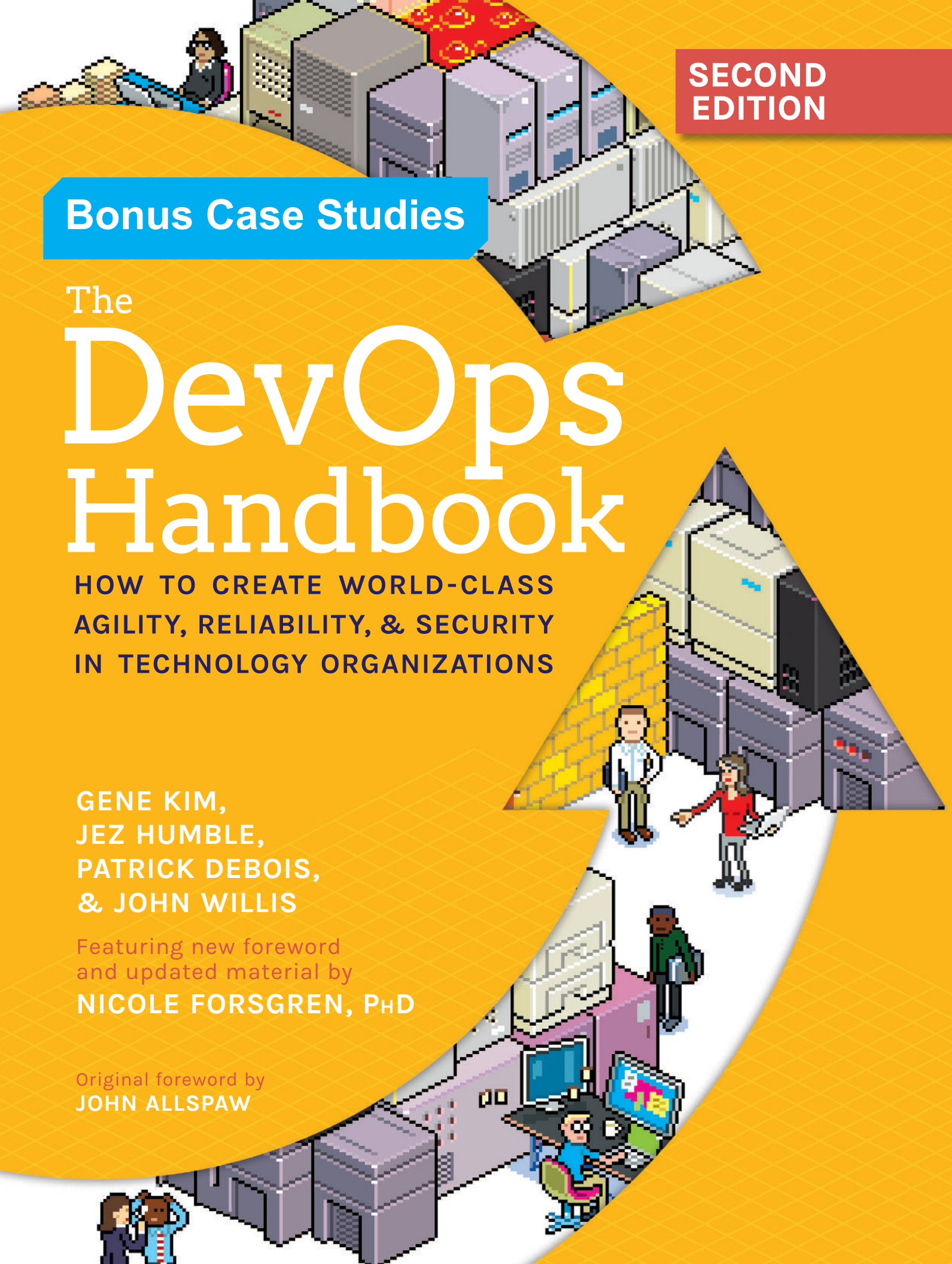
SECOND EDITION

Bonus Case Studies

The DevOps Handbook

HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS



GENE KIM,
JEZ HUMBLE,
PATRICK DEBOIS,
& JOHN WILLIS

Featuring new foreword
and updated material by
NICOLE FORSGREN, PhD

Original foreword by
JOHN ALLSPAW

25 NW 23rd Pl
Suite 6314
Portland, OR 97210

Cover and design by D. Smith Creative

DevSecOps Implementation: Not for the Weak of Heart (US Navy) (2021)

The US Navy is in the middle of a massive information technology modernization, to include new hardware suites, software architectures, and continuous integration/continuous delivery software factories. To realize the full potential of the IT supporting Sailors and Marines worldwide, the Navy's leadership realized it also had to make significant changes in how its software product teams were organized and led.

