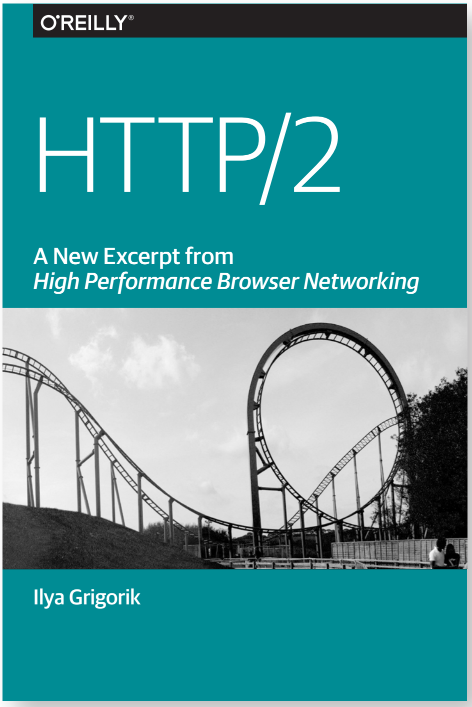
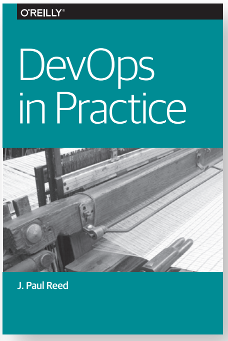
DevOps in Practice

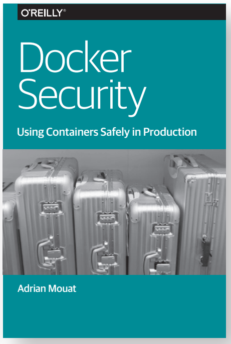
J. Paul Reed

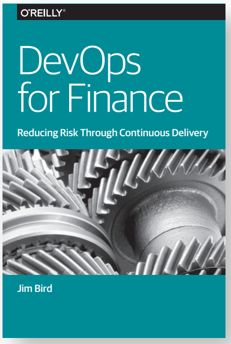
Short. Smart.
Seriously useful.

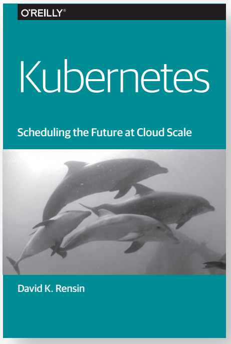
Free ebooks and reports from O'Reilly
at **oreil.ly/devops**


O'REILLY®
HTTP/2
A New Excerpt from
High Performance Browser Networking
Ilya Grigorik


O'REILLY®
DevOps in Practice
J. Paul Reed


O'REILLY®
Docker Security
Using Containers Safely in Production
Adrian Mouat


O'REILLY®
DevOps for Finance
Reducing Risk Through Continuous Delivery
Jim Bird


O'REILLY®
Kubernetes
Scheduling the Future at Cloud Scale
David K. Rensin

Get even more insights from industry experts
and stay current with the latest developments in
web operations, DevOps, and web performance
with free ebooks and reports from O'Reilly.

O'REILLY®

©2016 O'Reilly Media, Inc. The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. D1710

DevOps in Practice

J. Paul Reed

Beijing • Boston • Farnham • Sebastopol • Tokyo

DevOps in Practice

by J. Paul Reed

Copyright © 2014 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editor: Brian Anderson

September 2014: First Edition

Revision History for the First Edition

2014-08-26: First Release
2015-03-24: Second Release
2015-12-11: Third Release

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *DevOps in Practice,* the cover image, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the author(s) have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author(s) disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-491-91306-2

[LSI]

Table of Contents

Introduction. ix

Nordstrom. 1
A Campout at the Colo 1
The Event™ 2
Reflections on the Journey 6
A "Have-Coffee" Culture 12
Flipping On the "DevOps Bit" 14

Texas.gov. 17
A Public/Private Partnership 17
"The Only Way to Do It" 19
Continuous…Security? 20
Lessons from the Lone Star State 21
A Unicorn with a Cowboy Hat 23

Acknowledgments. 25

Introduction

Practice makes perfect.

It's an adage we hear from an early age, usually around the time we start learning to tie our shoes, ride a bike, or play an instrument. As DevOps gets ready to celebrate its fifth birthday,[1] DevOps practitioners and the movement itself are starting to hear this familiar phrase.

It can be easy to forget that deliberately practicing a skill to hone and make our own is a time-honored technique. It can be hard to find the time for the necessary focused practice, as work, family, projects, and circumstance all impact our ability to find the time and space to do so. It can also be difficult when that "we" is a large organization, comprised of many different facets and personalities, with various motivations and incentives floating about.

Contained herein are two stories of organizations figuring out what "DevOps" means to them. Based on a series of interviews with people at different levels of the organization and working on various teams, we get to see them undertake the tasks of discovering what DevOps means in the context of their own organizational cultures. We also get to see them wrestle with how it looks functionally within their companies, expressed in the structure of their teams, and the path code takes from commit to customer. The characters in our story may surprise you, as they're not in the list of companies that generally come to mind when the phrase "DevOps posterchildren" is uttered.

---

[1] Patrick Debois, widely considered to be the father of the word "DevOps," held the first DevOps Days in Ghent, Belgium, in October 2009.

Much is made of the fact that DevOps is about both "tools and culture! Tools *and* culture!" But as we shall see, while tools and culture are both important, perhaps the most important aspect to take note of is the journey itself.

What Is DevOps?

