | Website | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

KEY LESSONS

1. **PACKAGE YOUR APPS AS .DEB OR .RPM ARCHIVES FOR EASY DEPLOYMENT.**
2. **WHEN PROVISIONING THE COMPUTER, SPECIFY WHICH VERSION OF YOUR APP YOU WANT TO DEPLOY BY DEFAULT.**

When talking about continuous delivery, people invariably discus their delivery pipeline and the different components that need to be in that pipeline. Often, the focus on getting the application deployed or upgraded from that pipeline is so strong that teams forget how to deploy their environment from scratch.

After running a number of tests on the code , compiling it where needed, people want to move forward quickly and deploy their release artifact on an actual platform. This deployment is typically via a file upload or a checkout from a source-control tool from the dedicated computer on which the application resides. Sometimes, dedicated tools are integrated to simulate what a developer would do manually on a computer to get the application running. Copy three files left, one right, and make sure you restart the service. Although this is obviously already a large improvement over people manually pasting commands from a 42 page run book, it doesn't solve all problems.

For example, it doesn't take into account that you want to think of your servers as cattle and be able to deploy new instances of your application fast. Do you really want to deploy your five new nodes on Amazon Web Services with a full Apache stack ready for production, then reconfigure your load balancers only to figure out that someone needs to go click your continuous integration tool to deploy the application to the new hosts? That one manual action someone forgets?

There's an easy approach to this process, and it comes with even more benefits. It's called *packaging*. When you package your artifacts as operating system (e.g., .deb or .rpm) packages, you can include that package in the list of packages to be deployed at installation time (via Kickstart or debootstrap). Similarly, when your configuration management tool (e.g., Puppet or Chef) provisions the computer, you can specify which version of the application you want to have deployed by default.

So, when you're designing how you want to deploy your application, think about deploying new instances or deploying to existing setups (or rather, upgrading your application). Doing so will make life so much easier when you want to deploy a new batch of servers.

> ***"When you package your artifacts as operating system (e.g., .deb or .rpm) packages, you can include that package in the list of packages to be deployed at installation time."***

Bring your code and user feedback closer together

Intuit founder **Scott Cook** is an advocate for a "rampant innovation culture" and allowing employees to do rapid, high-velocity experiments. Several years ago Intuit's Consumer Division took this to heart, and transformed the TurboTax website through **Continuous Delivery**.

The result?

They ran 165 experiments during the 3-month tax season. The website saw a **50% increase** in the conversion rate. The employees **loved it** because they saw their **ideas come to market**.

Gene Kim, Author and Researcher, IT Revolution Press discusses success through DevOps practices.

zend

“WHERE THE MONEY’S AT”: POSITIVE SOFTWARE EVOLUTION

AXEL FONTAINE

CEO at Boxfuse

Axel Fontaine is an entrepreneur, public speaker, and continuous delivery expert based in Munich, German. He specializes in continuous delivery and hates complexity with a passion. A regular speaker at technical conferences, Axel is the founder and project lead of Flyway—Database Migrations Made Easy. He currently works on Boxfuse, radically simplifying the deployment of applications by turning them into ultracompact, perfectly isolated, secure virtual machines within seconds and deploying them on any hypervisor with a single command.

Twitter | Website | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

Over the past 15 years of working with clients to improve their software systems, I found that one major challenge of building such systems is not an inability to write the code—that’s the easy part—but being able to understand the system and use that knowledge to develop the system in an efficient and cohesive way. That’s “where the money’s at.” It’s also where the going gets tough.

The complexity of a system is directly proportional to the number of moving parts. In software, the vast majority of these moving parts originate in configuration—a quick setting here, a convenient way to change something there. It all seems harmless, yet the complexity devil is already creeping in and quickly starts to create an explosion of potentially software-breaking combinations.

KEY LESSONS

1. **UNDERSTANDING SOFTWARE SYSTEMS IS PARAMOUNT TO THEIR POSITIVE EVOLUTION.**
2. **MOST OF THE “MOVING PARTS” IN A SOFTWARE SYSTEM ARE CONFIGURABLE ELEMENTS.**
3. **WHEN INTRODUCING CONTINUOUS DELIVERY, RELY ON HARD-CODING.**

Taking a closer look at this, I distinguish two types of configuration in the systems. The first type is the configuration that genuinely expresses the differences between the environments. After all, you wouldn’t want to connect to your development database in production, would you? The second type consists of the elements that may change some day. They have traditionally been made configurable to compensate for slow release cycles and to provide a faster backdoor to change the system in case of emergency. Although the former configuration type is a necessary evil, the latter is not. When introducing continuous delivery; rapid, reliable releases; and short cycles to my clients, I relegate this second type of configuration to the dark corners of history. In a world where a new production deployment is just one code commit away, new possibilities open. The backdoors of the past are eradicated and replaced by a simple and reliable alternative.

In the world of continuous delivery, hard-coded is the new configuration.

> ***“The complexity of a system is directly proportional to the number of moving parts. In software, the vast majority of these moving parts originate in configuration.”***

THE LAZY MAN'S GUIDE TO SOFTWARE SUCCESS

ANDI GUTMANS

CEO and Co-Founder of Zend

Andi Gutmans provides vision and direction for Zend, the leading provider of solutions for the rapid and Continuous Delivery of mobile and web applications. Zend's goal is to help companies deliver innovation faster. Since 1997, Andi and Zend have been key contributors to the evolution of PHP, which today powers more than 240 million websites and applications. Under Andi's leadership, Zend has partnered with IBM, Oracle, Microsoft, and other companies to advance enterprise PHP adoption. Andi holds a B.S. degree in computer science from the Technion, Israel Institute of Technology.