Our team participated in a Navy DevSecOps pilot program to

- learn how to build a generative culture within our product team,
- address an organizational need using modern software practices and tools,
- and involve fleet end users to understand and address their software pain points.

Our team found that while each DevOps element was simple to understand, the emotional and cognitive work involved in meshing those practices into a high-performing methodology was non-trivial.

In collaboration with an industry partner, we began our DevSecOps pilot by selecting product team leaders: a product owner, an engineering lead, and a user experience lead. These three leads represented the needs of the organization, the needs of the system, and the needs of the end users. Using these three viewpoints on a daily basis allowed us to balance operational capability, system reliability and security, and user adoption for our product.

With our leaders in place, we collaborated with the staff of Naval Air Forces to understand how we might use software to optimize their aircraft maintenance schedules. With an understanding of the scope of effort and a high-level charter, the product team leaders recruited software developers and a cybersecurity expert to round out the team. The entire product team set up an off-site office space, where we hoped that the team could reinforce new habits without distraction.

Once there, we started deliberate team-building exercises. We created a social contract, where all team members provided input on how the team wished to operate and how all team members wished to be treated. We also conducted exercises designed to get to know each other, such as identifying similar hobbies and interests. We embraced a problem-solving mindset to encourage collaboration among team members by using phrases such as "How might we . . . ?" and "Yes, and . . .".

Creating a generative environment was quite challenging for some of the team's senior members, as they had come from bureaucratic organizations with a clear leadership hierarchy. To participate as equals in a collaborative team environment took practice and patience. This was evident on multiple occasions: while debating the value of a fea-

ture, during a candid team retrospective, or when discussing the merits of a particular technical approach. Our team members learned to check their egos at the door, acknowledge that the team knew more than any individual did, and assume positive intent from every engagement.

The next phase of our team's development, using modern software practices and tools, was also taxing for some team members. Depending on the tasking of the day, we would work as an entire team, as breakout teams, or as pairs. Working in small teams encouraged focus on the task, resulting in accelerated learning and increased productivity. It was also cognitively challenging to focus for long periods while being accountable to your teammates. The first few weeks performing at this level were both exhilarating and exhausting.

With our team organized and ready for action, we brought in subject matter experts in aircraft maintenance to better understand how maintenance is scheduled and performed, to include policies, procedures, and information flow. We also interviewed end users to understand their pain points. The synthesis of this information allowed our team to create its roadmap of epic features, along with a priority to address the most prevalent pain points first.

Having the product team self-organize its efforts based on end user feedback was a refreshing change from deciphering software performance specifications. It also placed full product ownership on the team, which highlighted the importance of rigorous user experience methods, defendable research results, and actionable user stories.

In addition to frequent demonstrations to the users, we made cybersecurity best practices transparent across all members and stakeholders. During our sprint demonstrations with users and stakeholders, we would review how the software quality and security changed between demonstrations. Communication of security priorities allowed us to increase understanding of why we would decide to delay a new feature in order to fix a critical security flaw or improve code quality.

By creating a generative environment, using modern software practices, and focusing on the end user, our team was able to create a workable maintenance scheduling prototype within eight weeks, and then delivered a minimal viable product with all required security certifications to Naval Air Forces in six months. Our team members now share their lessons with other Navy product teams with the hope that they find these DevSecOps practices as useful as we did, while managing the expectations of the serious commitment it takes to perform.