New to DevOps? Welcome! This book delves into the details of how two different organizations are working to become more "DevOps-like"; if you're unfamiliar with DevOps or would like to read more, we recommend:

- "10+ Deploys Per Day: Dev and Ops Cooperation at Flickr", Velocity 2009 presentation by John Allspaw and Paul Hammond
- The Phoenix Project, by Gene Kim, Kevin Behr, and George Spafford
- *Building a DevOps Culture* (O'Reilly), free ebook by Mandi Walls

The organizations profiled also employ infrastructure as code and continuous delivery to accomplish their goals; these pieces give a more in-depth treatment of the fundamentals of those topics:

- Adam Jacob's chapter on "Infrastructure as Code" from *Web Operations* (O'Reilly), by John Allspaw and Jesse Robbins
- *Test-Driven Infrastructure with Chef, 2nd Edition*, by Stephen Nelson-Smith
- *Continuous Delivery*, by Jez Humble
- *Lean Enterprise*, by Jez Humble, Barry O'Reilly, and Joanne Molesky

Nordstrom

A Campout at the Colo

Rob Cummings hated deployments.

The year was 2004. Cummings was an operations engineer working on the team that supported *nordstrom.com*. After a bit of prodding, Cummings chuckles and admits that it is a bit odd. After all, it's not like it snuck up on him; getting code pushed out to production was a big part of any operations engineer's job in 2004.

By that point, large-scale retail websites were no longer a shiny new concept. The brick and mortars had started developing their online identities during the first dot-com boom of the early 2000s, as it became clear to a number of industries—sometimes viscerally—that this "World Wide Web thing" was not a fad and was very much not going away.

"It was a traditional environment," Cummings recalls: separate development and operations teams, operations peppered with myriad "throw-away shell scripts," anywhere from a few days to several weeks to provision compute resources for development and testing. The web applications were monoliths, deployed with a heavyweight process to facilitate the required heavy lifting, all frosted over with the amount of pageantry such a system implies. For its time, the team was doing a good job of meeting the business' needs; remember that new major versions of the browsers customers used to get to *nordstrom.com* were only released once a year back then. For the Nordstrom website, the company performed its major deployments about once a quarter or so, in a process they called "site-downs."

"I hated, *hated* site-downs, so much so I think I blocked most of them out of my mind," Cummings muses. Out of an abundance of caution, each site-down involved Nordstrom's operations engineers driving to the colocation facility to perform the deployment. Cummings recalls the amount of manual work to complete the deployment: "We'd just sit there all night and work on it until…it worked." If his description conjures up images of 2 am hacking sessions in the NOC fueled by pizza and Mountain Dew that so many of us lived through back then, think again: "We didn't even have that! The colo was out in the middle of nowhere, and it was the middle of the night. If you didn't bring any food, you weren't getting any food."

"It rarely went well," Cummings says. "But that was part of being in operations back then: you'd just figure it out."

Life in website development wasn't particularly easier, recalls Nordstrom's Courtney Kissler, who worked on the Website Engineering team. Even though a few years had passed, it was still the era of site-downs and heavyweight deployments; developers were trying to keep pace with the increased rate of change experienced in the latter half of the 2000s, working on features ever more furiously and trying to get them integrated and shipped ever more rapidly. "We had all these opposing forces; we had this long-standing throw-it-over-the-wall mentality, where the way we did things was just to give it to Rob's team and they'd figure it out." Kissler remembers a number of occasions where the operations team had to take point for figuring out why a feature was underperforming (or in worse cases, impacting a more noticeable part of the website). "Teams were staying up all night to get things going, and it was a pretty big morale hit." That made the job of developing features tough enough, but the real issue, Kissler said, which wasn't even clear to the team at the time: "Frankly, we were causing production outages and service interruptions" due to moving so quickly, yet so haphazardly. This translated into inconsistent releases, where about 30% of the time, features had to be turned dark after they'd gone live.

It was a tough period for those supporting the online customer experience, no matter which side of the site you were on.

The Event™

Organizations of a certain size and with a certain amount of history embed events within their consciousness. As stories of The Event™

are passed down from manager to individual contributor and spread among the new hires, team by team, they become part of the company lore. They're given proper names. They serve as warnings to others, that "Here, there be dragons" and one team, many moons ago, wrestled that dragon. The goal is to help spread knowledge of what made the organization succeed (or what deficiency made it a bitter loss). The most pervasive Events become part of the institutional consciousness precisely because they contextualize the journey the organization and its people are themselves on today; it's the thing that spurred them to get on the road in the first place.

For Cummings, this event was the website falling over on one of the company's highest-volume days. "It was traditionally a very festive day." For both website developers and operators, this was the day to let their work from the past months shine, Cummings said: "The feeling was always 'Oh, we will watch all of this traffic coming to the site, isn't this great?!' But it wasn't great." In a pattern that will be familiar to anyone who's worked on large-scale sites "The website came crashing down under load. It had *never* done that before." For Cummings and his crew, the next couple of days were long, as it became an all-hands-on-deck problem. "It was really difficult for us, because it's one of our highest visibility days and our customers were having a terrible experience."

After getting the situation under control, Nordstrom reacted as most organizations do: put ointment on the still-stinging wound. For them, that ointment was spinning up a complete performance-testing environment. Code, it was decreed, now had an additional gauntlet of load tests to face before making its way into production. It was a reasonable first stab at the problem, Cummings recalls, and "was totally catching problems." But the solution quickly spun out of control: "Because we didn't significantly change how we deployed not only our servers, but our code, it took awhile to get that environment up and running. And so other teams wanted their own complete performance environment, just to test their portions of the code. It was all about environments, more environments," Cummings said.

