Continuing the Journey

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

Download the Full eBook Now!

CONTINUING THE JOURNEY

You've made it. You have established continuous delivery and DevOps as your methods of choice for producing and releasing software. Now what? Here's one idea: do more.

In *Continuing the Journey*, six of the industry's finest continuous delivery thinkers offer advice on how to keep moving forward along the continuous delivery path. Some writers offer specific tips and techniques based on their own experiences. Others climb to the 40,000-foot level, offering unique, overarching takes on the wisdom they have acquired on their continuous delivery journeys.

Jeff Sussna, founder of boutique consulting firm Ingineering.IT, counsels that continuous delivery is a path, not a leap. It requires discipline and maturity. At its heart, he writes in his essay, continuous delivery is about small batch sizes and finding ways to make them ever smaller. "The key," he writes, "is to continually question your own assumptions about what's possible."

For enterprise technology architect and Scrum master Andrew Yochum, achieving agile delivery that is both confidently performed and consistently repeatable boils down to a three-word mantra: *Collaboration. Automation. Discipline.*

When his group used this mantra during a Software as a Service company project, the results were terrific. "We accomplished huge improvements overall," he writes. "As the team achieved agility, customers and management saw new features reliably delivered more quickly."

Matthias Marschall, chief technology officer of gutefrage.net GmbH, offers a simple formula for continuous delivery. When moving along that path, think of individual software feature releases the same way you think about bug fixes. In the end, he writes, there's not much difference.

When Marschall's company tried that, the results were admittedly painful at first. Some wanted to revert to the old method of releasing software upgrades twice a year. Instead, they persevered, concentrating on automating tests and release processes. It worked. Today, he writes, instead of wanting to sue over glitchy software, clients are loving their steady stream of new working features. Max Lincoln, a DevOps instigator at ThoughtWorks, compares continuous delivery to training for a marathon and identifies several parallels. Both are ambitious endeavors. Both require maintenance of strict routines. In either case, cutting yourself too much slack can be dangerous behavior.

"The key to success is a routine, fixing the root cause of all setbacks, whether it's better hydration for runners or better automation and communication when deploying software," Lincoln writes.

These and many other, similarly cogent lessons are offered in *Continuing the Journey*. The mini e-book is the sixth and final installment of a larger e book, ***Lessons from 29 DevOps Experts on the Best Way to Make the Transition to Continuous Delivery.*** The publication, sponsored by Zend, provides best practices and advice from DevOps industry leaders. For those who want to learn more about implementing continuous delivery, this e-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.

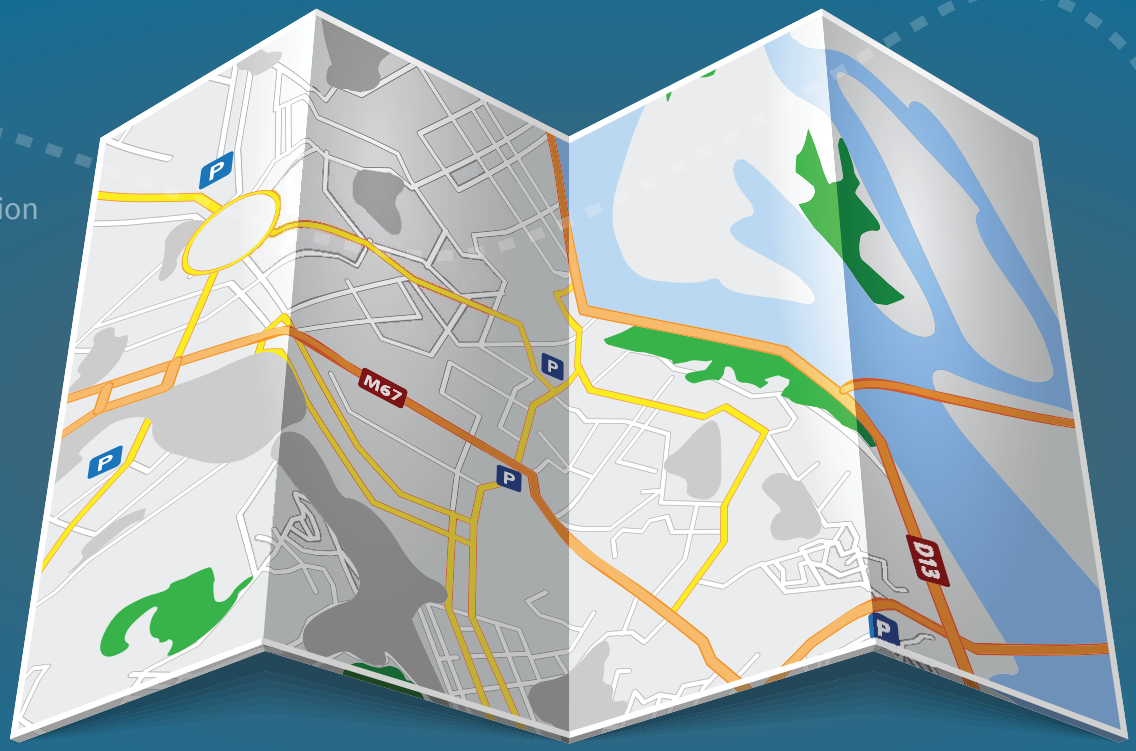
This e-book has been created for companies at virtually any stage of the journey toward Continuous Delivery. In the following pages, you'll find essays from software industry leaders whose experiences, insights, and solutions can make it a lot easier to get started, progress smoothly, and finish strong.