Building a DevOps Mindset across the Business and Technology at Target (2015–2021)

Circa 2015, Target, one of the largest retailers in the US, was struggling with multiple priorities on which they were making very little progress. Their delivery model was based on third parties, so they retained very little IP, which presented morale and stability challenges, especially since the organization was rooted in legacy ways of working.

Target needed to address issues of what the company was prioritizing, update the skill-set and makeup of the workforce, anchor more to architecture, and move from project to product using Agile principles.

At the 2021 DevOps Enterprise Summit, Brett Craig, Senior Vice President of Digital, and Luke Rettig, Senior Director of Product Management, Merchandising Capabilities, described how Target used a DevOps mindset to influence the organization's culture, highlighting two key examples: the revamping of the Fresh Food Merchandising division and the company's response to the COVID-19 pandemic.

In late 2018, the Fresh Food division was underperforming based on all of Target's core metrics—they were unable to keep high-quality perishables in stock, which was calling into question whether or not they would still be in the business of fresh food at all in five years' time.

Their first instinct was to add more management reporting, meaning Target would bring in a large group from a consulting organization to fix the issues they were experiencing. However, Brett Craig, who had recently moved into the merchandising space from technology, deduced through conversations with coworkers that fewer people were needed to fix this problem. Instead, he suggested standing up a small group of functional experts composed of business, technology, operations, and product people—designed to build sustainable, repeatable capabilities—led by Luke Rettig.

This small group was given the guardrail of driving profits without sacrificing sales in Fresh Food businesses and worked in a shared common space to empower them to communicate in person to figure out what handful of measures actually mattered. This enabled the team to leverage short sprints, kanban boards, and fast feedback loops.

This group determined how to segment out approximately two thousand stores and utilized analytic capabilities that allowed them to go into those segments and decide how much space was needed to maintain sales. Space is the most constrained asset in brick-and-mortar stores, and using data and intelligence to optimize how much space each category was allotted before they saw diminishing returns on profit represented a significant shift in thinking.

This small team built a unique assortment of space-informed segments that each had localized decision-making, introducing base metrics such as mechanical shrink and putting data and sustainable tools in the hands of the people making decisions.

As a result, Target saw an increased gross margin of 32% in fresh-food categories while only sacrificing +/- 2% in sales, decreased square footage allocated to fresh-food categories by 15%, and built Version 1 of the Assortment Planning Product toolkit (a toolkit the rest of Target's businesses are able to iterate off of).

Culturally, it was liberating to not automatically default to the standard way of doing things, and it built trust between team members to see positive results from adopting a more communication-oriented method for addressing the issues in Fresh Food. This trust in new methods also helped team members in the organization stay healthy, safe, and thriving in the beginning of the COVID-19 pandemic.

Before March of 2020, the decision-making and operations of Target's merchandising structure were siloed. Target was merchant-driven, less thoughtful of what was best for end-to-end operations or guests, and rarely considered how to come together as a team of teams. There were weekly, merchandising-focused action meetings that were separate from weekly operations meetings. These routines were neither enterprise thinking nor all-inclusive, and although product and technology teams were operating more in the spirit of DevOps, they were viewed as a separate tech arm.

When the pandemic began, Target was in crisis, starting with the essentials and food and beverages businesses. Shelves in these categories were empty, while in the meantime, there were overages in apparel and accessories, with stockpiles of inventory in stores and distribution centers. As the pandemic progressed, Target saw a shift to increased sales in electronics (to support home offices) and leisure activities (such as board games and other things families could enjoy in isolation). Additionally, Target had to deal with the chaos introduced by guests needing a clean and safe place to show and field team members who had their own safety and processes to maintain.

The Merchandising Capabilities team was uniquely positioned to see the total value chain and orient to a critical path by setting strategy and allocating space. They did so by pivoting to daily action meetings in which they reviewed progress and identified roadblocks and the correct people to make cross-enterprise decisions. Luke Rettig refers to this as "standup at a scale," in which business owners from across the value chain come together in decision-making.

During these daily action meetings, the team considered how to protect field team members, shifted decision-making into what was best for guests and those team members, and started looking at metrics that weren't quarterly, such as longer term win-wins with vendor partnerships and looking ahead to two-year results.