The solution created a big hit to the already-impacted operations team. Cummings notes that other teams often perceived Operations as the bottleneck, even though he had worked through the numbers and could show they weren't. But with the blooming of all of these environments, Cummings knew this process, while wellintentioned, just wasn't going to scale. It was this event (and its solution), only observable in the crispness of hindsight, that started Nordstrom on its continuous delivery and DevOps journey.

Enter: The "DevOps Team"

That journey started like it often does in larger organizations: with the creation of a "DevOps team," though they didn't name it that. Such teams are topics of hot conversation within the DevOps community—do they work, aren't they just creating another siloed team, how can they possibly help with the necessary cultural changes—but for Nordstrom, getting started on the journey trumped debating the "conventional wisdom" (if such a thing exists in a movement on the heels of celebrating its fifth birthday[1]) about what the "thought leaders" think it has to look like. Doug Ireton, a Nordstrom infrastructure engineer, was one of the first engineers to join this new team. "The first thing we tried to do was pick something small, so we picked host files," which were used extensively within the infrastructure at the time, Ireton said.

The team had settled on Chef as their tool of choice to start implementing "infrastructure as code," one of the pillars of DevOps practice. Server host files seemed like a good idea to Cummings too, who was now leading the team as engineering manager. Working on a heavily-used, production-required element within their infrastructure would give them real-world experience not only learning the automation tool, but also figuring out which workflow was best for them and their own organizational and technological requirements. Plus, it was a small, easy-to-identify scope of work, not too prone to so-called bike-shedding.[2]

"It sounds really good in theory," Cummings said. "Bad idea, it turns out." The pervasive nature of the host files are precisely what made it difficult for the team to pull this one aspect of their entire infrastructure under the control of the new initiative. "When you try to bring in your legacy snowflakes, it's bad. And we're talking hundreds and hundreds of snowflakes, across about twenty environments, and that singular file was different in ways we weren't expecting. That made the implementation extremely complex, just for managing this one file," Cummings explains. The idea of trying to shoe-horn a singular, widely-used element of their infrastructure hadn't worked and had frustrated the team to boot.

---

[1] DevOps Days Belgium 2014

[2] Also known as Parkinson's law of triviality; C. Northcote Parkinson's argued in 1957 that organizations give disproportionate weight to trivial issues; its use was popularized in the software industry in 1999 by the BSD community.

Try, Try Again

After struggling so much to put something seemingly so simple under configuration management, Cummings realized this approach wasn't going to result in success. He realized the team was still figuring out the fundamentals of not only a new technology, but a new way of modeling and interacting with the systems that comprised their infrastructure. Fortuitously, another project that was critical to the business presented itself: the "payment store processor." Servers at each Nordstrom location ran a legacy application that needed to be virtualized. Experience had shown that creating this server was a manual process that took about 18 hours. At 200 stores, the staggering scope of the task, were they to do it manually, became clear. "We decided to totally pivot our approach and tackle this: we were going to build this server end-to-end, all with Chef," Cummings decided. "But it was Windows Server 2003, so it was the hardest thing you can imagine to automate."

To make the project successful, Cummings roped in engineers from application development, the database team, and other layers of the stack. Despite having a much larger scope and being a more difficult technological problem (and one not even related to the website!), it increased the team's focus. After a few weeks locked in a conference room working together, the team was able to build these "store processor" servers in four hours, in a fully-automated, repeatable fasion. The time-savings to the business and the sanity-savings to the engineers who would be conscripted to do the 18 hours of manual work, in shifts, were obvious. It also gave Cummings' team confidence that the tool really could perform in an odd environment, with a nonstandard use case on an older OS and foreign platform, yet serve a very real business need, and do it end-to-end. "In some ways, this was the hardest thing we could've picked," Ireton notes. "But it really made our team gel."

Reflections on the Journey

The journey is more important than the destination, or so the saying goes. The sentiment could not be truer of an organization's DevOps transformation. In discussions with engineers on both sides of Nordstrom's technology organization, a number of lessons were highlighted on their path toward an operations environment based on infrastructure as code and development teams taking more ownership and moving toward continuous delivery.

The First Project Is Exactly That: Your First Project

Counter to Apollo 13 Flight Director Gene Kranz's famous quotation, when embarking upon a journey to transform company culture and technological practice, there are a lot of moving parts: failure is an option. As Nordstrom's first "DevOps project" illustrates, it's important to remember that the first attempt at working on a concrete project to incorporate these new ideas into your organization may not result in a completed, fully functional continuous delivery pipeline, backed by your next-generation configuration management tool of choice.

But that doesn't mean the experience is worthless. In the throes of stumbling around, your teams can gain a lot of valuable insight about the intricacies of the technology stack they've chosen, the nuances of the workflows around those tools, and the organization's unique touch points that will be required to make your teams successful. It is also likely to reveal assumptions about your infrastructure that will be sobering to your team, like just how many "server snowflakes" have accumulated to create that snow drift everyone has avoided shoveling.

In the end, it's all about the framing of the initial project and how the team grapples with the outcome, whatever it may be. Despite the initial stumble with managing something "simple," like host files, across the entire infrastructure, Cummings considers the most valuable part of the successful "payment store processor" automation project to be how it kickstarted their journey, even though it wasn't their first attempt: "By having people from those silos all working together, it built a lot of empathy. And we were finally able to get moving."

