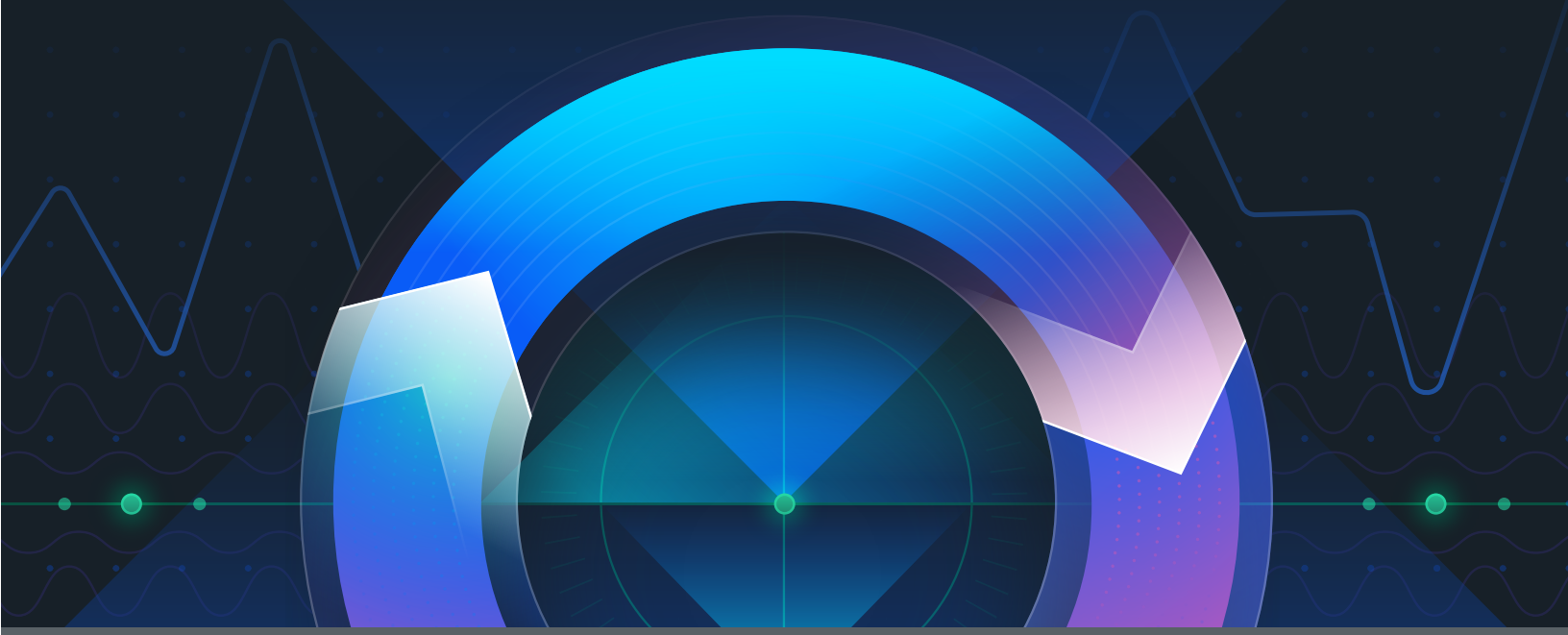
DZone

THE 2018 DZONE GUIDE TO

DevOps

CULTURE AND PROCESS

VOLUME V



BROUGHT TO YOU IN PARTNERSHIP WITH

ca technologies | Automic

CHEF

cloudbees

CLOUD FOUNDRY FOUNDATION

CLOUD NATIVE COMPUTING FOUNDATION

Datical

INSTANA

outsystems

Perfecto

rainforest

redgate

signal fx

XebiaLabs Enterprise DevOps

(x) matters

Dear Reader,

It seems lately that DevOps has become less of a question mark in the minds of developers and organizations are more of a line item, something that needs to happen sooner or later in order to stay competitive in a world where major companies like Amazon and Netflix can deploy changes every few minutes. As DZone has covered the space for over five years, we've seen understanding deepen within the community' major industry players come, go, and shift their efforts; and a recognition of exactly what barriers stand between the concept and practice of Continuous Delivery.

As we approached the focus of the 2018 Volume of *DZone's Guide to DevOps*, we thought of what needed to be covered beyond tooling, code, inputs, and outputs. One of the more difficult barriers to overcome was support from management and fostering a culture of DevOps, including the willingness to fail fast, experiment with new technology, and focus on quality. Top executives that we spoke with have also pointed to company culture as the most important ingredient to implementing Continuous Delivery. So, this year, we decided to focus on *Culture and Process*.

To focus on those points, we're covering the process of choosing what KPIs determine success, avoiding silos in an organization, sprint planning with Continuous Integration and Continuous Delivery, how to take user personas into account when designing your pipeline (particularly non-technical personnel), and how to move beyond those towards a "product-first" culture where the needs of real users can be collected and acted on. We also spend some time re-focusing on basic principles and anti-patterns to acclimate those new to DevOps processes and remind everyone of the fundamentals.

We've found through our research that while cultural change is still difficult to produce, management has already gradually begun to understand why DevOps is a necessary step, and that integrating automated testing solutions in the pipeline has become more difficult — or the issue has become more apparent — in recent months. We've seen microservices adoption grow significantly since last year's DevOps Guide, and a renewed focus on security has started to take hold in the minds of developers after several high-profile attacks with enormous implications.

As we're reminded every year, even though DevOps has become a stable presence in the world of enterprise development, and its importance is well-understood, the barriers, and solutions, to adoption are still changing. We hope this guide provides you with some guidance on how to navigate the "softer" sciences of cultural change and establishing repeatable processes that can help you on your journey. Happy reading!

By Matt Werner

PUBLICATIONS COORDINATOR, DZONE

DZONE IS...

PRODUCTION
CHRIS SMITH, DIR. OF PRODUCTION

ANDRE POWELL, SR. PRODUCTION COORDINATOR

G. RYAN SPAIN, PRODUCTION COORDINATOR

ASHLEY SLATE, DESIGN DIR.

BILLY DAVIS, PRODUCTION ASSISSTANT

MARKETING
KELLET ATKINSON, DIR. OF MARKETING

LAUREN CURATOLA, MARKETING SPECIALIST

KRISTEN PAGÀN, MARKETING SPECIALIST

NATALIE IANNELLO, MARKETING SPECIALIST

JULIAN MORRIS, MARKETING SPECIALIST

BUSINESS
RICK ROSS, CEO

MATT SCHMIDT, PRESIDENT

JESSE DAVIS, EVP

SALES
MATT O'BRIAN, DIR. OF BUSINESS DEV.

ALEX CRAFTS, DIR. OF MAJOR ACCOUNTS

JIM HOWARD, SR ACCOUNT EXECUTIVE

JIM DYER, ACCOUNT EXECUTIVE

ANDREW BARKER, ACCOUNT EXECUTIVE

BRIAN ANDERSON, ACCOUNT EXECUTIVE

CHRIS BRUMFIELD, SALES MANAGER

ANA JONES, ACCOUNT MANAGER

TOM MARTIN, ACCOUNT MANAGER

EDITORIAL
CAITLIN CANDELMO, DIR. OF CONTENT & COMMUNITY

MATT WERNER, PUBLICATIONS COORD.

MICHAEL THARRINGTON, CONTENT & COMMUNITY MANAGER

KARA PHELPS, CONTENT & COMMUNITY MANAGER

MIKE GATES, SR. CONTENT COORD.

SARAH DAVIS, CONTENT COORD.

TOM SMITH, RESEARCH ANALYST

JORDAN BAKER, CONTENT COORD.

ANNE MARIE GLEN, CONTENT COORD.

Executive Summary

BY MATT WERNER
PUBLICATIONS COORDINATOR, DZONE

Continuous Delivery and DevOps are still very popular topics of discussion, particularly on DZone, but 8 years after John Allspaw's talk at Velocity and 7 years after the publication of *Continuous Delivery*, haven't all the issues been discussed and solved? Of course, the answer is "no." In 2018, it seems that DevOps adoption has not seen significant gains in several areas since the previous year. While there's been an insignificant growth in those who claim to have implemented Continuous Delivery across their organization (17% in 2017 vs. 18% in 2018), there was a significant decrease in the number of respondents who had implemented CD for some projects (35% in 2017 vs. 28% in 2018). This is the first year that DZone has not seen an increase in the percentage of respondents adopting CD for some projects. Is this the start of a new trend, or a blip on the radar?

To find out just how DZone's audience was implementing Continuous Delivery, and why adoption might be falling, we surveyed 549 technical professionals to learn about their struggles, successes, and focus areas.

DEVOPS PROCESS ADOPTION SEEMS STAGNANT

DATA

Several statistics between our 2017 and 2018 survey respondents changed insignificantly. For example, there was a 2% drop in operations involvement in designing the CD pipeline (43% vs. 41%), a 2% drop in those who claimed they had implemented CD (20% vs. 18%), a 1% drop in those who claimed they had implemented CI (31% vs. 30%), and no change in those who used push-button or automated deployments via a pipeline (42% in both years). However, a major change was observed in those who had adopted CD for some projects, a decrease of 7% from last year (35% in 2017 vs. 28% in 2018).

IMPLICATIONS

These numbers suggest that several organizations are currently at an impasse in their DevOps adoption efforts. Some of the main causes of the stagnation may be either brand new or just newly realized pain points that are stifling progress for now. The decrease in CD adoption for some projects may be attributed to a different range of respondents for this year's survey compared to the surveys for earlier years, but data from future surveys will help create a clearer picture.

RECOMMENDATIONS

One commonly cited issue with DevOps adoption has been management buy-in. However, since management was only seen as an inhibitor by 17% of respondents in both 2017 and 2018, it's unlikely that management support, or lack thereof, is a significant factor in the above statistics, showing little change other than the odd drop in CD adoption for some projects. As outlined in books like *The Phoenix Project*, regular audits of processes and the deployment pipeline are necessary to eliminate any bottlenecks that are made apparent, since code moves at the speed of the slowest bottleneck. When these are implemented, the issues may be more readily apparent to organizations and DevOps teams.