This allowed them to "get rid of the noise" and work collaboratively with trust in leadership and each other while also breaking down silos. They were able to save significantly on team member labor in the early days of the pandemic by pausing a large number of clearance programs and pausing and reconfiguring the introduction of new-

ness into the stores that ordinarily occurred with every seasonal change—decisions that were made in these daily action meetings and fueled a broader enterprise mindset.

More recently, Target has rolled out a refreshed culture framework, which reminds the entire organization of what it takes to reinvest in the business and each other as they "care, grow, and win together." This framework promotes connection, inclusivity, and drive, and encourages team members to "choose progress over perfection."

Moving forward, Target sees room to grow in maturing product thinking, investing in technology talent to help drive development, and continuing to emphasize the principles of DevOps in the organization.

Iterative Enterprise SRE Transformation at Vanguard (2016–2021)

Vanguard, a global asset management company, began to adopt SRE best practices to make their DevOps teams more effective around 2016, and shared their experiences at the 2021 DevOps Enterprise Summits.

At the time Vanguard began its transformation, they had not yet begun public cloud migration. All monolithic applications were hosted in a privately owned data center. All deployments were released on a quarterly schedule by deployment and operations teams—not development teams. These deployments were monitored via an "alert-only visibility" policy, under which it was assumed that if there were no alerts, the application was up and running. Ownership of the alerts was centralized. In order to get an alert configured, the application team had to submit a request to a central team and wait for them to have a spare cycle to set up an alert.

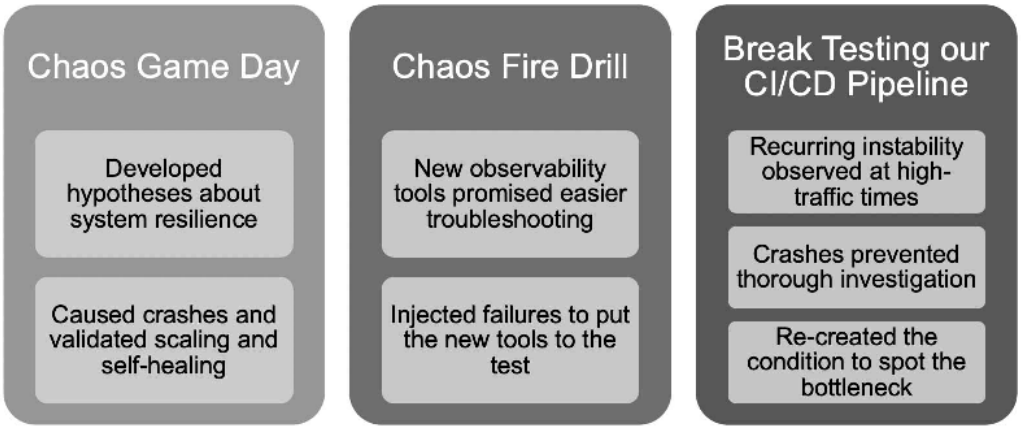
In order to migrate from a data center to a public cloud, Vanguard had to "break down the monolith" by slowly carving out microservices, which reduced the duration of the regression cycle. They began running microservices on a platform-as-a-service (PaaS) private cloud that was run from their data center. They also introduced a test-automation engineer role to create tests for smaller slices of functionality covered by the microservices. This allowed Vanguard to increase deployment frequency, to automatically generate change records, and to attach automated test evidence to increase the velocity of the change-management process.

These changes helped Vanguard focus on lifting and shifting PaaS into the public cloud. This meant that all microservices that had been running PaaS in the on-premise data center had to make few changes to complete migration to the public cloud because the platform was the same; however, it was difficult for infrastructure teams doing that work—the shift expedited the cloud migration for microservices, but left Vanguard with an unnecessary abstraction layer.

Next, Vanguard needed to take the microservices that had been running in the public cloud on the PaaS and get them into a more cloud-native solution. This would reduce the operational complexity for the infrastructure teams that put the PaaS in the cloud initially and allow them to leverage out-of-the-box resources provided by public cloud providers. Because the carved-out microservices worked a little differently, it made sense to put some of them into AWS Lamda (for a serverless compute option), and Amazon EKS was a good option for other services that could benefit from the control plane of Kubernetes.