Change Is a Difficult Process

That change is a difficult process is not a revelation to anyone who's grappled with it in a personal or organizational context. What may be surprising is the specific ways in which the difficulty of change presents itself when working toward adopting a more DevOps-like culture: "You're potentially asking these senior engineers who are experts in a specific ecosystem to move away from that, and sort-of start over," Ireston said. That can be a tough sell, especially when a team is already accountable for keeping the business' lights on.

In Nordstrom's case, the experience of looking deliberately at the team's problems and the change required to move them toward continuous delivery involved looking at how teams were structured: "We're trying to make a cultural change to a model where development teams own their app and they run it in prod and they care about it," Ireton said. Nordstrom had previously structured their technology operations around "shared service teams," like QA or operations. To get teams to feel like they actually had ownership over their applications, that meant those previously "shared" roles needed to be embedded, as appropriate, within the application teams. The idea of "shared service" teams also had to shift from the concept of engineers providing a service, like QA running tests, to engineers developing and supporting a service to *provide a service*, such as integration tests or virtual machine deployment, which application teams could then use as they needed. One might even call it a "Service as a Service" model. Change is often also measured, and Nordstrom continues to support the shared services teams for development teams that are still evaluating exactly what a move to an embedded, "full stack" team structure would mean for them.

Another insight Ireton noticed was how certain technology stacks can assist or hinder these sorts of transformations: in Nordstrom's case, the way Microsoft's technology stack interacted with itself tended to be very siloed. Ireton was, in fact, originally hired because of his deep experience with Microsoft's Windows Server Group Policy. "The ecosystem was such that you had the area you knew and were responsible for, but if you needed to go beyond that, you had to find an engineer that was 'Microsoft certified' for that area," Ireton explained. That's a very different model from the "full-stack" teams tasked with direct involvement with all aspects of their application's needs that Nordstrom wanted to move toward.

Prioritization Is the Elephant in the Room

Often, "the business" plays the role of product owner and drives the prioritization of work. But this simplistic model can have disastrous effects if you're trying to introduce an initiative such as continuous delivery. "At the time, for the business, all they wanted was more features," Kissler recalls. "We had to use data showing the sometimes-difficult outcomes, the system outages, and the missed feature commitments to illustrate that we needed to focus on our technical debt." Kissler said one of the big questions that started being asked was "Why aren't we focusing on these production issues, and why does it take so long to get a feature into production?"

Repeatedly asking these question spurred discussions that allowed the development teams to successfully get a notable portion of each release cycle dedicated to not only paying down technical debt, but specifically for working on developing a continuous delivery pipeline and migrating applications to use it. "After we got someone who had tremendous credibility on the business side who was able to surface those problems in those discussions, it really shifted; then it wasn't a technology story about this whiz-bang continuous delivery thing, it was a story about how the product we were delivering wasn't meeting their needs," Kissler explained. "That story resonated."

Don't Tie the Initiative to Individuals

Once Kissler found her counterpart on the business side, keeping the journey going became easier. But Kissler has a warning about how it could have played out: "You should never create process or confidence around an individual, because that person is not going to be around forever." In Nordstrom's case, her business team counterpart moved, and the initiative stalled. Without someone in the business meetings keeping the torch burning and explaining how the project was doing, the initiative started to backslide, Kissler recalls: "Things got pretty rocky for a bit; I wouldn't say they fell apart, but we certainly had some bumpiness. The organization wanted to return to its previous state."

Interestingly enough, this prompted Kissler and Cummings to actually shift from a story focused on business needs back toward telling a technology story, now that the business had a taste for what was possible. Ultimately, the situation was a mere speedbump on the road of change, but given other circumstances, the departure of key people driving the change can bring an abrupt halt to the journey, sometimes in ways that aren't immediately visible.

Savings Versus Speed

Many enterprises approach IT with the goal of cutting costs wherever possible. This makes sense in a model where the IT department is accounted for as a cost center. But Nordstrom, like many enterprises, realized that isn't the path to the desired results, especially when it comes to their online presence: "We, as a technology organization, had been optimizing for cost. A couple of years ago, we realized we need to be about speed-to-value, especially in our customer-facing areas," Kissler recalls. "It was challenging to get everyone to make that transition."

As an example, Kissler managed the rollout of the first in-store mobile application to handle sale transactions. The application took six months to develop, and included a lot of features that ended up addressing use cases that customers in stores didn't care about; those features ended up being thrown away. Even the platform—iOS versus Windows tablet—changed. But Kissler wouldn't have done it differently: "In the spirit of speed-to-market, that was our fastest path." After that first successful project, even though its scope was larger than an initial project Kissler would scope and undertake now, it translated into great strides on the journey: "Once we had that, people were like 'let's do more of this.' Let's figure out how we can test and learn, pivot, and fail fast, and create this environment where we can do more of this."

Determine the Flow of Value

One technique that kept coming up in discussions with Nordstrom's application development and operations teams was the process of value stream mapping. Value stream maps attempt to model work that flows through a system. It captures where handoffs occur and how various teams turn raw materials, such as commits, into finished products the business can sell to a customer (or utilize, like an e-commerce website). It is especially good at illustrating the delta between "work as perceived" and "work as performed."[3] Cummings describes a quintessential example of this situation: "Teams would say 'Don't worry about us; our piece is automated.' But then you'd go and look at why things took so long, and the 'automation' was an engineer following a Word document to process something, or some team was undoing work the team before it had done."

Kissler echoes "I need to be able to deliver faster; I want continuous flow with as close to zero waste as possible. So when we needed data to make the case for our business teams, it made it hard for people to dispute it when we surfaced the waste and made it visible." The result of these value flow exercises has revealed so much actionable data that it's still an on-going process for Nordstrom as an organization to work through how to holistically address it: "With the release teams, the development teams, the ops teams, and the QA teams optimizing locally for such a long time, they were hurting the whole system; we're still working through detangling that," Cummings said.