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

Continuing the Journey

MAKING ONE-DAY TURNAROUNDS POSSIBLE

JEFF SUSSNA

Founder and Principal
at Ingineering.IT

Jeff Sussna is founder and principal of Ingineering.IT, a boutique consulting firm that facilitates adaptive IT through teaching, coaching, and strategic design. Jeff has more than 20 years of IT experience and has led high-performance teams across the DevOps–quality assurance spectrum. He is a highly sought-after speaker and was recognized as a Top 50 Must-Read IT Blogger for 2012 and 2013 by *BizTech Magazine*. His interests focus on the intersection of development, operations, design, and business.

Twitter | Website | Blog

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

When my clients ask me how to adopt continuous delivery, I tell them, "Don't do continuous delivery; do *more* continuous delivery." Achieving true continuous delivery requires tremendous discipline and maturity. It also requires a shift in mindset that has to happen over time. Getting there is a path, not a jump.

At its heart, continuous delivery is about small batch sizes. You should always ask yourself how you can make your batch sizes smaller. For example, if you're building a software component with a front, middle, and back tier and it takes two days to build the whole thing, what happens if you build and release the back tier by itself? Because it's invisible without the other tiers, there's no harm in putting it into production alone. If you do, you've suddenly shrunk your batch size from two days to a few hours.

The key to following the continuous delivery path is to continually question your own assumptions about what's possible. I recently helped an agile development team test a new, stand-alone service. They were used to releasing every two weeks. One day, after I helped a developer validate a simple bug fix for the service, he said, "This is ready for release next week." I responded by asking, "Why not release it today? It doesn't affect anything else. We're not going to test it any more than we just did."

At the following morning's stand up, he announced that we'd fixed, tested, and released a bug fix, all in one day. Everyone's eyes got big! No one realized that one-day turnaround was even possible. Suddenly, they started thinking and talking about how to do it more often.

> ***"The key to following the continuous delivery path is to continually question your own assumptions about what's possible."***

KEY LESSONS

1. **ALWAYS ASK YOURSELF HOW YOU CAN MAKE YOUR BATCHES SMALLER.**
2. **CONTINUOUSLY QUESTION YOUR ASSUMPTIONS ABOUT WHAT'S POSSIBLE.**

SURVIVAL OF THE FITTEST: BEING RESPONSIVE TO CHANGE

EIKE THIENEMANN-DEHDE

Agile Product Lead at CoreMedia

Eike Thienemann-Dehde has worked on continuous delivery since 2012, when he was a product owner for tools and automation, support, and documentation efforts. He has worked in various roles, from software development to program management, over the past 16 years and is passionate about solving next-generation customer problems in an international and challenging environment.