AUTOMATION IS A MAJOR CHALLENGE

DATA

Compared to 2017, almost all the major barriers for adopting Continuous Delivery saw decreases in the past year, but we saw the biggest drops in the areas of corporate culture and

management support. Only 44% of respondents pointed to corporate culture as a major pain point, compared to 53% in 2017, while management support was less of a barrier, dropping from 30% in 2017 to 20% in 2018. The only issue that was recognized as a barrier more often in 2018 was the ability to integrate automation technologies (26% in 2017 vs. 37% in 2018).

IMPLICATIONS

Organizational culture has grown to accept DevOps practices, and management has either been educated more thoroughly on the benefits of DevOps, or has seen proven results at other organizations. However, automating as much of the pipeline as possible has proven to be difficult.

RECOMMENDATIONS

Automation may be a new major barrier, but it is an integral part of reaping the benefits of DevOps practices. Replacing manual checks to proceed through the pipeline, performance testing, and unit testing with automated tools can significantly decrease time-to-market, and move your organization one step closer to deploying on a regular basis. It's also important to consider that automation is not, by itself, Continuous Delivery. It makes adopting CD and deploying code easier, but it's only a piece of the larger puzzle.

MICROSERVICES UNDER THE MICROSCOPE

DATA

22% of respondents are using microservices in development only, while 31% are using them in development. In 2017, 29% of people were not using microservices and were not considering them, but only 13% are not doing so in 2018. There was a 6% increase of those considering microservices from 2017 (27% vs. 33%).

IMPLICATIONS

While DevOps adoption seems stagnant by comparison, microservices are immediately getting recognized by organizations as a viable way to develop and deploy applications. The need to monitor microservices will likely increase, and deployments should be faster as organizations adapt to this new paradigm.

RECOMMENDATIONS

While microservices may not be the best fit for every organization, they can help development teams focus only on coding and deploying specific parts of an application, which can be easier to work with instead of deploying the entire application again. Start implementing microservices with a project or two, and then provide management with a proposal to implement them across more projects.

As outlined in books like *The Phoenix Project*, regular audits of processes and the deployment pipeline are necessary to eliminate any bottlenecks that are made apparent, since code moves at the speed of the slowest bottleneck.

A NEW FOCUS ON SECURITY

DATA

More respondents have begun including security issue detection measures in their development process, from 29% in 2017 to 37% in 2018. More emphasis has also been placed on implementing a thorough audit trail, growing 5% between 2017 (16%) and 2018 (21%).

IMPLICATIONS

After a few years of major security breaches and security vulnerabilities, including MongoDB, Target, Equifax, and Intel, security and DevSecOps practices are getting more of a focus, as these responsibilities are shifting left to developers. Issue detection will help identify vulnerabilities before release, and audit trails help stakeholders determine where mistakes are being made to make effective improvements to processes in the future.

RECOMMENDATIONS

No organization is totally immune to security threats and vulnerabilities, but one of the best ways to fight against them are to train developers to code with security in mind, and implement more automatic security checks so that development speed is not greatly impacted. While compliance is not yet seen as a major pain point (only 14% of respondents do), it may get on developers' nerves sooner rather than later, so educating developers on the importance of compliance while giving them the tools to deal with it in a timely manner will help improve DevSecOps practices down the road.

DevOps isn't DevOps Without the Database

More and more companies and organizations are looking into the advantages of DevOps. A glimpse into the newly published 2018 Database DevOps Survey from Redgate reveals that 52% of companies have already introduced a DevOps approach to some or all of their projects, and a further 30% plan to do so in the next two years.

It shouldn't really come as a surprise. By introducing collaboration, cooperation, and communication into the development process, benefits are gained at every level, from the development teams to management.

THE BENEFITS OF DEVOPS

To discover what those benefits actually are, David Linwood, an experienced IT Director, recently undertook an MSc research project into the advantages companies expect to gain from adopting DevOps. He found the seven leading benefits are:

- The faster speed and lower cost of releases
- Improved operational support and quicker fixes
- A swifter time to market
- Higher quality products
- A lower volume of defects
- Improved frequency of new releases and features
- Good processes across IT and teams, including automation

The understanding of those benefits is also spreading. 17% of respondents to the 2017 Database DevOps Survey quoted a lack of awareness of the business benefits as a main obstacle to introducing DevOps. This fell to 12% in the 2018 survey.

THE DRAWBACKS OF SILOED DATABASE DEVELOPMENT

This is all encouraging stuff… until the database enters the picture. Many front-end applications typically have a database at the back-end to gather, store, and analyze data. Those databases are often regarded as slow moving creatures, with changes introduced late in the development cycle, and are not regarded as worthy of being included in DevOps.

The result of this siloed approach to database development isn't pretty. Respondents to the 2018 Database DevOps Survey reported that it leads to four major drawbacks:

- The increased risk of failed deployments or downtime
- Slow development and release cycles
- An inability to respond quickly to changing business requirements
- Lack of communication between development and operations

Compare these with the benefits to be gained from DevOps and a worrying concern emerges. Excluding the database means it acts as a counterweight, actively resisting the benefits of DevOps being realized, and hindering the enhanced collaboration, cooperation, and communication it promises.

What to do?

> Excluding the database means it acts as a counterweight, actively resisting the benefits of DevOps being realized.

THE WAY FORWARD

Truth is, times have changed. It's now recognized that database code can be treated in the same way as application code, provided the security and integrity of the data within the database is protected.

Tools have also emerged that enable database code to be version controlled to maintain one source of truth, and for it to be included in Continuous Integration and automated release management.

Importantly, many of those tools integrate with and work alongside the same tools used in application development and, while some processes need to change, the infrastructure remains the same, and developers can continue to work in their favored development environment.

So rather than being a blocker to DevOps, the database can become the enabler that allows every benefit to be gained for both the application and the database.

Key Research Findings

BY G. RYAN SPAIN
PRODUCTION COORDINATOR, DZONE

DEMOGRAPHICS

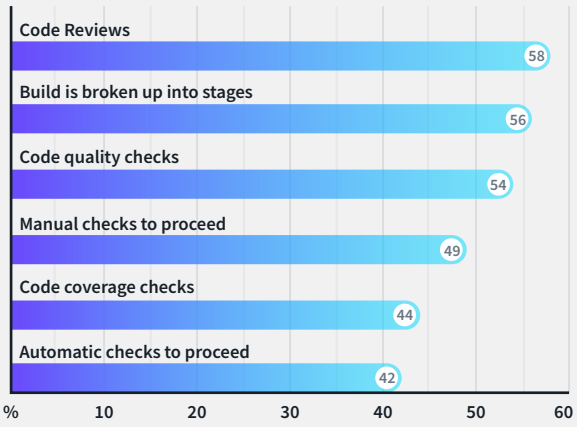
549 software professionals completed DZone's 2018 DevOps survey. Respondent demographics are as follows:

- 37% of respondents identify as developers or engineers, 20% identify as developer team leads, and 17% identify as software architects.
- The average respondent has 13 years of experience as an IT professional. 59% of respondents have 10 years of experience or more; 25% have 20 years or more.
- 36% of respondents work at companies headquartered in Europe; 35% work in companies headquartered in North America.
- 22% of respondents work at organizations with more than 10,000 employees; 21% work at organizations between 1,000 and 10,000 employees; and 24% work at organizations between 100 and 1,000 employees.
- 78% develop web applications or services; 47% develop enterprise business apps; and 29% are modernizing legacy applications.
- 82% work at companies using the Java ecosystem; 64% at companies that use client-side JavaScript; 32% at companies that use server-side JavaScript; and 30% at companies that use Python. 60% of respondents use Java as their primary language at work.

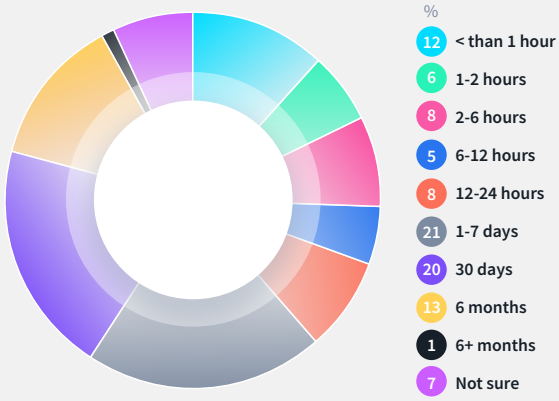
ADOPTION RATES

Compared to results from DZone's DevOps survey in 2017, it has been a pretty stagnant year for DevOps adoption. Last year

GRAPH 01. Top 6 Software Delivery Processes



GRAPH 02. Once a code change is committed, how long does it typically take to get it in the customer's hands?



we saw an increase in organizations with dedicated DevOps teams (up 7% from the year before); respondents who thought their organization had acheived Continuous Delivery for at least some of their projects went up 9%; CD pipelines across the SDLC and push-button deployments both increased (5% and 4%, respectively); the use of version control and CI tools in QA and production groups all had double-digit growth, around 15%.

This year, most of our questions around the adoption or use of DevOps practices, processes, or tools saw either statistically insignificant change or decrease (with a few exceptions discussed later). The percentage of organizations with dedicated DevOps teams only increased by 1%; respondents who believed their organization had achieved CD decreased a net 7% (respondents who thought their organization had achieved CD for all projects increased from 14% to 18%, but the respondents who believed their org had achieved CD for certain projects decreased from 39% to 28%). CI/CD pipelines for the whole SDLC and push-button deploys increased only 2% year over year, and while QA groups increased their usage of CI tools by 4% — just outside the margin of error — no other group (between development, QA, and production) had statistically significant changes in the use of version/source control, issue tracking, CI, or configuration management tools.