Beware How and Where You Pour Gasoline

When organizations decide to embark upon a journey, they have often committed to make the required investment. But Kissler warns that this investment must be tempered by the organization's ability to absorb the influx of resources, and a systems-view must be taken to ensure the extra resources aren't just being converted to waste: "We tripled our investment in this area, but it caused a problem where we were producing so much, not all the teams could handle it yet. We had to deal with what we called the 'burst-pipe' problem."

This is a common issue where the organization has decided to commit itself to continuous delivery, but the investment is spread unevenly across the organization or teams start to make heavy use of the so-called continuous delivery pipeline while it's still "under construction." Operations and other support teams are used to the metaphor of a life of "fighting fires." When those teams have only ever been given the resources to beat back fires to hidden but still smoldering embers, dumping gasoline on the situation—in the form of increased budget and hiring capability—can cause them to explode into raging fires again. In Nordstrom's case, this required focused investment on the operations side, to reduce time required to build and deploy infrastructure and increase consistency. Had they not undertaken this, the increased investment in development would have hit a major clog in the pipe when it came time to deploy to production. A similar situation applies to the quality assurance part of the pipeline. These clogs can contribute to the "burst pipeline" problem Kissler described.

---

[3] A concept described further by Sydner Dekker, as a model used in describing events during postmortem analysis.

"There's No Place Like Home"

Oftentimes, part of an organization's cost-cutting strategy involves outsourcing various IT or development functions to other companies. This can be a big impediment to a shift to continuous delivery since the model for outsourced teams typically has them delivering their artifacts and moving on. This makes it difficult to inculcate a DevOps-focused culture, where teams are responsible for their work via the operation and care-and-feeding of their application. Even though Nordstrom worked with external partners initially on developing some of their new applications and on their configuration management rollouts, they quickly realized they'd need to develop these capabilities internally to really be successful and further leverage that success: "Over time, we said we need to build this in house, or we won't be able to move as fast as we need to," Kissler said. This is not to say that Nordstrom didn't use consultants where it made sense, but they must be employed judiciously: as expert advisors providing guidance, not staff augmentation.

Nordstrom has accomplished this by adding talent, but also a focused investment in cultivating skills for its current employees. It is notable that the Nordstrom employees interviewed for this case study each had their own personal story through various roles and responsibilities at the company, often entailing entirely new skill sets: Cummings started as an operations engineer and moved into program management for infrastructure engineering; Ireton was originally hired for a very specific Windows skill set, but is now developing configuration management infrastructure after having stints on one of the build engineering teams; and Kissler has worked on both sides of the development and operations organization and at various points in time, has owned numerous parts of Nordstrom's in-store and customer mobile strategy.

A “Have-Coffee” Culture

Any discussion surrounding DevOps and its methodologies quickly comes to the often delicate issue of organizational dynamics and culture, at least if it’s an accurate treatment of the topic. There is often a tendency to downplay or gloss over these issues precisely because culture is thought of as a “squishy” thing, difficult to shape and change, and in some cases, to even address directly. But it doesn’t need to be this way.

Sam Hogenson, Vice President of Technology at Nordstrom, works hard to make sure it’s exactly the opposite: “At Nordstrom, we value these different experiences and we value the core of how you work, how you build relationships much more than whether or not you have subject matter expertise. It’s a successful formula.” Another part of that formula, Hogenson notes, is the ethos of the organization: “It’s a very empowered workforce, a very decentralized organization; I always remember the Nordstroms telling us ‘Treat this as if it were your name over the door: how would you run your business and take care of your customers?’” Ireton described it as a “have-coffee culture: if you need to talk to someone, you go have coffee with them.”

Planting the Seeds

This mindset has interesting implications when observed in a technology department in the throes of its own transformation. Hogenson describes the complexities of fostering cultural change in a system with a large technological component using the metaphor of a garden: “The biggest job is getting the seeds planted, and the seeds for continuous delivery are planted at Nordstrom; it’s getting those seeds from people like Rob Cummings, and then it’s a small bit of top-down leadership and committed investment for the garden; then you put down some anti-weed spray to make sure there’s space for those seeds to grow, and then you just need to pay very close attention to that part of the garden for awhile, because if you forget to water it or don’t tend to the weeds, it will die very quickly.”

Hogenson also takes care to make a distinction between a “push model” and a “pull model.” Perhaps surprisingly, for all of the development work completed on Nordstrom’s continuous delivery pipeline, it’s not required that application teams use it. Hogenson notes that the investment in the pipeline is critically important to the company's success, but also knows that some teams may have a "burning platform" that are a higher priority, to both themselves and the business, to get addressed first. "The things that die are the things you try to shove down people's throats," Hogensen said.

"If there's a place that doesn't want to use it right now, that's fine; there's others that will and they'll demonstrate the value. And soon enough, it'll be organic and spread; in our culture, I can't go 'Well, this is the right idea, so you know all that stuff I tell you about ownership and empowerment, well, that doesn't apply to you because I don't agree with you.'" But, creating a "safe space" to cultivate these new ideas and give them some time to become fully formed—this garden, as Hogenson calls it—is critically important to moving the organization forward, and doing so in a way that meshes with the culture the organization professes to believe in.