Twitter | Website | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

KEY LESSONS

1. **USE AUTOMATION TO CREATE BUILD ENVIRONMENTS QUICKLY AND EFFICIENTLY.**
2. **LET AUTOMATION DO THE HEAVY LIFTING IN YOUR DEVELOPMENT ENVIRONMENT.**

Years ago, I worked on F-15 fighter jet avionics simulations. The environment was complex, with more than 100 application binaries and hardware drivers written in C/C++ and other languages. When I joined the team, creating new development and production environments was a manual and tedious process. There was no easy way to set up a debug environment, and certain code changes required that many binaries be recompiled. This environment not only increased the potential for errors and inconsistencies in the code–test–debug process but also had a negative impact on software delivery velocity.

I quickly became frustrated by the lack of productivity and all the manual work involved. Fortunately, in my previous role as a systems administrator at a hosting provider, I had learned a lot about scripting and the environment. I like to work fast and dislike doing the same thing twice, so I was always building tools that would save time and steps and make my life easier. More time for breaks!

With this in mind, I rolled up my sleeves and decided to revamp the entire build and debug environment for the avionics simulation. I leveraged my automation skills to deliver a one-click build environment. One command was all I needed to rebuild the complete environment and install it into production. Every developer on the team had full control of his or her own environment, and no manual copying was involved. Getting an entire debug environment up and running was never more than one click away. A single code change could find its way into the test environment with one command. Looking back, my early work experience and a bias toward automation were what enabled me to have a big impact on productivity and quality for myself and my team.

In hindsight, I realize that I had adopted DevOps practices and implemented a Continuous Delivery process long before these terms existed—in an environment that was fundamentally the antithesis of Continuous Delivery. Today, with the methodology and all the great tools and best practices that exist to support DevOps and Continuous Delivery, there's no excuse not to embrace it fully. Go ahead: be a little lazy, let automation do the heavy lifting so your team can get the quality and productivity results they want, faster, without breaking a sweat.

> ***"In hindsight, I realize that I had adopted DevOps practices and implemented a Continuous Delivery process long before these terms existed—in an environment that was fundamentally the antithesis of Continuous Delivery."***

Continuous Delivery

Six Steps to Faster Releases without Breaking Anything

More Innovation • Better Quality • Earlier Feedback • Faster Releases

Start off on the right foot.

Read the White Paper ▸

zend

ARCHITECTURE AND CONTINUOUS DELIVERY

REBECCA PARSONS

Chief Technology Officer at ThoughtWorks

Dr. Rebecca Parsons is ThoughtWorks' CTO. She has more than 30 years' experience in leading the creation of large-scale distributed, services based applications, and the integration of disparate systems. Her interests include parallel and distributed computation, programming languages, domain specific languages, evolutionary architecture, genetic algorithms, and computational science. Rebecca received a BS in Computer Science and Economics from Bradley University, and both an MS and Ph.D. in Computer Science from Rice University.

Twitter | Blog

Download the full ebook: Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery

Reminiscing about past deployments, both good and bad, is a common occurrence when current and former operations people get together. Sadly, the focus is most often on the things that went right or wrong with getting the system live, not on the value (or lack thereof) of the functionality in the release. The bad times always seem to have involved that one little step that didn't get done properly, which actually isn't all that surprising, because these deployments tend to happen in the middle of the night.

Although the focus used to be on getting the instructions clear and right, automated deployment scripts won't miss a step, or mistype a configuration, or any of the other ways these manual deployment go horribly wrong. So, how can architecture help? In my experience, picking the right tools and knowing exactly what's running where are two of the big ways.

KEY LESSONS

1. **CHOOSE THE RIGHT TOOLS TO TAKE ADVANTAGE OF AUTOMATION.**
2. **KNOW WHAT'S RUNNING WHERE IN YOUR INFRASTRUCTURE.**
3. **USE STANDARD ENVIRONMENTS THAT HAVE KNOWN SOFTWARE VERSIONS AND CONFIGURATIONS.**

So many tools vendors seem to think that graphical interfaces are easy to use. They may be prettier, but they're much more difficult to automate. We've worked hard at times, even resorting to test automation tools, to automate deployment steps with such tools. Those making tool selections need to give high priority to the ease of deployment automation when making choices. Pretty interfaces might look nice and be easy to use in the middle of the afternoon, but things look very different at 2:00 a.m. Safe deployments are far easier to guarantee when you take humans out of the critical path.

Standard environments that have known versions of systems software and known configuration parameters also make deployments far easier. Guessing is not easy to do at 3:00 a.m., either. Being able to establish an environment to its known good state is critical. Scripted environment setup and tear-down mean you're always sure that the environment you're getting is the one you're expecting. Debugging issues on an unknown configuration brings far too many variables into the debugging equation. Ensuring environmental continuity allows troubleshooting to focus on the code that changed rather than worrying about whether it's just the environment that's problematic.

Deployment celebrations should be about the value of the new features, not joyous relief that nothing went horribly wrong. When these two things haven't been true, my life has been much harder.

> ***"Deployment celebrations should be about the value of the new features, not joyous relief that nothing went horribly wrong."***