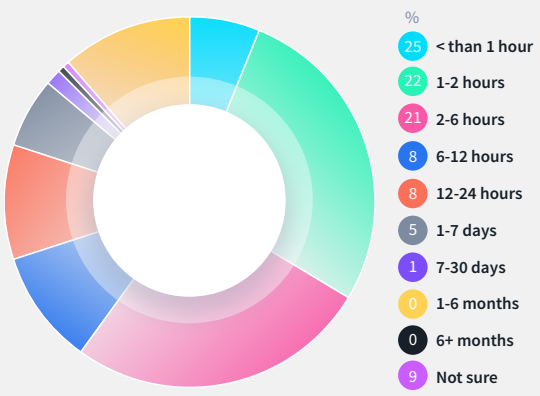
Notable changes this year revolved mostly around practices and tools that might help make DevOps goals, such as Continuous Delivery, more acheivable. We've seen, as we will describe later, fairly massive increases in the adoption of microservices architectures and container technologies, which correlate directly to DevOps practices and benefits.

AUTOMATED TESTING

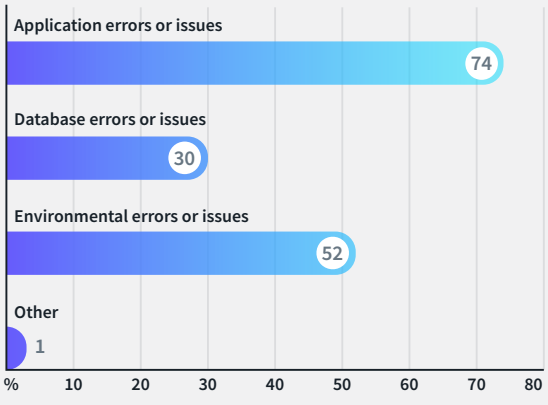
Asking respondents to let us know which tests in their pipeline are currently automated, we saw a slight (mostly insignificant) decline in the results for every single test. Notable drops were in BDD framework tests (21% to 15%) and integration tests (54% to 49%); the decline in automated integration testing is particularly interesting as integration tests are (and have been for our last 3 surveys) the most popular tests to automate. And while year-over-year changes between the 2017 and 2018 surveys may be rather minimal, there are many tests that have become significantly less automated (or perhaps even used) since 2016, including story-level tests (-8%), component/unit tests (-9%), UI tests (-5%), and post-deployment tests (-4%).

However, the benefits of automation also show themselves in our survey results. DevOps practices such as automatic checks to proceed, automated performance testing, automated featured validation, and automated validation of database code correlate with around a 13% increase over average for a 2-hour-or-less mean time to recovery (MTTR)–or 14%, 10%, 16%, and 11%, respectively. The number of tests an organization is able to automate, though, is in part related to the size of the organization: larger organizations are able to automate more tests. While companies under 100 employees showed an average of 2.06 tests automated based on our survey results, companies over 10,000 employees averaged 2.80 tests. While this difference may seem minor, it represents the largest companies being able to automate 6% more of their testing than the smallest, based on our questions on the automation of 11 different possible tests. Taking the top 5 most popular automated tests overall brings this to an 8% increase between small (under 100 employees) companies and large (10,000+ employees) companies.

GRAPH 03. How long does it usually take to restore service when something goes wrong in production (i.e. Mean Time to recovery)?



GRAPH 04. What is the leading cause for rollbacks or hotfixes?



DEVOPS PROCESSES

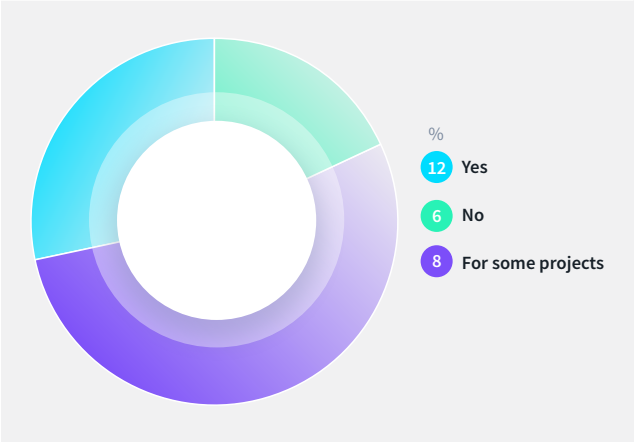
While the adoption of certain DevOps practices and processes has remained flat, the impact of what has been adopted shows pretty clearly. First, a few areas in this discussion did see some growth from last year; security issue detection increased by 8% (29%-37%), supporting the concept of DevSecOps; the practice of creating thorough audit trails increased 6% (16-22%); and the percentage of respondents saying their pipeline involves manual checks to proceed decreased 7% (56%-49%), indicating a shift towards automated alternatives to manual tasks.

The benefits gained through these DevOps processes are in line with the benefits from automated testing: performance issue testing, overall visibility to anyone in the organization, thorough audit trails, code quality checks, code coverage checks, and code reviews all showed an MTTR greater than 10% of what respondents without these practices in their SDLC showed, on top of the automated tests and validations mentioned earlier. While none of these processes are inextricably linked to respondents' beliefs that their organization has acheived Continuous Delivery, respondents using any one of these were, on average, 14% more likely to say that their organization had completely acheived CD (see graph for details).

MICROSERVICES

Microservices architectures have been a hot topic for years now, but adoption of microservices has been less frenetic. In 2016, 38% of respondents of our DevOps survey said their organization was using microservices in some capacity. In 2017's survey results, we saw this increase to 44%. This year's survey showed more than half (54%) of respondents using microservices — consistent with the results of the recent survey we conducted for our Guide to Microservices, where 53% of respondents said their organization was using microservices architectures for its applications.

GRAPH 05. Do you believe your organization has achieved Continuous Delivery?



Respondents saying their organization uses microservices were much more likely to believe their organization had acheived DevOps goals. Specifically, 74% of respondents using microservices architectures believed their organization had acheived Continuous Deliver, as opposed to 46% not using microservices; and 66% of microservices users said they believed their organization had acheived Continuous Integration, versus 41% of respondents not using microservices. Also, respondents using microservices architectures were much more likely (between 10-26%) than those not using microservices to say they broke their builds into stages (15% more likely); practiced performance and security issue detection (11% & 15%); automated performance testing (10%); conducted dependency, compliance, and code quality checks (13%, 12%, 26%); practiced code reviews (12%); and automated their checks to proceed with code commits and deployment (19%). 66% of respondents performing push-button deploys also said they were using microservices.

CHALLENGES

A few CD pipeline pain points seemed to get less painful in the past year. Responses noting requirements issues as a pain point dropped 4% (23% to 19%), and challenges with regression testing (32% to 23%), performance testing (27% to 21%), and user acceptance testing (31% to 24%) all fell. The biggest CD pain points, however, did not see significant changes from last year. Environment configuration and setup is still the most common challenge, and only dropped 1% from last year (56% to 55%); the second-place pain point, coordination of team members and resources, also only fell 1% (34% to 33%).

Respondents believing their organizations have not acheived Continuous Delivery showed that barriers to entry for CD are shifting. While corporate culture–the most common barrier from 2017–remained at the top of the list, it dropped 8% from last year's results (53% to 45%). Support from management as a barrier also decreased 8% from last year (30% to 22%), and budget as a barrier fell 5% (27% to 22%). The barrier of integrating automation technologies into the SDLC, on the other hand, increased drastically from 2017, increasing from 26% to 40%.

Without DevOps Intelligence, There is No DevOps

Companies are spending millions to adopt DevOps and Continuous Delivery to drive their digital transformations. Yet most still can't demonstrate a clear ROI for their initiatives. What's going on?

The growth of DevOps in large enterprises is exposing the painful reality that these organizations are unable to track appropriate performance and business metrics across their delivery pipelines. Without a clear understanding of what's working and what's not, companies are having a hard time improving and scaling their DevOps initiatives in a way that has the most positive business impact.

But these problems aren't inevitable. They're just a sign that it's time for the next frontier of Continuous Delivery — using DevOps Intelligence to analyze the complete delivery pipeline and make targeted improvements.

A DevOps pioneer, XebiaLabs has been central to the DevOps transformations of major brands worldwide. Its industry-leading DevOps Platform helps large organizations accelerate the delivery of high-value software at scale, even within the most complex, regulated, and diverse environments.

XebiaLabs' latest addition to its DevOps Platform, XL Impact, is the first DevOps Intelligence solution to use goal-based KPIs and predictive analytics to help companies strategically build, track, and adjust roadmaps for their DevOps transformations.

While traditional reporting or dashboarding tools simply show disconnected data without context, XL Impact combines DevOps best practices with historical analysis, machine learning, and data from across the toolchain to highlight trends, predict outcomes, and recommend actions. Guided by XL Impact's integrated KPIs and predictive analytics, enterprises can optimize software delivery, mitigate release risk, and drive ROI for their DevOps initiatives.

No more guesses or gut feelings. DevOps teams can finally put their energy in the right places from the start, using hard data to highlight the best decisions for a successful transformation.

WRITTEN BY DEREK LANGONE
CEO, XEBIALABS

PARTNER SPOTLIGHT

XebiaLabs DevOps Platform

Top-ranked intelligence, automation, and control for Enterprise DevOps

XebiaLabs Enterprise DevOps

CATEGORY

Continuous Delivery and Application Release Automation

RELEASE SCHEDULE

Continuous

STRENGTHS

- Supports the complex needs of enterprise teams, including automated application deployment, advanced release orchestration, and DevOps Intelligence
- Brings DevOps methods to all apps no matter the technology – public, private, and hybrid cloud, container, microservices, mainframe, and web
- Seamlessly works with what you have, and brings together your entire software delivery toolchain, including Jenkins, Puppet, Ansible, Docker, Jira, and many more
- Architected for enterprise use, with its acclaimed model-based and agentless deployment technology, high scalability, dual-mode DevOps, and goal-based DevOps KPIs

CASE STUDY

Major companies, like Expedia, GE, NASA, Paychex, and KeyBank, use the XebiaLabs DevOps Platform to scale DevOps across hundreds of teams and thousands of applications while drastically improving speed, efficiency, and quality. Expedia chose XebiaLabs to help increase deployment speeds and streamline processes while operating at scale. They increased release velocity by 20% in month one, 33% by month two, and later reached a 50% increase. GE uses XebiaLabs to speed deployment and control the software delivery pipeline. In year one, they saw a 2X return on their release automation investment, and releases that took months now take days. Paychex uses XebiaLabs to ensure consistency and visibility across environments and approaches. Their average deployment time decreased by 53% and production deploy success rate improved by 23%, while the total number of software deploys increased 656%.