Speaking the Business' Language

An issue that many organizations struggle with is how to sell it to the business, and Hogenson notes that Nordstrom isn't any different. But it's a problem that he's keenly aware of, and his solution has been to help cultivate another skill in his staff: speaking adeptly to "the business": "What I try to do is really listen and then coach my staff on how to speak to the topic so our business can understand. We sell shoes, and so Rob's gotta articulate in a way that's going to connect those dots, from continuous delivery to shoes, for us." Hogenson notes that both Cummings and Kissler's teams have succeeded at this task: "Our CFO continues to pour money into our technology investments, because our teams have shown they have the credibility to deliver, and because the return on investment is great," Hogenson said. "That has made future conversations a lot easier, too; when done correctly with the right culture, it's a virtuous cycle. Tending that garden early on is paying off."

Nordstrom's deliberate treatment of its corporate culture and self-awareness around how it permeates the decisions it makes and how it affects its various different teams is a component of their success in delivering the right technology to serve its customers. "Continuous Delivery and DevOps, as 'movements', will come and go," Hogenson explains. "What remains? Culture. That's the thing that enables you to realize when it's time to adopt something new *and* and when it's time to move on." Hogensen is surprisingly frank about Nordstrom's odometer reading on the journey: just a few years into a conscious reforming of how its technology teams mesh with themselves and the larger business, he estimates that Nordstrom is only halfway through working to address the discoveries they're unearthing while tilling that garden.

But the process, even though it dirties your hands, Hogenson says, is what makes the other stuff possible: "If you don't pay attention to culture, everything is really hard to do. But if you do, everything else works."

Flipping On the "DevOps Bit"

It would be inaccurate to present a picture that implies Nordstrom's journey is complete, with all of their applications—in-store, on the website, and on mobile—deployed continuously, with developers and operations living together in constant harmony and using a flawless, infrastructure-as-code-backed pipeline. Every single person interviewed for this case study tempered the stories about the progress that they've made with caveats that there's still more work to be done, more teams to bring aboard. But the successes are undeniable and present themselves in ways both large and small: Cummings recounts a recent "emergency change": "It was a total non-event, because we had infrastructure as code in place." He didn't even have to drive to the colocation facility.

Nordstrom's infrastructure team is currently investing a lot in developer-focused APIs that wrap their core services, like DNS and VM management; they're also working on providing APIs that can unlock the stores of data surrounding their infrastructure, so teams can not only get insight into the running system, but make good decisions for their own applications. There's a nascent public cloud initiative, which seeks to back these APIs with the capability of public clouds—Nordstrom is currently looking at two such providers—in addition to their own internal infrastructure. Application teams will be able to use that API to manipulate and get data from both environments, making the transition easier. Of course, Nordstrom's internal and customer-facing applications continue to be redesigned as teams have bandwidth to pay down technical debt and migrate their builds to a process that fits into the continuous delivery pipeline.

Kissler's goal is to bring the agility of the company's mobile applications to the store application; she knows they may not want to deploy as quickly as the mobile team, but Kissler wants to offer her VP the ability to ship whenever she wants, so it becomes a business decision, unfettered by technology constraints. “They like that story, but they have a hard time believing we can pull it off,” so Kissler and her team are walking them through the value stream process that helped their mobile team continuously deliver. Ireton echoes the sentiment with his recent experiences working with various application teams: “I think we’re mostly over the hump, at least in how people think and feel about the problem. Not all teams are doing continuous delivery or using our pipeline, but more and more teams want to be.”

When asking Nordstrom's team for any advice they'd have given themselves at an earlier pit-stop on their journey if they could, a lot of it surrounds how they'd communicate differently with consumers (i.e., application teams) and bring them into the fold earlier in the process; retrospect also offers pointers on initial projects they may have scoped or even approached differently.

But it was Hogenson who replied quickly and decisively, with the simplest advice: "Keep going."

Texas.gov

Security is on the mind of every management team. With murmurs of corporate information breaches and a steady stream of stories in the press of (sometimes massive) customer data breaches, it's an issue that affects every software development endeavor. No one is more aware of that than Tim Virtue, the Chief Information Security Officer of *Texas.gov*.

Virtue came to computer security through an interesting path: after earning his bachelors of science degree in criminal justice, he started in physical security and investigating fraud and other white-collar crime. At the change of the millennium, Virtue's boss said "It looks like this Internet-thing is going to take off," so from there, Virtue focused on threats from the Internet and went on to work in the finance industry in Washington, DC. Last year, he joined *Texas.gov* as its CISO.

A Public/Private Partnership

In terms of government programs, *Texas.gov* proves to be unique.

The software development and operations teams that work on the massive website serving over 26 million citizens of the state of Texas is run by Texas NICUSA, LLC, a division of NIC, Inc. NIC runs similar companies providing "e-government" services to 29 other states. Virtue's company is part of the public/private partnership managed by the state's Department of Information Resources (DIR) to provide the framework for state and local agencies to web-enable their services. Texas NCIUSA then develops and assists those agencies in operating and supporting services, under the direction of the DIR.

The work is all managed under a master agreement with the state and, perhaps most interestingly of all, the funding model is such that government agencies don't pay up front to get the services developed: rather, Texas NICUSA self-funds those costs and recoups its expenses from end users who pay a nominal transaction fee for the online services.

Services the Department of Information Resources leverages the *Texas.gov* program to provide include everything from driver's license renewals to paying state taxes to obtaining government records—birth certificates and the like—to reserving campsites in state parks to applying for concealed-carry permits, over 1,000 online services in all. "The public is starting to expect the same web-enabled experience when interacting with their government as they do in other parts of their daily life," Virtue said. But it's a delicate balance: constituents demand their government do more with less, as in many other industries.