Now that Vanguard was exploring cloud-native solutions outside of the PaaS, much of the infrastructure responsibility rested on the microservice application teams. These teams were able to leverage new features (autoscaling, automated task replacement), but they needed to test new configurations, which were no longer centralized by on-premise configurations or PaaS configurations. They needed to adopt new day-to-day processes, such as using chaos engineering to validate the teams' hypotheses about how their systems behave in times of stress. Performance testing was another way to test systems, but as the deployment frequency increased, the teams also needed to increase the frequency and flexibility of performance testing. To address this, they built a performance-testing-as-a-service application so that all individual product teams had access to the hosted load generators they needed to test their own applications.

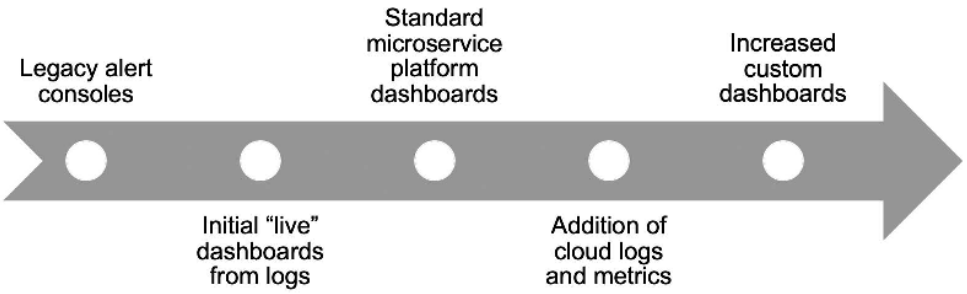
What seemed like significant additional cognitive load on the application teams actually yielded several successes. Teams used chaos game days (in which they developed a hypothesis about system resilience and then tested that hypothesis by purposefully causing crashes to different components of the application and validating scaling behaviors and self-healing) and chaos fire drills (in which they intentionally injected faults they knew would raise system alarms in order to test new observability tools before they went live). In addition to testing systems and tools, these processes were also helpful for onboarding and training.



In a unique process that blurred the line between chaos experimentation and performance testing, Vanguard also performed brake testing on the CI/CD pipeline. As a large IT organization, they ran into growing pains as they were onboarding microservices into the CI/CD pipeline, observing recurring instability during high-traffic times, in which crashes were preventing thorough investigations because they were wiping out the critical logs before they could be offloaded to log-aggregation tools.

To troubleshoot this issue, they performance-tested the pipeline over a weekend by creating builds and deployments that recreated specific resource-intensive conditions, recreating crashes while someone watched the log file be created and then captured the relevant log files and thread dumps, allowing them to identify and address the bottleneck. By the following Monday, they saw immediate improvements to the pipeline.

Vanguard also had to find the right tools for their observability journey. "Alert only" visibility led to legacy alert consoles that were still almost exclusively used by operations teams. As microservices were carved out, the key benefit of the PaaS was that all the applications were operating in the same containerized environment. Their logs were filtering in the same ways and into the same places, so they were able to create standard microservice platform dashboards, originally intended for use by the platform owners.



They soon realized how beneficial these dashboards were for application teams, who began working with infrastructure teams to make them more customizable by cloning the dashboards and adding their own panels that were specific to their use cases. Application teams began adding alerts, which helped them to move quickly now that they were no longer submitting requests to central teams. This allowed them to make data-driven decisions as a unit; however, some of the consequences of these customizable dashboards included dashboard clutter, ignored alerts, and alert fatigue, because teams had access to alert customization but not all of the necessary information about best practices for alerting and dashboarding.

Because there were so many dashboard and alert queries running, "everything was logs." Everything was in one tool and everyone used one querying language, but the scope and utilization were increasing. Costs were also rapidly increasing, and the performance of the tool was degrading, meaning that troubleshooting was held up by dashboard performance.

To solve this problem, Vanguard pulled metrics and traces out of the central tool and utilized tools like Amazon CloudWatch for metrics and Honeycomb for traces. The distributed tracing functionality offered by Honeycomb was used in chaos fire drills to see how much easier it would be to identify sources of latency within a complex web of microservices. To get from the user interface all the way to the datastore and back, they were encountering many different microservices depending on each investor's account structure.