NOTABLE CUSTOMERS

- GE
- NASA
- TD Ameritrade
- KLM
- Paychex
- Xerox

WEBSITE xebialabs.com | **TWITTER** @xebialabs | **BLOG** blog.xebialabs.com

Measuring in a DevOps World

QUICK VIEW

01. Use a Big Data and analytics approach to your delivery pipeline

02. Leverage real-time dashboards wherever possible

03. Get data out of their native applications into a common place

04. Critical review impact of DevOps on SLAs and KPIs

BY MIRCO HERING
MANAGING DIRECTOR - APAC AGILE & DEVOPS LEAD

It is quite easy to find articles and books about DevOps practices like Continuous Delivery out there, and there seems to be general agreement on which practices are good. In the measurements space, I am not so sure that we have found a general consensus yet, and there are a few very counterintuitive things happening that challenge our traditional approach — just like Agile and DevOps did to project management best practices.

In this article, I want to address two aspects of measuring in the DevOps world to help navigate this space: the technical dimension of how measurements should be done and some of the challenges associated with them, and then, how to look at some common measures with a new set of eyes.

For years, we in IT have been speaking to businesses about the use of analytics and big data to make them more effective by leveraging the insights coming from that data. Yet, when we look at what we have done in IT, we can see that most of the data we create is not being analyzed with the same mindset. Think of all the data that is "hiding" in your Agile lifecycle tool, your Continuous Integration server, your code quality tools, etc. You are likely looking at that data within the context of those tools but don't have a formal way to correlate the data and derive analytics from it. That data is mostly "wasted" without its full value being realized — it's "dark data."

The answer, of course, is to use the practices we have been recommending to business ourselves: collect all the data, visualize it through a dashboard, and use analytics and machine learning to improve performance. My team has been using a Splunk-based solution to do this, and I know that Capital One has **open-sourced their Hygieia dashboard solution** for the community to use. I am surprised that there are not many more solutions for this problem easily available. A key challenge is, of course, that pretty much every organization is using a diverse tool stack.

The challenge that we often have to overcome while collecting data from our IT delivery tools is that many tools do not easily part with their data. Some of the data is only accessible within the tool itself, and the database structures that sit behind it are convoluted and not intuitive to query (or maybe not even possible to query). I think tool vendors have some work to do here to make this a more straightforward process for us in the community. In the meantime, we can use hooks and logging mechanisms to write important events and data out into a place that can be accessed by our analytics solution. There is no reason why your Continuous Delivery pipeline cannot write out the results after

each step with as much metadata as possible so that we can use that as input into our analytics solution later.

The data that we create within IT will not grow that quickly, which means we can be quite generous with the amount of logging that we do. Imagine what we can do when we have collected data from many years of work, hundreds of releases, and thousands of application builds and test runs. You can start to validate your assumptions about IT delivery. Did the quality of our software improve when we increased unit test coverage? Did late changes in requirements cause higher production outages? Do teams with high amounts of build failures have a higher chance of delaying their deployments? These are all questions that in the past we answered with examples or intuition but that we will now be able to answer with real data.

There is one other aspect that we can start to address through this approach, as well: status reporting. I cannot believe that the main way of looking at status continues to be through PowerPoint and Excel. The data in those documents is at least a few hours old (if not several days) and has gone through many hands and interpretations before it is presented. We should be able to look at real-time status by gathering information straight from the systems we use, which is something my team has started doing and is something I highly encourage you to do, too. It will change the way you see status. It will force a rigor across your processes and systems that will improve data quality and transparency for everyone. It will take a while to clean the data up because you cannot "fudge" things anymore, but it pays back later when you can always get accurate data from your system.

What else can we do with all the data that we now collecting for our dashboard? You can start using some basic machine learning and data analysis to look at ticket data, for example, and all those requests that your team is dealing with. Of course, the quantitative data is important for measurement, but the content of the tickets can also be used. You can analyze where the volume of work is coming from and identify opportunities for automation. With machine learning, you can derive from the text where to route the ticket to and thus improve resolution times because you are avoiding ticket ping-pong between teams. And as you automate services and expose them as self-service capabilities, your team will start to have more and more time to focus on improving automation — a virtuous cycle that you just have to start. It will take off and feed itself.

There are some surprising things you need to consider when you go down the path of DevOps measurements that are worth exploring. One of the things that most people measure is compliance with Service Level Agreements (SLAs), which is a deeply conflicted space. Assume your team has only two SLAs: two hours for a password reset and 24 hours for a Sev 2 incident in production. When both happen at the same time, which one do you think your team will do first? Which one is more important to the company? The two answers are, unfortunately, often conflicting. You can see how this quickly becomes a problem with more SLAs; it creates a complex system with outcomes we cannot predict.

Measuring KPIs and trends is much better than adherence to an SLA because we want to see that we are becoming better for each category of work. But even here, automation throws us a curveball. A common thing to measure is first-time resolution rate and time to resolve a ticket. The first one, we want to see increase, and the second one, we want to see to go down as we get better. As we are moving more towards self-service capabilities, the easy tasks will be automated, which means only the more complex and difficult problems require a ticket that the team has to resolve. But this means that we will need more time to resolve those — and the chance of getting it wrong with our first attempt increases. As a result, our time to resolve and first-time resolution rate will look worse as we increase the levels of automation. If you don't prepare your management for this counterintuitive situation, you might get into trouble.

I think there is tremendous opportunity in this space and we are just at the beginning of our measurement story. As we get better at this, we will have more meaningful conversations and will learn a lot more about what works and doesn't work. We will move from intuition to a more scientific discussion on DevOps. I gave some examples of legacy ways of looking at measures that have to change, and I am sure we will learn about many more. Let's be open-minded and curious to find new ways to look at the data we have.

MIRCO HERING leads the Accenture Agile and DevOps practice in Asia Pacific with focus on Agile transformations, DevOps and Test Automation. He has for over a dozen years worked on accelerating software delivery through innovative approaches. He started using Agile as a last resort when one of his projects was faced with ever-changing requirements resulting in frequent updates to the project plan. Adopting an Agile methodology turned the project around, and he has promoted Agile principles since. He shares his experiences here and at international conferences. He recently authored the book "DevOps for the Modern Enterprise."

Datical

DevOps, Meet Database

Release Faster or Die.

Database Release Automation

---

www.datical.com

DevOps Meets Database

Software is reshaping every industry – from healthcare, to finance, to retail, and beyond. The pace of change is accelerating, and companies must master their application release process in order to survive in the digital era. In the software economy, innovation and speed are the new standards.

As companies have adopted automation to help increase the pace of software delivery, a new constraint has emerged: data. The database has long been a unique, separate discipline, not part of the DevOps tool chain. In fact, most companies — even those are using the latest DevOps automation tools — are still managing and deploying database changes manually, anchoring development teams.

That's why we developed Datical. Our database release automation solution is designed to break the application release logjam created by today's manual database deployment processes by bringing DevOps to the database. Datical offers a great opportunity to improve productivity and performance, allowing development, test, and DBA staff to focus on more important projects and initiatives. With Datical, application teams can forecast the impact of database schema and logic changes prior to their deployment, eliminating the possibility of promoting bad changes to Production. Put simply, database release automation from Datical helps speed the delivery of better performing applications into production faster, safer and with more reliability.

Data is the toughest problem to solve in DevOps thus the database has emerged as the newest DevOps constraint

Datical is strategically guiding some of the world's most admired companies through their DevOps database efforts and delivers business innovation through database automation. Learn more here: **A Guide to Digital Transformation, Enterprise Applications and the Database**.

WRITTEN BY BEN GELLER
VICE PRESIDENT MARKETING, DATICAL

PARTNER SPOTLIGHT

Datical

Our goal was to decrease the release cycle of our applications from 5 weeks to 2. That was impossible because it took 2 weeks just to do the database portion.

CATEGORY
Continuous Delivery and Release Automation

RELEASE SCHEDULE
Continuous

OPEN SOURCE?
No

CASE STUDY

While one of the world's leading entertainment and media companies had been successful in bringing DevOps Automation to many of the manual processes in modern software development, the same was not true for the data underpinning the application. Aligning database changes to the application changes they support was a manual process that added considerable friction to the application release process and slowed the pace of innovation. The DevOps team knew they had to automate their database deployment process alongside their application release process in order to achieve optimal DevOps. The team now uses Datical to bring DevOps to the database. As a result they have been able deliver innovation faster by decreasing the release cycle of their applications from 5 weeks to 2 weeks.

STRENGTHS

- Treat database code just like application code. Create database Continuous Delivery by unifying application and database changes.
- Provides application developers with instant feedback on database code.
- Simulates the impact of database changes before they are deployed.
- Automatically tracks and reports the status of every database deployment across the enterprise.

NOTABLE CUSTOMERS

- NBCUniversal
- Nike
- LabCorp
- Sony
- AIG
- GE Transportation

WEBSITE datical.com | **TWITTER** @datical | **BLOG** datical.com/blog

Eleven Continuous Delivery Anti-Patterns

BY BADRI N. SRINIVASAN
AVP AND ENTERPRISE AGILE COACH AT **SOCIETE GENERALE GSC**

QUICK VIEW

01. Continuous Delivery (CD) is a design technique that is used in software engineering to automate the process of software delivery.

02. CD practices facilitate the ability of the organization to ship software quickly.

03. This article focuses on the anti-patterns that are prevalent while implementing CD and which need to be addressed for an organization to implement effective CD practices that enhance agility.

Continuous Delivery (CD) is a design technique that is used in software engineering to automate and enhance the process of software delivery. Software is considered to be notoriously difficult to ship, and CD practices facilitate the capability of organizations to quickly and reliably push software features, enhancements, or bug fixes to production that result in improved business value for the customer. This article focuses on some of the important anti-patterns that are prevalent while implementing CD and which need to be addressed for an organization to implement effective CD practices to enhance business and organizational agility.