One constraint that is uniquely interesting about this structure is the specific requirements Texas NICUSA has to incorporate into the way they develop and operate software. Unlike most shops, their software requirements can (and often are) driven by changes in the law. Pete Eichorn, Texas NICUSA's Director of Technology, is responsible for ensuring his teams deliver on these requirements: "When something comes to us that's legislatively mandated, these are not just suggestions. Those are due dates."

The partnership also requires that certain services be operated within the state, so as to keep the data within the state's boundaries. This means certain shared services run by Texas NICUSA's parent company can be leveraged to serve *Texas.gov*'s needs, but others must be run locally. All are governed by IT security requirements that, unlike most businesses, are actually enshrined in law.[1] Many of the documents that define portions of *Texas.gov*'s operational requirement are, in fact, entirely open and available to the public: "It's all transparent, and by law, it has to be, so that's very different than a traditional private sector business," Eichorn said.

---

[1] Chapter 202 of the Texas Administrative Code covers "Information Security Standards."

“The Only Way to Do It”

As we saw with the previous case study, organizations’ journies toward DevOps methodologies can often be traced back to a particular event. Eichorn says that for them, it was a project that came up two years ago that the team knew they wouldn’t be able deliver any other way. “We actually had deadlines and scope on that project that nudged us toward really getting serious about agile,” Eichorn said. “We just didn’t think we’d make it; we had our toe in the water, but this project was really the tipping point.”

Eichorn says it wasn’t perfect the first time around, but they were able to get the project completed about six months ahead of its original schedule, by developing it using more agile methodologies. As the organization brought more teams into the agile fray, the beginnings of some nascent DevOps methodologies started to rear their heads during stand-ups, such as creating cross-pollinated teams and accounting for operational work the same way feature work is accounted for. “We loosely coupled our burgeoning DevOps initiatives and the Agile work our teams had been doing at first; I didn’t want to say ‘Thou shalt DevOps now,’” Eichorn said, “because we wanted to give the teams the leeway to experiment and find out what DevOps meant for us. But as you can imagine, it has to come together at some point.”

Like Nordstrom, Texas NICUSA’s “DevOps ethos” is embodied in a team it calls the “Continual Service Improvement” team, a name which may ring a bell with those familiar with ITIL.[2] The team currently supports the organization’s largest services and takes a shared approach with the service developers and the teams operating the services to fulfill their mission. Eichorn echoed a familiar problem with the traditional hands-off approach between development and operations teams often creating “loss of insight and understanding for both teams.” The CSI team helps to address this by being comprised of cross-functional members from development and operations, but is tasked with improving specific aspects of the interfaces of those applications as well as their delivery and operation in production.

---

[2] The Information Technology Infrastructure Library, a set of practices for implementing IT service and application lifecycle management.

“The Only Way to Do It” | 19

"They're some of the most 'pure', if you will, DevOps stuff we're doing," Eichorn said, but he wasn't sure what the team might look like in a year. The team may be redistributed back out into the various teams to help further seed DevOps capability and culture among *Texas.gov*'s wider team, "if that seems like it would make the most sense," of course.

Continuous…Security?

Like other large-scale websites that perform payment processing and handle vast amounts of customer data, *Texas.gov* must take special care when it comes to security. Virtue is responsible for ensuring *Texas.gov* meets the alphabet soup of security requirements, including PCI-DSS, SSAE 16, HIPPA, and various ISO standards; he's also responsible for complying with all of the State of Texas' laws about information security, parts of which are enshrined in statute.

As Virtue looked at how to integrate this work with the team, the organization's Agile transformation led him down a path that is improving the implementation of security for *Texas.gov*: "We embed a security team member on every Agile team within the company." Virtue is quick to clarify that this means they're not just overseeing the team or signing off on their work, but are part of the daily standups, the retrospectives, and track their work within the larger context of development work. Gone is the model where Security would do its work during the final QA cycle and gates and signoffs were the only feedback mechanism about the larger state of security within the organization's applications.

The Other SaaS: Security as a Service

Virtue has even turned the concept of security itself into a "product" that, as CISO, he is the product manager for. Everything from vulnerability scanning, penetration testing, and audit compliance is now tracked via the Agile sprint process. Signoffs, previously destined for the end of the release, have now been incorporated into the organization's Agile definition of "done," and so must be obtained to clear the sprint. "I'm delivering secure applications and I'm delivering a successful audit; so it's easier to interact with the other teams if we model it in the way they're already working," Virtue said.

Virtue said the shift in how *Texas.gov* addresses its security needs has resulted in faster response times for addressing security issues:

"In many cases, we're catching issues before they get released, because we're embedded in the development process," Virtue said. As an example: developers will discuss new technologies they want to use, like NoSQL for instance. Security team members then add a story to the sprint to research the current state-of-the-art security practices on those technologies and provide strategies to protect *Texas.gov*'s systems and citizens' assets.

Lessons from the Lone Star State

Because Texas NICUSA is in such a unique "industry," and because they started their DevOps journey with a firm grounding in Agile—so much so that it informed how they implement and experience "DevOps"—the takeaways from their experiences provide a unique insight into implementing DevOps in larger organizations.

Focus on Feedback Loops

Keeping feedback loops very tight is a well-known DevOps (and Lean) methodology for improving processes and outcomes. But using it to integrate an often-assumed facet of a product, namely security, is an interesting way to ensure it's kept in the development and operations discussions.

Materially, this plays out in every sprint, as security becomes (quite literally) part of the story, and the prioritization conversations are held constantly. "Doing it that way allows us to ensure we address security issues in a strategic manner, instead of just trying to 'slip' security into the product," Virtue said.