As part of the move to distributed tracing and the adoption of Honeycomb, Vanguard also standardized around OpenTelemetry—a move they saw as investing in the future by learning from the mistakes made in the past with other tools. Because standardizing around the OpenTelemetry framework for sending telemetry data to backend collectors is now common practice in the industry, Vanguard believed that the same practice would allow them to more effectively avoid vendor lock-in. Many observability tools offer integration with OpenTelemetry collectors out of the box, which makes it simple to swap out backends in the future for logs, metrics, or traces if it becomes necessary.

As part of this investment, central teams at Vanguard began creating shared libraries for the benefit of application teams, e.g., to extract common fields that they may want to add to their trace context regularly.

More recently, Vanguard has made changes to SRE best practices by changing the way they measure availability by injecting nuance in the previously binary way availability is discussed. They have been rolling out the SRE practice of using SLIs and SLOs to talk about availability and latency in terms of reasonable thresholds based on the expectations of clients, as opposed to saying that an application must be as fast as possible and always available, a mindset that leads to burnout since it is impossible to achieve 100% uptime.

Moving forward, Vanguard is aiming to strike the right balance between efficiency and flexibility, and between time spent training and upskilling and time spent delivering. They are also seeking to decrease on-premise workloads by moving more applications to the public cloud and to become fully observable by developing a single visualization tool to aggregate the telemetry data. Finally, they are striving to grow in blameless post-incident reviews by sharing knowledge throughout the entire IT organization, which, when done correctly, maximizes learning for the entire organization and improves their ability to operate as DevOps teams effectively.

DevOps and Internal Audit: A Great Partnership at Nationwide Insurance (2021)

Nationwide Insurance offers financial services, property and casualty lines, and pet and travel insurance, among other services. At the 2021 DevOps Enterprise Summit, Ethan Culp (NETC Sr. Associate), Rusty Lewis (IT Audit Specialist), and Clarissa Lucas (IT Audit Director) described how the Nationwide Internal Audit office is putting DevOps theories into practice, how they think about risks and controls, and the influence of automation on risk mitigation.

As the Nationwide team shared, here are three lines of an organization. The first line owns risk and executes controls to manage them (e.g., development teams that write and develop code, and review and approve changes). The second line is responsible for policy creation, defining risk tolerance, and monitoring adherence to policies (e.g., an information risk-management function). The third line is Internal Audit, which provides assurance to the audit committee of the board and senior management through independent assessment of risks and controls.

Internal Audit achieves this by seeking to understand what could prevent the organization from achieving its objectives, taking into account all of the many risks. They evaluate the action management is taking to mitigate those risks within established tolerances and, to tie everything together, they conduct integrated audits to evaluate business versus controls and systems and applications that support business processes. They don't tell management how to manage a particular risk, but they do partner with the first and second lines and external parties to provide assurance of the organization's ability to meet objectives.

The primary risks of change processes are compromised systems, data availability, integrity, and confidentiality. Internal Audit performs audits to evaluate controls and determine whether the items are well managed, using a strong control environment to mitigate these risks. Currently, Internal Audit manages risk through testing that depends on three controls: approvals, segregation of duties, and metrics monitoring.

The objectives of the approvals control are that all application changes should be tested by a person other than the developer before they move into production, a sample of 1–30 changes are used for detailed testing, and documentation shows evidence of approval for sample changes at various stages throughout the change-management process.

Metrics monitoring includes reporting changes to monitor effects on business operations. New tools, analytics, and upskilling are allowing Nationwide's Internal Audit team to test the full population of changes with greater accuracy, which enables continuous auditing in parts of the business and partnership between business units to mitigate risk as part of the development life cycle.

Segregation of duties in a DevOps environment means that duties are separated to ensure that the same individual isn't writing, testing, and promoting their own code into production without independent checkpoints in order to guarantee that code isn't malicious or fraudulent and that it doesn't lead to system disruptions or data issues. To prevent these issues, the person writing the code reviews the code to detect defects early in the process and to maintain consistency in design and implementation, which allows for uniformity of understanding and helps team members be interchangeable. During automated reviews of tests, a script reviews a developer's code and promotes it into production once it's been tested, reviewed, and approved. This allows developers to more efficiently review code on demand while separating the duties between a human developer and digital worker.