Twitter | Website

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

I consider continuous delivery an ongoing journey. When you have mastered the principles of continuous integration and release automation, including deployment and quality assurance, and adopt your own practices, you quickly recognize that there will be always something to improve further.

After my company's successful agile transition and the migration to a distributed source code management system some years ago, we ramped up a dedicated team driven by the need to move quickly and incorporate support for automation tools like Chef and related processes directly into the product we were deploying in various operating environments, with their individual tool chains. When tackling change management challenges, it helps to create a community of practice made up of team delegates.

Being able to bootstrap continuous integration slave machines within minutes was the key to initially embracing the concept of Infrastructure as Code. In conjunction with the adoption of the Jenkins Job DSL plug in, it helped coordinate among all product development teams.

Baking build and test or production machines from well-defined Packer templates for any environment (e.g., Vagrant and vSphere or Amazon Web Services) increased reproducibility significantly and helped establish a fast in-house feedback loop so that we could spot issues such as commits that broke the deployment in some operating environments early.

> ***"When you have mastered the principles of continuous integration and release automation, and adopt your own practices, you quickly recognize that there will be always something to improve further."***

KEY LESSONS

1. **START WITH WEEKLY RELEASES WHILE PREPARING FOR MORE ON-DEMAND ROLLOUTS.**
2. **CONSIDER YOUR INITIAL CONTINUOUS DELIVERY TRANSITION SUCCESSFUL IF SUDDENLY YOUR PROCESS IS WAITING FOR YOUR PEOPLE.**
3. **LEARNING AND MOVING FAST IS A COMPANY DIFFERENTIATOR.**

SURVIVAL OF THE FITTEST: BEING RESPONSIVE TO CHANGE

EIKE THIENEMANN-DEHDE

Agile Product Lead at CoreMedia

Eike Thienemann-Dehde has worked on continuous delivery since 2012, when he was a product owner for tools and automation, support, and documentation efforts. He has worked in various roles, from software development to program management, over the past 16 years and is passionate about solving next-generation customer problems in an international and challenging environment.

Twitter | Website

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

So, to conquer the maturity level of automated tests, it helps to start with a dedicated code-freeze or release branch approach to avoid an initial moving target effect. Depending on the complexity of your regression tests in all supported operating environments—and especially the benefits versus cost of automated user interface tests versus manual tests—it might be feasible to start with weekly releases while preparing for more on-demand rollouts. You can consider your initial continuous delivery transition successful if suddenly your process is waiting for your people!

I suggest leveraging open source software infrastructure tools and innovating on a higher level by embracing a culture of "proudly found elsewhere" (preventing the "not invented here" syndrome). Collaborate in the open to give and get instant feedback. The impact on productivity and quality will create more freedom and options for experimentation.

Learning and moving fast can also be your company's differentiator, especially in a highly competitive environment. Always remember, "It is not the strongest of the species that survives, nor the most intelligent, but those most responsive to change" (Charles Darwin).

KEY LESSONS

1. **START WITH WEEKLY RELEASES WHILE PREPARING FOR MORE ON-DEMAND ROLLOUTS.**
2. **CONSIDER YOUR INITIAL CONTINUOUS DELIVERY TRANSITION SUCCESSFUL IF SUDDENLY YOUR PROCESS IS WAITING FOR YOUR PEOPLE.**
3. **LEARNING AND MOVING FAST IS A COMPANY DIFFERENTIATOR.**

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

COLLABORATION. AUTOMATION. DISCIPLINE.

ANDREW YOCHUM

Enterprise Technology Architect

Andrew Yochum strategically leads technology in digital agencies, finance, and start ups. With a background in highly scalable Web development and a passion for data, Andrew sees his work as an art and a science. His current focus is on emerging technologies in cloud computing and big data. He guides the adoption of agile methodologies as a Scrum Master and Coach.

Twitter | Website | Blog

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

I'm a big advocate of agile methodologies, having experienced vast improvements on development teams over my career. Many teams adopt agile practices, like Scrum, DevOps, and continuous delivery, to bring agility to the business. Each of these practices plays a role in a team's realization of the Agile Manifesto's principles, which should be interwoven to achieve even greater results. The key to achieving agility is being able to deliver confidently in a repeatable fashion—every time. But, how?