Virtue's team has not only embraced Agile by making themselves available to and participating in the work of development and QA teams, but they also hold their own daily standup meetings. This helps them crosscoordinate larger security efforts, such as audits, and exchange the details they learn from the team standups, helping them to identify work that could induce emergent behavior in the final software system, which the security team would need to address.

Reframe Risk

Most organizations are accountable for the results of what they produce, but at the end of the day, *Texas.gov* is also accountable to the public, a sentiment both Eichorn and Virtue mentioned repeatedly.

One of the big wins Eichorn worked on initially (and Virtue was then later able to leverage in his security work) was around how the organization assesses, manages, and communicates the concept of risk to stakeholders. "Oftentimes, there's a want to study 100% of the risks, mitigations, and liklihoods," Eichorn said. "That's not to say we don't still do that, but we've also increased our ability to be able to move quickly to resolve risks that weren't even in the original model."

This has moved the focus from complete and total understanding of all risks in a project toward the organization's ability to react both to "known unknowns," those situations that it knows it needs to find an answer to, but more importantly, to "unknown unknowns." "It's taken some time to get there, but it's been a very interesting journey, and it's made our state partners more successful," Eichorn said.

Continuously Innovate

Both Eichorn and Virtue spoke repeatedly of Texas NICUSA's "innovative environment." The funding model for getting state projects developed is certainly innovative, but Eichorn describes the further commitment to innovation within the company, all the way up to the executive leadership. This includes the familiar list of items like hackathons and "innovation lunches." But it's how *Texas.gov* runs those events and what it does with the output that is interesting.

In the case of the innovation lunches, Eichorn tends to kick them off with some unique problem statement, sometimes with a solution, sometimes not. A recent lunch started with discussion of the Sikorsky Prize, the helicopter manufacturer's longstanding challenge to build a human-powered helicopter. He showed a video in which it was recently achieved. He uses that to kick off converastions among wide-ranging roles: "We get people from our service desk, finance, security, and even a few developers from time to time," Eichorn said with a chuckle. The conversations often start with "Hey, did developers know that we get all these questions about feature X?" The result of the conversation is a funnel of ideas that get turned into three to five imminently actionable tasks for monthly improvements to *Texas.gov* services.

If this process sounds familiar, it should: it's the Toyota kata, a core-component of the success of the Toyota Production System, itself the grandfather of so many DevOps concepts.

De-Fang the Dogma

As we have seen before, adoption of Agile and DevOps methodologies is a journey, and every person and organization's journey is going to be different. Eichorn had an interesting note of advice when working to adopt new techniques, especially around team interactions: "Patience while you're adopting things is super-important, as is not necessarily believing it all at the beginning; the biggest risk over time is you start believe all the new stuff as rigidly as you believe the old stuff."

It's a cautionary tale of getting caught up in the dogma of a new set of suggestions, a new worldview, a new toolchain, a new conference, a new community. Eichorn warned that it's important to have the commitment to the cultural and technological changes, but you don't want to be so attached to it that you can't adapt it to your own organization's products, market, culture, or needs.

A Unicorn with a Cowboy Hat

Discussions of organizations working through what DevOps means for them don't usually get very far without mentioning "the unicorns." Whether it's streaming movies over the Internet or rounding out your collection of hand-knitted tea cozies at the push of a button, there is a go-to list of companies that seem to many to live in a magical stable somewhere, full of rainbows and gumdrop grain.

But Texas NICUSA's own journey with combining Agile and DevOps shows that any organization can be successful at it and can be successful in helping partners improve at it, even if the entire journey isn't understood when they begin. "We don't have the full map, but we have the intent and direction," Eichorn said.

"We just have to remain realistic, to be sure we're not blinded by our enthusiasm about these new methodologies and tools, so we can continue to execute on the Lone Star State's needs."

Acknowledgments

These case studies would not exist had Rob Cummings, Sam Hogenson, Doug Ireton, Courtney Kissler, and Jeff Raffo from Nordstrom and Pete Eichhorn and Tim Virtue from Texas NICUSA not set aside time in their busy schedules to discuss their experiences. A special shout-out to Jeff Raffo, whose words do not appear in the case study but whose insight and context-setting were invaluable to the piece.

Special thanks to O'Reilly editor Courtney Nash, who noticed that I have a tendency to say things from time to time, trusted me that first time to blog on *oreilly.com*, and started me on this journey; and to editor Brian Anderson who adeptly dealt with the eventual result.

Thanks to Sascha Bates, Kevin Behr, Pete Cheslock, EJ Ciramella, Jennifer Davis, Youssuf El-Kalay, Nathen Harvey, Jez Humble, and Seth Thomas who continue to provide encouragement (and smackdowns as appropriate). Thank you to Mary Thengvall, who has on numerous occasions dragged me to the coffee shop when I needed to write to meet a deadline and to the bar when I needed to talk.

Oh, and thanks to my roommate James Castañeda's basset hound, Clipper, who on many a morning pushed her way onto the bed for a snuggle, but who was also astute enough to push me out of the bed when it was time to get the day started.

About the Author

J. Paul Reed is a Principal Consultant at Release Engineering Approaches and Visiting Scientist at Praxis Flow. He has over fifteen years' experience as a build/release engineer and has worked with companies such as VMware, Mozilla Corporation, Symantec, and various start-ups and clients across numerous sectors. Paul is the author of several whitepapers and articles on DevOps and continuous delivery and is a founding host of The Ship Show, a podcast covering "Build engineering, DevOps, release management, and everything in between." He lives in San Francisco, tweets on *@SoberBuildEng*, and blogs at http://soberbuildengineer.com/.