The focus of Continuous Delivery is to enable product teams to get quick feedback regarding the state of the software product at every stage in the CD pipeline, and thereby enable quicker releases to the production environment based on market requirements. CD improves what and when you release to the customer. Hence, every product team in any domain and industry (e.g. healthcare, avionics, pharmaceuticals, retail, banking) can and should practice Continuous Delivery, and the team should make it a priority item to build a CD pipeline during the beginning of their project so that they are able to release high-quality software to the customer in the shortest possible time and build their competitive advantage in the market. CD also enhances customer and employee satisfaction as the routine and repetitive work is automated and the product is also brought to the market rapidly.

However, even the best organizations can get trapped in CD anti-patterns during the process of implementing CD practices and trying to release quickly. They may apply engineering best practices that are suited for traditional software development lifecycle models, or improperly apply CD patterns which may not yield the appropriate outcomes. A pattern is an acknowledged, effective way to find a solution to a problem, while an anti-pattern is the exact opposite, i.e. it is known to be an ineffective way to resolve a problem. Patterns are generally considered as best practices, and anti-patterns are generally considered as counter-productive practices. The aim is to focus on the patterns and minimize the occurrence of anti-patterns.

The most important CD anti-patterns that need to be avoided are explained below:

1. **Long and slow deployment pipelines** – There are a lot of steps between code commit and release. The deployment pipeline is serial and very long, which leads to a slow pipeline. The aim should be to focus on short and wide pipelines. This enables faster release times.

2. **Rigid pipeline design** – If the design of the pipeline is rigid, it prevents the team from optimizing their processes. In some cases, the team might want to do some things in parallel and not serially, e.g. exploratory testing and acceptance testing. Hence, pipelines should be structured such that they are flexible and can incorporate sudden changes and the team should be able to manage the pipeline to meet these requirements in real time.

3. **Inadequate understanding of CD practices** – Inadequate understanding of CD practices may lead to a poorly designed CD pipeline that could lead to issues in the future. Hence, under-

standing the concepts behind CD practices through reading books and blogs, watching training videos, and other activities are useful for implementing these CD concepts through the appropriate design and structuring of the pipeline, which facilitates the reduction of issues during CD implementation.

4. **No capability to track metrics (logging/application)** – This leads to ineffective decision making as no one is aware of the issues that are causing bottlenecks which impede effective delivery. The focus should be on tracking application metrics and performing detailed and cumulative logging to enable effective decision making.

5. **Anti-CD mindset** – The team is not interested in CD and asks, "Why do we need CD? We are doing fine, and we don't need CD for any additional improvements, as we do not release to production often." The focus should be on improving awareness of CD practices and explaining success stories and case studies to the team so that they can understand the critical benefits of CD and how it will help them reduce the time to market (TTM) for their product. Even if they do not release to production, they could focus on releasing to the simulation/staging environment and into production when required.

6. **Investment in build and deployment pipelines is minimal, and CD tools are implemented just for the sake of implementing them** – There is no significant investment made in building and deploying CD pipelines, which will lead to issues in the future when the practices mature and need to be scaled, as the CD foundation is not strong. The focus should be on building infrastructure capabilities, versioning approaches, and evaluating new techniques. Based on the requirements, one team member may undertake these activities to maintain the pipeline and implement other CD practices. The focus should be on continuously re-architecting the Continuous Delivery pipeline and design based on the project and market requirements (this is based on Kaizen – a system of continual incremental improvements).

7. **Operational focus is not addressed appropriately** – There is no appropriate focus on the operational features (features which impact the visible features of the product). The focus should be on a single backlog which contains all the types of features including operational and visible features.

8. **Database changes are left out or not done properly** – Database updates and changes are not managed appropriately and this affects the final delivery of the product. The focus should be on using a tool to manage the database changes with appropriate versioning practices.

9. **Limited DevOps collaboration and organizational culture** – There is limited collaboration between the development and operations teams. Team members work in silos and the organizational culture is not conducive to collaboration. CD practices promote collaboration among all the sub teams to facilitate the design and creation of the CD pipeline. Limited collaboration between Dev and Ops leads to issues and defects in production. The focus should be on increased DevOps collaboration, an agile mindset, and an open and transparent organizational culture to maximize effectiveness.

10. **Usage of design techniques like microservices and containers but without the safety harnesses like unit tests and metrology tracking being followed as good practices** – Before introducing design techniques like microservices and the use of containers, the focus should be on ensuring that a good and robust infrastructure to track metrics and unit tests as safety harnesses is being used appropriately in the project. This will help the team to build on these basic practices in the future.

11. **Other anti-patterns which cumulatively lead to a deficiency in the implementation of CD practices** – These include a focus on the project instead of the product, customer voices not being heard or utilized to effectively resolve problems, requirements scope creep, the focus on only a subset of CD practices like automation, or a focus on tooling compared to a focus on systems engineering. In order to alleviate these anti-patterns, we must focus on the product, customer feedback, a manageable product scope, and systems engineering to ensure the effective implementation of CD practices.

In the future, with the advent of artificial intelligence, data science, machine learning, prescriptive and predictive analytics, and the integration of these technologies with Continuous Delivery models, many of these CD anti-patterns will be managed more effectively through rules, data sets, self-learning, and inference engines.

Hence, while implementing CD practices in any organization, we must be aware of, and able to identify, the important CD anti-patterns so that we can avoid or minimize them, and focus only on CD patterns that maximize organizational and business agility and improve customer satisfaction.

BADRI N. SRINIVASAN is working as an AVP and Enterprise Agile Coach with Societe Generale GSC, Bangalore, India. He has 20+ years of experience with a focus on agile, process improvement, and organizational change management processes. His extensive experience includes coaching Scrum Masters, product owners, and product teams on agile practices and facilitating Continuous Delivery and DevOps.

DevOps + Automation is the key to high velocity app delivery.

With Chef, you can:

- Detect state of your apps and environments.
- Correct issues to improve performance and security.
- Automate deployments faster with reduced risk.

Learn how at https://www.chef.io/devops/

[70% of the Fortune 500 automate their apps and platforms with Chef.]

CHEF™

Hybrid IT: The Way of the Future (and the Past!)

New technology is exciting. From containers, Kubernetes and serverless to machine learning and IoT, it always seems like there's something new to learn and potentially a bandwagon onto which to leap.

A big mistake, though, is assuming your company will be "all-in" on something in a short period of time. The reasons are twofold. First, the pace of disruptive, technological innovation will always outstrip your ability to execute. And secondly, you have a business to run, with existing applications deployed in working environments even if they aren't using the newest, coolest technology. It's easy to make fun of mainframes, but that software works and serves critical business functions today. If it didn't, it would have been rewritten long ago. Hybrid IT is not only here for the foreseeable future; it's also been our past.

Even though IT environments are heterogeneous and can comprise anything from mainframes all the way to functions-as-a-service, we can still derive significant benefit from automation to help us manage things uniformly. The same tasks — like keeping systems stable, ensuring security & compliance, while making changes to applications – are necessary no matter what your operating environment.

The Chef open-source configuration management project pioneered this concept in infrastructure automation, which provides a common programming language for automating system tasks like installing packages and managing services across diverse operating systems, clouds, and data centers. We've extended this concept to compliance, with an open-source project called InSpec that allows for previously-opaque compliance rules to be expressed as code. And finally, we extended it into applications, with Habitat, a project that allows you to package an application independent of its underlying infrastructure and give it portability to run anywhere.

Hybrid IT means we end up owning a little bit of everything. Automation, expressed as code, allows us to adapt and apply the same mental model to every type of environment, thereby making it easier for engineers to perform the tasks they need to ship software at velocity. You can learn more about the benefits of automation in a hybrid environment at **chef.io/solutions/infrastructure-automation**.

WRITTEN BY JULIAN DUNN
DIRECTOR OF PRODUCT MARKETING AT CHEF

DevOps vs. Siloed Cultures

BY JOHN VESTER
SR. ARCHITECT AT **CLEANSLATE TECHNOLOGY GROUP**
FREELANCE WRITER AND ZONE LEADER AT **DZONE**

QUICK VIEW

01. A siloed culture introduced metaphorical blinders, creating boundaries and walls between application developers and application administrators.

02. A DevOps culture realizes that DevOps is another Agile team, working in a small group tightly coupled with the applications they support.

03. CI/CD concepts and a "* as Code" approach allows DevOps to automate the deployment process, leveraging the same pipeline to execute on the developer's workstation that will ultimately be used for the production release.

The emergence of DevOps philosophy has introduced a welcomed manner in the way Information Technology (IT) professionals view the gap between development and deployment. Coupled with Continuous Integration/Continuous Deployment (CI/CD), implementation of a DevOps mentality can streamline efforts which were tedious at best for teams which utilized legacy deployment routines. As DevOps gains momentum, it is important to understand the DevOps culture and how it differs from alternative strategies.

THE SILOED CULTURE

In order to gain an appreciation of the DevOps model, providing a backdrop of a common deployment approach can help illustrate the issues that were faced for years — as application development teams built applications and interacted with application server administrators (or web administrators) to deploy the result of their development efforts.

Consider this comic. Under the model illustrated above, there was a clearly defined separation between the role of the application developer and the responsibilities of the application administrator. The development team would finish their development and unit testing, then toss the resulting application over to the application administrator, who was responsible for deploying the application properly in order to pave the way for testing and final validation.

On one side, the development team wore metaphorical blinders to limit their focus to the application functionality, with the underlying mechanics of the application server foreign or out-of-sight to them. In contrast, the application administrator was focused on getting the application deployed and available, wearing a similar set of metaphorical blinders to deflect any issues or bugs with the underlying program code.

To the development team, their primary objective was getting their application deployed — unaware of all the other applications the application administrator had to maintain. In fact, the expectation of the application administrator was that they almost had a photographic memory — remembering everything they did last time to make the deployment successful, plus keeping track of additional requirements based upon drive-by conversations which occurred days (if not weeks) earlier.