In addition to segregation of duties, automated production deployment is also present in a DevOps auditing environment. In this process, the code is automatically moved to production as long as it meets predefined requirements, like a suite of controls, that lessen the risk that the code will not perform correctly or introduce vulnerabilities. Automated deployment control rejects code that doesn't meet the predefined requirements.

Internal auditors test this control by looking at its configurations to determine whether it is designed to mitigate the risk by accepting only code that meets the requirements and rejecting code that does not. They also examine this control through observation and review once the control has already run to make sure code is appropriately accepted and rejected as expected. This is a different process than other audits, wherein auditors review many transactions, like code deployments or changes.

One example of how automated testing has been implemented effectively at Nationwide comes from one of the build teams, Partners and Beyond. This team is responsible for generating insurance quotes for customers and was previously a monolithic

service hosted on premise. They are currently designing and building a new application using cloud-based managed ETL services and streaming data to themselves.

As the team develops a new process and skill-set, it is essential that they have breathing room to manage themselves. Partners and Beyond have developed a manager-approved deployment pipeline that goes through a series of checks and gates that allows them to operate more efficiently. This pipeline has multiple dependencies but can only advance if the automated testing gives the green light and if all events are logged in their tool. Errors trigger automated emails and are displayed in a process tree dashboard.

This automated testing also promotes poly-skilling through code reviews and an emphasis on documentation, which also helps the audits maintain objectivity. Developers are spending less time testing and are able to do more releases in a given time, so velocity goes up as lead time decreases. Faster deployments and more releases keep Nationwide competitive in their marketplace.

The Nationwide Internal Audit office is itself also adopting more Agile practices. In late 2020 and early 2021, the Internal Audit organization tested specific financial reporting controls in partnership with Nationwide's external auditors in a high-profile endeavor with a tight timeline that drew attention from key stakeholders. Due to the tight timeline, the Internal Audit office needed to think about work differently, and because they were performing testing on behalf of another party, they needed to be able to pivot quickly and maintain continuous feedback. To do so, they incorporated Agile practices.

The self-organized auditing team avoided multitasking using dedicated resources, which differs from the usual operating model, where an auditor might have multiple engagements happening at the same time. Objectives were clear, so minimal time was spent negotiating priorities. The project manager determined what testing was to be done and which procedures to perform, while the team was responsible for deciding how best to accomplish that testing. This flexibility of resources let people jump from one task to another as they were finished without having to be formally assigned items by the project manager.

The team incorporated continuous delivery in short sprints by dividing testing into four buckets with a number of controls to be completed in each bucket. Each bucket lasted approximately one month. They worked together frequently with stakeholders throughout the audit, which was accomplished via iterative meetings with internal and external stakeholders, after which the team sent out detailed notes and sought feedback—a step toward adoption of the Agile principle in which business people and developers work together daily throughout the project.

This team also conducted daily standups and reinforced the urgency of the audit, which allowed them to shift resources to items that needed attention, to share knowledge, and to change testing procedures as needed. A blameless retrospective review was held at the end of the third bucket to reflect on opportunities to be more effective during the final bucket, during which this team found a way to automate testing of controls in the R

programming language to eliminate the need to manually filter via Excel, saving fifteen hours per quarter.

In a future auditing pilot, Nationwide will add more DevOps and Agile practices by prioritizing customer needs via aligning Audit with the client's workflow process; increasing focus on continuous delivery through sprints and fostering a collaborative environment; continuing to explore ways to automate tests; and holding a blameless post mortem alongside clients.

Driving a Tech-Led Reimagination through DevOps at eBay (2021)

According to Mark Weinberg and Randy Shoup, eBay is too slow and lags behind industry leaders in engineering velocity. At the 2021 DevOps Enterprise Summit, Mark Weinberg, the VP of Core Product Engineering, and Randy Shoup, the VP of Engineering and Chief Architect, described how they are using DevOps to transform engineering at eBay.