Collaboration. Automation. Discipline.

One team I lead for a Software as a Service company had multiple, interdependent applications supporting consumers, business owners, customer service, and internal administration. When I joined, the applications were deployed manually to physical servers. Deployments were time consuming, requiring downtime and carefully executed processes from systems administrators who knew little about the applications themselves. Troubleshooting was tedious when things went wrong. Testing the deployment was equally challenging. Often, the result was failed deployments and downtime.

I knew it was time to bring agile methodologies to the team, fostering collaboration with Scrum and DevOps. The goal: add automation with automated testing, builds, and deployment, and through these elements, bring a disciplined methodology to achieve agility.

> ***"The benefits were clear right away. Committing code kicked off a build, deployment, and automated tests. The results of the build showed up in the developers' inbox and the integrated development environment."***

KEY LESSONS

1. FOSTER COLLABORATION AMONG DEVELOPERS, QA TEAM MEMBERS, AND SYSTEMS ADMINISTRATORS.
2. ADD AUTOMATION WITH AUTOMATED TESTING, BUILDS, AND DEPLOYMENT.
3. INTEGRATE DATABASE SCHEMAS INTO THE AUTOMATION PROCESS.

COLLABORATION. AUTOMATION. DISCIPLINE.

ANDREW YOCHUM

Enterprise Technology Architect

Andrew Yochum strategically leads technology in digital agencies, finance, and start ups. With a background in highly scalable Web development and a passion for data, Andrew sees his work as an art and a science. His current focus is on emerging technologies in cloud computing and big data. He guides the adoption of agile methodologies as a Scrum Master and Coach.

Twitter | Website | Blog

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

We took things to the next level by adding a continuous build and test process internally on a virtualized infrastructure that our systems administrators had set up. The benefits were clear right away. Committing code kicked off a build, deployment, and automated tests. The results of the build showed up in the developers' inbox and the integrated development environment. Over time, the team added unit and functional tests to ensure that everything worked as expected. Our quality assurance (QA) team evolved from manual testing to automated testing. They became integrated with the development team, writing tests and code as development occurred. We set up a dashboard for me and the executive team to review the state of the development at a glance.

The next big step was adding Infrastructure as a Service cloud servers, using the same deployment process we used internally. Because our systems administrators had become intimately familiar through the initial automation, the leap from physical to cloud servers was reasonably easy.

But, there was a snag. Over time, as new features were developed that required changes to the database schemas, tests failed. Developers had been manually propagating changes to the database schemas ahead of their commits and builds: a big no-no! The schemas hadn't been integrated as part of the automation process. We took a deep dive into it during our Scrum sprint retrospective, adding user stories to our backlog to address the issue. The team collaborated to apply tools and automate the same methods we used in other parts of the development process.

We accomplished huge improvements overall. The collaboration among developers, QA, and systems administrators fostered greater shared knowledge and improved morale. With full end-to-end automation in place, deployment times went from hours to minutes, with few failures. The team's adoption of agile methods brought understanding and appreciation for discipline, which allowed them to better their craft. As the team achieved agility, customers and management saw new features reliably delivered more quickly.

KEY LESSONS

1. **FOSTER COLLABORATION AMONG DEVELOPERS, QA TEAM MEMBERS, AND SYSTEMS ADMINISTRATORS.**
2. **ADD AUTOMATION WITH AUTOMATED TESTING, BUILDS, AND DEPLOYMENT.**
3. **INTEGRATE DATABASE SCHEMAS INTO THE AUTOMATION PROCESS.**

DON'T LOOK BACK: PERSEVERING THROUGH THE TRANSITION TO CONTINUOUS DELIVERY

MATTHIAS MARSCHALL

CTO of gutefrage.net GmbH

Matthias Marschall is a software engineer "made in Germany." His four children make sure that he feels comfortable in lively environments and stays in control of chaotic situations. A lean and agile engineering lead, he's passionate about continuous delivery, infrastructure automation, and all things DevOps. Matthias is CTO at gutefrage.net group, helping to run Germany's biggest Q&A site among other high-traffic sites. He's the author of the *Chef Infrastructure Automation Cookbook*.