As one might imagine, this model didn't seem to work out well, especially as the ratio of applications/development teams increased for a given application administrator. Not only were bottlenecks commonplace, but the ability to understand every supported application became problematic...if not impossible.

INTRODUCING DEVOPS

Adoption of the Agile software development introduced improvements to the way applications were designed and built. While focusing on shorter iterations, or sprints, smaller teams could work directly with a business representative to deliver a series of features and functionality. This process alone delivered value to the business faster, in contrast to other development lifecycles which provided a larger delivery over a longer time period.

The concept of Continuous Integration/Continuous Deployment (CI/

CD) introduced the desire to release updates or features on a continual basis. With the Agile teams delivering features on a two- or three-week basis, the aspect that was initially missed was the deployment of the unit of code that was ready for use. Application administrators had worked in a model where a large release would happen on a quarterly (or longer) basis, allowing the ability to support multiple teams. With the adoption of Agile and CI/CD, the workload for the application administrator was increasing, since each team had something new to deploy on a monthly basis — if not sooner.

DevOps became this new idea that broke down the walls between the application developers and the application administrators. The realization of CI/CD demanded a new model. As a result, DevOps became the union of Development (Dev) and Operations (Ops). Instead of being outside the Agile process, the DevOps team would become part of the Agile process, participating in the planning stage, procuring tools and technologies to make CI/CD a reality and providing a level of monitoring for the applications under their support.

* AS CODE IS BORN

In the days of the comic illustration (above), the process to prepare a release for deployment often required a series of manual steps. These steps would range from validating dependencies, creating some type of archive that could be deployed, making configuration changes to match the current environment, and validating the application deployed as expected. Since a majority of these tasks were manually driven, the potential for an issue to occur during deployment was higher than desired, with challenges centered around debugging scenarios where an unexpected error or failure occurred.

As a part of the CI/CD implementation, the idea of "* as Code" was born. The notion behind this concept is to rid the deployment of manual steps and embed those steps into a script that can be automated and executed when desired. In fact, with application container technologies, a CI/CD pipeline can be established that executes on code check-in and will accomplish all the manual steps noted above — including starting and validating the application is running. With a successful CI/CD pipeline in place, the same steps that are executed on the developer's workstation are ultimately executed in production, largely removing the challenges with a deployment model based upon a series of manual tasks.

THE CULTURE OF DEVOPS

With this understanding of DevOps and how this role fits into an Agile lifecycle, it is important to understand the culture behind DevOps. Unlike the Siloed Culture represented in the comic (above), DevOps teams are simply another Agile team. They work in small groups, tightly coupled with the applications they support, moving fast to support the needs of not only the developers introducing features, but the customers utilizing the applications being deployed.

The DevOps team is an extension of the Agile team, focusing on continuous improvement of the process and walking away from the previous "us vs. them" mentality. In fact, in the Agile world, the metaphorical blinders must be removed. Those same metaphorical blinders are removed from the DevOps team as well.

The culture of DevOps also bridges the gap between the application code and the deployment code. Since the "* as Code" concept is in place, everyone on the team has the ability to view the underlying application code and the deployment code, which is called via an established CI/CD pipeline (that is also stored in a human-readable format). As a result of DevOps, a culture is born where silos no longer exist.

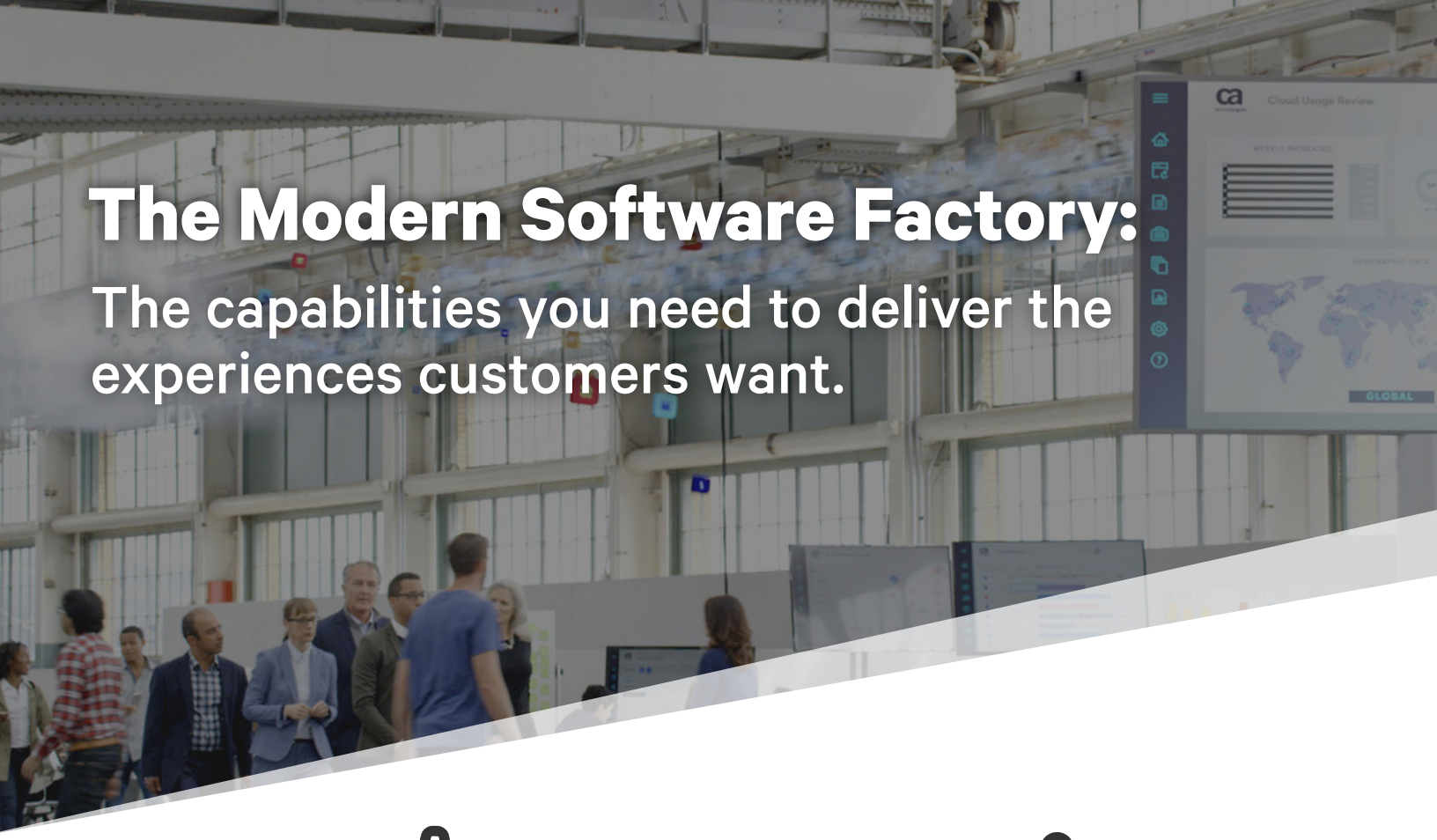
CONCLUSION

For years, if not decades, a siloed culture had success within IT. Often paired with long-running Waterfall development projects, the application administrators were able to maintain the workload to manually deploy application code — since releases were quarterly, yearly, or even longer. However, as development teams began to utilize Agile practices, the ensuing workload on the application administrator became unmanageable and problematic.

These deployment teams soon realized a culture shift was required to meet the new demand for their skills. With the idea of CI/CD gaining momentum, the concept of DevOps was born, changing the way in which teams worked together to deliver applications in a code-driven, standardized manner that could be reproduced in every aspect of the development lifecycle.

The change which complements a successful DevOps implementation provides teams who work together (thus avoiding the blame game) and participate in Agile ceremonies with the side benefit of continuous improvement across the IT landscape.

JOHN VESTER is an Information Technology professional with 25+ years expertise in application development, project management, system administration and team management. Currently focusing on enterprise architecture/application design/continuous delivery utilizing object-oriented programming languages and frameworks. Prior expertise building Java-based APIs against React and Angular client frameworks. Additional experience using both C# (.NET Framework) and J2EE (including Spring MVC, JBoss Seam, Struts Tiles, JBoss Hibernate, Spring JDBC).



The Modern Software Factory:

The capabilities you need to deliver the experiences customers want.

AUTOMATION
Delivering greater velocity with reliable quality

AGILITY
Providing faster times to market

INSIGHT
Ensuring constant improvement through insights gathered from data

SECURITY
Guaranteeing safe, frictionless access

For more information or product demonstration please visit www.automic.com

ca technologies | Automic™

Automic, the leader in business automation software owned by CA Technologies, helps enterprises drive competitive advantage by automating their IT and business systems - from on-premise to the Cloud, Big Data and the Internet of Things. With offices worldwide, Automic powers 2,700 customers across all industry verticals including Financial Services, Manufacturing, Retail, Automotive and Telecommunications. More information can be found at www.automic.com.

Intelligent Automation for the Modern Software Factory

Today we are living in the application economy era, where every business is now a software business. While this is exciting, it also presents an array of unprecedented challenges.

Not only must the business be able to build and release software quickly, but they must be able to do it more frequently - and with no delays or disruptions. Ultimately, modern businesses need to streamline traditional methods of delivery and navigate bottlenecks, while also finding ways around the inefficient use of resources.

Successful companies need to focus on four key areas when looking at how they build and deploy new business applications:

- **Automation** – delivering greater velocity with reliable quality
- **Agility** – providing faster times to market
- **Insight** – ensuring constant improvement through insights gathered from data
- **Security** – guaranteeing safe, frictionless access

"Major new automation capabilities have been introduced with CA Automic V12.1 to help you cope with an ever-changing digital landscape."

The end goal is to eliminate the barriers between ideas and outcomes. This is where the Modern Software Factory and CA Automic V12.1 come into play. It should be designed to adapt to both market disruption and customer demand. It is where your company needs to be. CA Technologies can help you get there.

WRITTEN BY CHRIS BOORMAN
CMO, CA AUTOMIC

PARTNER SPOTLIGHT

CA Automic v12.1

CA Automic 12.1 release includes three major themes: intelligent automation, modern software factories, and agility for Ops.