Their mission is to make eBay's software delivery velocity a competitive advantage for the company by addressing the systemic challenges that have accumulated over the years, including monolithic code, missing tools and infrastructures, poor processes, etc. Ultimately, fixing these issues will be a company-wide endeavor and not solely limited to engineering.

To begin this initiative, Weinberg and Shoup assessed the situation at eBay using value stream mapping, which flows: Plan → Develop → Deliver → Iterate. What they found were problems and bottlenecks at every stage.

At the planning stage, they found significant inter-team dependencies and conflicting priorities. Next, Development was facing slow build and test times, too much context switching, and bottlenecks created by wait states. At the delivery stage, there were staging issues, and many teams lacked an effective end-to-end, automated development pipeline. Finally, in the iteration stage, they found no end-to-end monitoring, tracking issues, and dysfunctional experimentation.

With all these challenges, Weinberg and Shoup had to figure out which to address first. They decided to focus on software delivery and development because fixing this issue would help clean up challenges both up and down the value stream.

They also needed to decide how they were going to measure if they were achieving their goals. They decided to go with the DORA metrics popularized in the *State of DevOps Reports* and the book *Accelerate*.

Based on these metrics, eBay was a medium performer overall. However, through this pilot initiative, teams began to move into the high-performing area, achieving triple deployment frequency while reducing lead times by 2.5x, and improving the change failure rate 3x. As of fall 2021, 10% of the active apps at eBay are involved in this pilot, which focuses on both short-term wins and longer capabilities. The goals include driving improvements in developer productivity, software delivery, instrumentation and monitoring, and re-architecting critical areas.

Shoup and Weinberg attribute these successes to focusing efforts on a select number of pilot domains (which cover big areas of the product, like Selling, Search, and Ads), applications within those domains, and platform tracks (which are the tools and infrastructure that support builds, CI, staging environments, and the education and training of engineers).

In the initiative, there is a collaborative effort made across the technology platform teams and the application teams. They work closely, which didn't use to be the case at eBay. Today, senior architects are embedded in teams to advise and occasionally write critical code. There are daily standups between leaders, weekly team-of-teams meetings, weekly deep dives, and monthly operating reviews with the executive team.

As part of the pilot, the DORA four key metrics were added to the development dashboard of every application, at which point Shoup and Weinbeg iterated by adding granular viability to every deployment pipeline in order to help teams debug their processes. The teams work in a tight Plan, Do, Check, Act (PDCA) cycle. Shoup and Weinberg use the collaborative environment to steadily remove impediments by centering the initiative's goals on conversations about what teams need from leadership in order to achieve their aims.

In addition to the previously stated quantifiable successes, the teams involved in the pilot have also reduced build, startup, and PR validation times; invested in their staging environments by using good data and reliable components; introduced automated upgrades, testing, deployment, and increased site speed; streamlined team processes and code reviews; and moved from monthly to weekly mobile releases.

The adoption of more DevOps practices also led to a culture change among teams. They look at metrics more regularly and are inspired to communicate and help each other. Teams in product organization have begun automating their own workflow. There are regular demonstrations of new tools and practices, learnings are shared between different groups, and executive support and engagement is high.

This cross-functional velocity initiative also continues to face challenges, however. Shoup and Weinberg still hope to improve overall eBay outcomes, and at times the initiative feels under-resourced while leadership feels overtaxed. Some team members are too

focused on metrics and less on how those metrics support the pilot's overall goals, and other inhibitions include fear of failure and consequences and lack of confidence in the program's approach.

As they continue to move forward, Weinberg and Shoup hope to employ rolling planning with small, frequent, and cheap experiments. In the development phase, they want to utilize small batch sizes, fast build and testing automation, daily merges and deploys, and decoupled architecture. Success in the delivery space would include a fully automated test and deployment pipeline, one-hour commit to deploy, and iteration in production using feature flags. Lastly, in the iterate stage, there's room for more end-to-end monitoring, tracking everywhere, and more small, cheap experiments with rapid feedback on results. The initiative will soon scale from 10% to 50% of applications.