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. **CONSIDER RELEASING INDIVIDUAL FEATURES LIKE CRITICAL BUG FIXES.**
2. **CONCENTRATE ON AUTOMATING YOUR TESTS AND RELEASE PROCESS TO INCREASE QUALITY.**

If you're working in an environment where you do infrequent, big releases, you might find yourself in a situation where the pressure to deliver faster becomes unbearable. When I was building an Internet platform for the construction industry, it was best practice to do a big release twice a year. Unfortunately, each such release was a big disaster followed by weeks of fixing critical bugs and customers threatening to sue us. We knew that the huge amount of code changes forming a big release was an issue, so we cut down our release cycles to bimonthly.

Although the releases shrank, the shortened release cycles put a lot of additional pressure on us. Not rethinking our process, we simply had less time to do all the upfront analysis and manual regression testing we considered mandatory at that time. Of course, we also had to make critical bug fix releases all the time, without a lot upfront analysis and definitely without manual regression tests of the whole product.

One day, we stood together and wondered how we could solve our troubles. A colleague raised the key question, "What's the difference between a feature and a critical bug fix? Why do we force ourselves to do a lot of upfront analysis and manual regression testing for features, while we're releasing critical bug fixes at any time?"

The question was spot on. Eventually, there was no real difference. Why not release individual features like critical bug fixes? We tried it.

In the beginning, it hurt—badly. We were even thinking about going back to bigger releases. But we persevered. Instead of going back, we concentrated on automating our tests and release process to increase quality. Slowly, we were seeing the benefits of small releases: they became a nonissue. And instead of suing us, our clients were happy to get new, working features much more quickly than before!

The key in such a transition to continuous delivery is to expect things to get worse before you'll be able to make them better. But if you persevere, you'll be able to make things so much better that you'll never want to look back.

> ***"The key in such a transition to continuous delivery is to expect things to get worse before you'll be able to make them better."***

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

TRAINING FOR THE WIN: TRANSITIONING TO CONTINUOUS DELIVERY

MAX LINCOLN

DevOps Instigator at ThoughtWorks

Max Lincoln works in the Continuous Delivery group at ThoughtWorks. He passionately promotes the use of continuous delivery and DevOps to achieve technical success, lean startup to drive business success, and user experience to deliver enjoyable products. He is a frequent contributor to open source cloud and quality assurance projects.

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. **TRANSITION TO CONTINUOUS DELIVERY REQUIRES A SUSTAINABLE PACE.**
2. **DEVELOP A ROUTINE, THEN TRY NOT TO BREAK IT.**
3. **LEARN HOW TO DEAL WITH SETBACKS.**

Continuous delivery is often compared with physical exercise. It's difficult at first but becomes easier as you develop skills and form habits. When I first heard this analogy, I thought of weight training. I should have thought of training for a marathon.

I've noticed techniques used by people who are training for a marathon that are similar to the techniques used for successful transitions to continuous delivery. This isn't surprising: both continuous delivery and running a marathon are ambitious goals that require a sustainable pace rather than heroic sprints. Both share three techniques for ensuring success: continuous improvement, forming positive habits, and dealing with setbacks.

The first week of a typical 16-week marathon training program looks more like a training regime to run a better 3.1 mile race than one for a 26.2 mile race. The lengths of the runs start short but increase each week. This is the surest way to reach an ambitious goal: as a series of achievable improvements starting from your current state. A sudden leap into enterprise-wide continuous delivery may be just as foolish as attempting 20 miles on your first run.

The same training program involves four runs per week, always on the same days. Setting a routine minimizes the overhead of planning and maximizes the opportunities to learn which changes are working. Runners learn about diet and hydration, while we learn about topics like automation and feature toggles. The routine also ensures that there is never too big a gap. I would rather deploy at least weekly, then deploy daily but occasionally have a multi-week deployment freeze. Breaking the routine is risky.

The biggest lesson is how to deal with setbacks. Our resolve toward continuous delivery is always put to the test after a rough deployment. Someone usually suggests that our problem was deploying too often. Runners face this situation, as well. It seems logical to think, "I should run less because I am too fatigued to finish." This is a dangerous idea. Running fewer total miles compromises your goal; running more miles per run risks further fatigue or injuries. Even worse, it does not address the root cause. The key to success is a routine, fixing the root cause of all setbacks, whether it's better hydration for runners or better automation and communication when deploying software.