CATEGORY

Release Automation / Workload Automation / Service Orchestration / DevOps, AgileOps

RELEASE SCHEDULE

Annually

OPEN SOURCE?

Yes

CASE STUDY

With CA Automic Workload Automation, SAIF, a not-for-profit workers' compensation insurance company, eliminated graveyard batching shifts, freed up valuable employee capital, and cut down transfer times by 75% (quadruple throughput). They used CA AWA to automate its laborious data entry tasks, increasing efficiency and reducing operational costs. Automating the generation of payments to run hourly rather than daily has enabled the company to ensure workers receive their prescriptions and benefits faster.

STRENGTHS

- V12.1 provides improved UI/UX
- V12.1 allows for scalability
- V12.1 provides enhanced cloud-readiness
- V12.1 enables zero-downtime for upgrades/updates

NOTABLE CUSTOMERS

- ING Bank
- SAIF
- 84.51°
- University of Florida
- Electric Supply Board
- TASC

WEBSITE automic.com | **TWITTER** @CAAutomation | **BLOG** automic.com/blog

The Complexities of Continuous Integration, Continuous Delivery, and Sprint Planning

BY JEFFREY LEE
TECHNICAL CONTENT WRITER AT **PARTICLE**
ZONE LEADER AT **DZONE**

QUICK VIEW

01. Explores the origins of CI and CD in relation to agile methodologies

02. Explores the relationship between culture, CI/CD, and sprint planning

03. Best practices for sprint planning, continuous integration, and Continuous Delivery

04. Explores the challenges of sprint planning and Continuous Delivery

Continuous Integration (CI) and Continuous Delivery (CD) are complementary development processes that allow product teams to continuously release new software. Continuous Integration is the practice of frequently merging code to a central shared repository, triggering automated builds and tests. Similarly, Continuous Delivery is the practice of frequently releasing software in short cycles, typically by automating deployment builds and harnessing a CI pipeline. In practice, these development processes work together to allow teams to build, test, and deploy their code rapidly, reliably, and repeatedly.

The origins of both CI and CD can be traced back to Agile methodology, which is an iterative and incremental method of product development that focuses on collaboration and continuous releases. The backbone of Agile is the Scrum framework, a form of software development management that focuses on building products that fit business needs. Scrum breaks actions into time-boxed iterations called Sprints, which are timed intervals that break down work into manageable tasks. Sprint planning sessions are held to agree on the scope of work and includes work that needs to be completed for product backlog items. However, implementing CI and CD in an Agile-driven organization is one of the greatest challenges facing organizations.

As we have seen, the success of DevOps entirely depends on company culture. Internal teams must be able to adopt cross-functional methods to make sure software is iterated with a continuous cadence but also complements marketing and sales campaigns. Seeing how CI/CD are major components of the DevOps ecosystem, they are not excluded from this culture factor. The product that comes out of the CD pipeline needs to suit the needs of the organization. Otherwise, those who are not part of the deployment team will never support Continuous Integration efforts. This is where Sprint planning comes to the rescue.

SPRINT PLANNING

Sprint planning typically consists of fixed length Sprints of one, two, or four weeks. In these periods, teams set specific goals and tasks they wish to accomplish. This helps the organization and teams prioritize features to release. Teams can also parallelize feature development by allowing multiple teams to develop complementary features and then use CI/CD to merge, test, and deploy. CI and CD are development processes that free Sprint planning of artificial process constraints. For CI, code is merged frequently and errors are caught before deployment builds take place. For CD, production-ready code can run through automated deployment builds for fast releases and user testing. Both of these processes enable development teams to keep pace with their Sprint cadence.

BEST PRACTICES

To ensure your CI and CD pipelines are properly configured, here are some best practices:

A effective Continuous Integration pipeline embraces:

- Everyone takes responsibility for the CI build.

- Every team member knows how to check the CI build results.
- If the CI build breaks, teams work together to figure out who should fix it.

A performant Continuous Delivery pipeline embraces:

- Teams taking ownership of the process.
- Monitoring and measuring.
- Reacting to problems quickly.

A good Sprint planning meeting allows teams to:

- Discuss problems and readjust for the next sprint cycle.
- Build in time that allows teams to monitor and measure their progress.
- Respect and listen, but also ask, "Why?"

REGRESSION TESTING

Regression testing is one way of the best ways to tackle these best practices. Regression testing verifies that previously developed software still performs after being changed or interfaced with new software. Ideally, teams should perform regression testing every few Sprint cycles and not towards the end of a project. Teams should discuss during Sprint planning sessions when regression tests should occur in their Sprint process. Teams should find tools that allow them to create automated regression tests versus manually testing every few Sprint cycles. It's much less of a pain to maintain automated regression tests versus assigning team members to run manual tests during Sprint planning sessions.

CHALLENGES OF SPRINT PLANNING AND AND CONTINUOUS DELIVERY

There are also many challenges teams face when conducting Continuous Delivery and Sprint planning together.

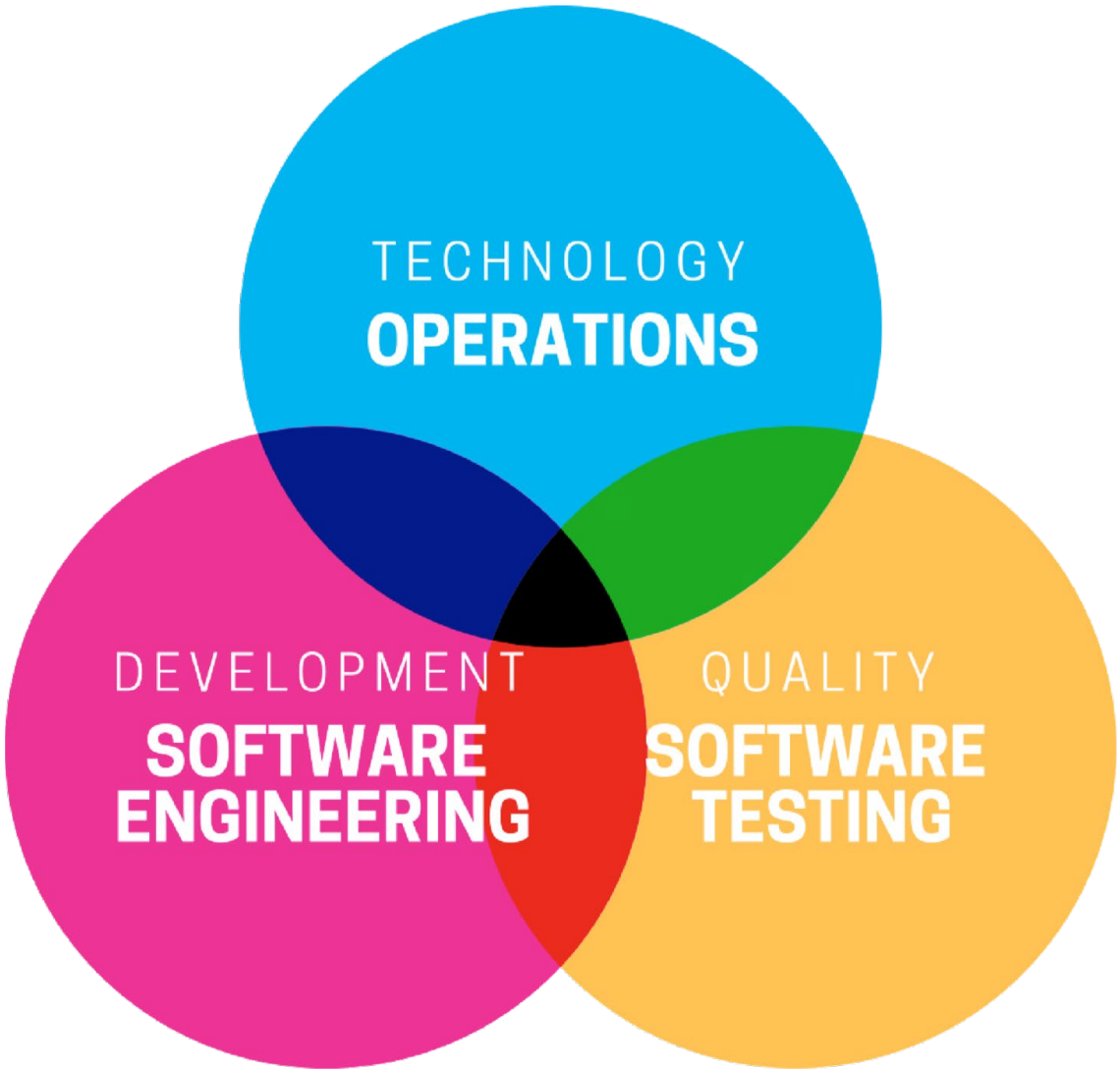
1. **Culture Adoption**
   To implement CD and CI, an organization needs to embrace co-ownership of the development process. Typically, there are three main obstacles for culture adoption: (1) resistance from the status quo, (2) resistance to the process, and (3) strict instead of adaptive implementation. CI/CD adoption should be scoped to the organization's needs; there isn't a one size fits all solution. To overcome this, organizations should ensure that their product and business leaders understand the benefits of CI/CD and how it powers better and faster software delivery. New leaders should be cultivated, not appointed.
2. **Projects With Ambiguous Scopes**
   Highly unknown projects can be difficult to deconstruct and prioritize in the development cycle. The risk profile has changed and the teams may not be aware of that. In these situations, good leadership is required to be able to plan for unknowns.
3. **External Teams**
   When working with an external team, your team's resources can be split and their ability to move quickly can be compromised. They may have to focus on other efforts that aren't in line with maintaining the build or properly planning the next Sprint cycle. It's important to maintain a team that manages the CI and CD pipeline when this occurs, making sure it stays optimal until teams that are interfacing with external resources can return. This requires teams to teach the rest of the culture the importance of maintaining CI builds so that whole teams aren't pulled away.
4. **Complex Technology Stack**
   Development stacks are highly complex, typically with multiple layers of software, processes, servers, and API layers. This can make it challenging to build a CD pipeline and Sprint planning structure that addresses these complexities. Frequent tool changes could also leave team members frustrated and constantly chasing dead ends. This lowers team morale, which can have negative impacts on culture. If you build using a tool that is going to be changed in the future, you could lose a lot of progress, which is another way to quickly lower morale.