> ***"The biggest lesson is how to deal with setbacks. Our resolve toward continuous delivery is always put to the test after a rough deployment."***

CONTINUOUS DELIVERY VS. DELIVERING CONTINUOUSLY

MICHAEL HÜTTERMANN

CEO of huettermann.net

Java Champion Michael Hüttermann is a freelance delivery engineer and expert in DevOps, continuous delivery, and software configuration management/ application life-cycle management (ALM). He wrote some of the first books on DevOps (*DevOps for Developers*, 2012) and agile ALM (*Agile ALM*, 2011).

Twitter | Website

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

Continuous delivery is a new mindset for development and delivery of software to add value to the system and satisfy the customer. It is also possible to "deliver continuously" (for some people, delivering once a year is "continuously," too) just by throwing changes to production randomly, with bad quality. Thus, a systematic process of staging of software is crucial, along with introducing collaboration between development and operations to reduce cycle time and fulfill holistic business goals, often called *key performance indicators*. Without DevOps, there's no continuous delivery.

Whereas DevOps deals with collaboration between development and operations, continuous delivery focuses on the concept of bringing changes to production continuously and with minimized cycle time—and being able to do so frequently. Particularly crucial elements for implementing DevOps are:

- Aligning development and operations with the same goals, which are in turn aligned with overall business goals;
- Fostering a mind shift and introducing slack time and room for experimenting; a bit of firefighting is okay—you learn a lot through firefighting—but time for innovation is even better;
- Aligning processes and tools and finding the optimal tradeoff for configuration management;
- Finding the right balance of manual work and automated processes (be aware of the irony as well as the paradox of automation);
- Aligning the changes in the delivery pipeline with the business, not the technique;
- Realizing that the delivery pipeline is not a one-direction fire-and-forget tube but rather an integrated life cycle;
- Fostering the mindset that a single change may result in a *potential* release and *may* start the whole process; and
- Finding semantic containers for your release, including all artifact types, that are versioned and executable.

KEY LESSONS

1. **FIND THE RIGHT BALANCE OF MANUAL WORK AND AUTOMATED PROCESSES.**
2. **ALIGN THE CHANGES IN THE DELIVERY PIPELINE WITH THE BUSINESS, NOT THE TECHNIQUE.**
3. **FOSTER THE MINDSET THAT A SINGLE CHANGE MAY RESULT IN A *POTENTIAL* RELEASE.**

> ***"Without DevOps, there's no continuous delivery."***

CONTINUOUS DELIVERY VS. DELIVERING CONTINUOUSLY

MICHAEL HÜTTERMANN

CEO of huettermann.net

Java Champion Michael Hüttermann is a freelance delivery engineer and expert in DevOps, continuous delivery, and software configuration management/ application life-cycle management (ALM). He wrote some of the first books on DevOps (*DevOps for Developers*, 2012) and agile ALM (*Agile ALM*, 2011).

Twitter | Website

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

Continuous delivery spans the software life cycle and is based on *continuous integration* (that is, checking changes into version control multiple times a day, with the constraint that developers can check out current states of the software at any time without having any build errors locally afterwards), *continuous deployment* (that is, the frequent deployment of changes to target environments), and *continuous inspection* (that is, the inspect-and-adapt pattern to keep up the internal and external quality of the software).

Continuous delivery is not necessarily part of a DevOps initiative. In other words, if you implement continuous delivery, you'll most likely have to implement some sort of DevOps, too. I wish you much fun and success with your continuous delivery initiative—and don't forget to shape your DevOps approach.

KEY LESSONS

1. **FIND THE RIGHT BALANCE OF MANUAL WORK AND AUTOMATED PROCESSES.**
2. **ALIGN THE CHANGES IN THE DELIVERY PIPELINE WITH THE BUSINESS, NOT THE TECHNIQUE.**
3. **FOSTER THE MINDSET THAT A SINGLE CHANGE MAY RESULT IN A *POTENTIAL* RELEASE.**