CONCLUSION

Sprint planning powers CD and CI processes by allowing teams to accurately plan and prioritize features that fit the business needs. CD and CI also enable development teams to keep pace with their Sprint cadence and create a product that serves the organization as a whole. To ensure adoption, teams must teach stakeholders the benefits of CI/CD. Sprint planning is one of the best ways to teach stakeholders the benefits of CI/CD by showing them how CD/CI processes help the business.

JEFFREY LEE is a technical content writer for Particle where he writes and designs articles that explores the complexities of IoT. In his spare time, he writes original content for DZone and is currently working on his MSc in Technical and Professional Communication at University of Wisconsin-Stout. In the past, he was a freelance writer for developer tools and tech startups. As a technical writer, he loves writing about topics that challenge him and force him to think in new ways.

rainforest



RETHINK QA FOR DEVOPS

The way we build software has changed; the way we do QA hasn't. It's time to rethink QA and bring it into the era of continuous delivery.

Rainforest's model for doing QA right recognizes where test automation affords value and efficiency, leverages machine learning to optimize accuracy and make tests smarter, and introduces human intervention, when critical to the best user experience. Our model introduces QA as an API so it seamlessly integrates with development team workflow. Deploy faster and ship code confidently with Rainforest.

Learn more at www.rainforestqa.com

QA and DevOps: Striking a Balance Between Speed and Quality

Whether you're still considering adopting DevOps or have already started on your journey, you've likely encountered challenges when it comes to your QA process. What should you know about the role of QA in a DevOps organization, and how can you optimize testing for better, faster results?

THE FUTURE OF QA IN A DEVOPS WORLD

A DevOps QA team must shift their focus from finding to preventing issues. QA teams must surface as many issues as possible before code hits production by testing earlier and more frequently than ever. Automating as much as possible, wherever possible, is a central component of implementing DevOps. Building a QA process that automates as much as possible keeps testing from becoming a bottleneck and provides more confidence with every deployment.

SOLVING THE QA PROBLEM: STRIKING A BALANCE WITH A HYBRID APPROACH

Unfortunately, testing automation solutions still fall short for DevOps, as even the most robust automated test suites require time and technical resources to script and maintain test cases.

Our model for doing QA right recognizes where test automation affords value and efficiency, leverages machine learning to optimize accuracy and make tests smarter, and introduces human intervention when critical. By incorporating human talent at the right point — the point where real users bring the greatest value — teams can accelerate innovation without trading off quality or risk.

RETHINK QA AND BRING IT INTO THE ERA OF CONTINUOUS DELIVERY

The way we build software has changed; the way we do QA hasn't. By rethinking the QA process to better align with DevOps goals, teams can get ahead of the curve and start shipping better software more rapidly.

WRITTEN BY RUSS SMITH
CTO AND CO-FOUNDER OF RAINFOREST QA

PARTNER SPOTLIGHT

Rainforest QA

Time to rethink QA and bring it into the era of continuous delivery.

CATEGORY
On-demand QA Solution

RELEASE SCHEDULE
Continuous

OPEN SOURCE?
No

CASE STUDY

Guru is a knowledge management solution that helps teams capture, share and access knowledge easily. Their development team manages their own QA testing. In order to keep their team small as their product, Guru integrated Rainforest QA into its development workflow, encouraging their engineers to write tests from their code editor.

The Rainforest machine learning algorithm confirms all test results, allowing Guru to have confidence in the quality of their test results. By leveraging Rainforest, Guru has scaled their developer-driven quality process rather than hiring a dedicated QA manager. As a result, they have recovered 100+ hours of developer time from testing each month without sacrificing product quality.

Read their story **here**.

STRENGTHS

- Scalability and coverage: test in parallel across multiple browsers on mobile and web apps
- Provide AI-verified results in under 30 minutes to quickly pinpoint bugs and resolve issues before releases
- Integrate fast, actionable QA insights into your development tools and processes
- Reduce time between rollout, feedback and bug remediation
- Relieve testing bottlenecks and accelerate time-to-market

NOTABLE CUSTOMERS

- Adobe
- Bleacher Report
- TrendKite
- Oracle
- StubHub

WEBSITE rainforestqa.com | **TWITTER** @rainforestqa | **BLOG** rainforestqa.com/blog

Issue Resolution Anti-Patterns

BY DAN GOLDBERG
SENIOR MANAGER OF CONTENT MARKETING AT **XMATTERS**

QUICK VIEW

01. When an incident occurs, incident managers are in a race against time to resolve it before it can affect customers.

02. Organizations that practice DevOps limit errors and resolve incidents faster than those that don't.

03. Teams in DevOps environments do a good job of sharing tools, but they struggle to share information from those tools.

04. Sharing data between systems and targeting the appropriate people can help make issue resolution more efficient.

In the battle between software delivery speed and infrastructure stability, speed is winning. Organizations are releasing software faster than ever before to keep up with market demand. But when speed comes at the expense of software quality and incident response, organizations suffer along with their customers and partners.

The benefits of DevOps are well documented, and companies continue to prove them. According to the **2017 Puppet State of DevOps Survey**, top performers deploy 200 times more frequently, have change failure rates that are three times lower, and recover from failure 24 times faster.

When xMatters worked with Atlassian on a **survey of more than 1,000 organizations** about their DevOps environments, more than 60% said they were enjoying the benefits of DevOps that they expected. Lost in the details, however, another 1,000 organizations didn't even make the cut in the survey report because they didn't have a well-defined DevOps plan.

For top performers and luddites alike, there are barriers to ideal development and operations processes, and there are solutions. We're all spoiled for systems that work perfectly every time from anywhere, so near perfection is about the only way to keep our customers happy.

BARRIERS TO PERFECT DEVOPS

Sometimes people outside the DevOps realm think DevOps is a prescriptive set of guidelines like ITIL, but it's really a philosophy of working together and sharing the load to produce faster deployments and releases. How that philosophy manifests itself in the real world is up to each organization based on their use cases, personnel, and appetite for risk.

So, organizations have had nearly a decade to replace their separate development and operations teams with a more collaborative culture, and they have been pretty successful overall. In fact, according to the xMatters-Atlassian survey, more than 80% of organizations share tools between development and operations. The breakdown is in the knowledge sharing.

When teams have to request access to information, or access timeouts, or their access is limited to certain areas, delays build up and trust is reduced.

According to the **xMatters-Atlassian survey**, more than 90% of organizations share knowledge in at least some way. However, three-quarters of those organizations have restrictions on what knowledge is shared.

Part of the problem is that organizations still have to look at other systems to get the information they want. If companies automated the way data moves between systems, they wouldn't have to stop and give explicit consent every time they wanted to share information.

For instance, if an APM tool catches an error in an application running in a dev environment, an IT ops manager might open a Slack or HipChat channel, and an incident manager might open an incident in Jira. If they pull in additional engineers and they have to search both Slack for the conversation thread and Jira for incident details, they waste valuable time. If the Slack channel is available in Jira, everyone can collaborate easily in one place.

PROBLEMS WITH DEVOPS AND INCIDENT MANAGEMENT

As you can see in the Puppet survey, leading DevOps organizations are

limiting errors and resolving incidents faster than their less DevOps-ish counterparts. But the number of errors that reach production after companies release software is still alarmingly high.

Nearly half of organizations say they have to fix errors in production. One in every 15 organizations has major issues with new application releases, forcing rollbacks.

Why is the error rate so high? With so many quality monitoring tools available, organizations are recording virtually everything that happens in their systems, so it's not a data collection issue. In fact, more than 60% say their monitoring solutions predict potential issues before users are affected. Instead it goes back to the knowledge sharing we talked about earlier.

When an incident occurs, incident managers are in a race against time to resolve it before it can affect customers, employees, or even a wider swath of people out there. So, every element that delays discovery and action puts the company more at risk.

Are such elements showing themselves during incident management situations? According to the **xMatters-Atlassian survey**, they certainly are:

> "DevOps has a clear call for more individual autonomy, yet 50% say they have to wait for the operations center to declare a major incident before taking appropriate action.
>
> DevOps relies heavily on automation, but 43% still use manual process to keep customers and internal stakeholders up to date.
>
> DevOps is supposed to empower individuals to have an impact on the organization, yet 34% say waiting for subject matter experts delays incident resolution.
>
> DevOps is supposed to improve communication across teams and systems, but 29% say duplicate tickets are created while the incident is being resolved.
>
> DevOps is supposed to streamline processes, yet 23% say tickets are routed without proper assignments and must often be rerouted."

INCIDENT MANAGEMENT SOLUTIONS

There are a few methodologies and technologies to mitigate these issues, including:

- **Targeted messaging:** Developers love themselves some code, and when they don't have a ready solution, they are apt to build it themselves. When it comes to CI tools like Jenkins or TeamCity, developers on large teams are likely to build their own instances until the cluster gets too confusing.

  So, when a message from Slack or HipChat comes in, exactly who should respond can be contentious. As a best practice, document instances of your CI tools and target messages to the developers who need to receive them.

  An error in your CI processes might just be some discomfort at first, but unaddressed discomfort becomes pain.

- **Culture and process:** Surprisingly, more than half of organizations practicing DevOps do not have documented incident management procedures they can follow and repeat. ITSM organizations live and die by their incident response processes. For some reason, that culture of repeatability has not moved over to DevOps yet.

  Virtually every DevOps organization has monitoring tools in place and processes for testing. When things go wrong, most organizations have the tools in place to implement similar processes during development cycles or for more major incident situations on production.

- **Data and information:** When a product test fails in production, QA or testing teams log it so the errors can be fixed. Advanced APM tools can even uncover the root cause of the errors. But that information can get locked up in siloes, and the engineers who have to do the work have to ask for it or discover it for themselves by poring over code.

  Automated routing can pass along not only the test results but detailed analysis from monitoring tools and MOM systems.

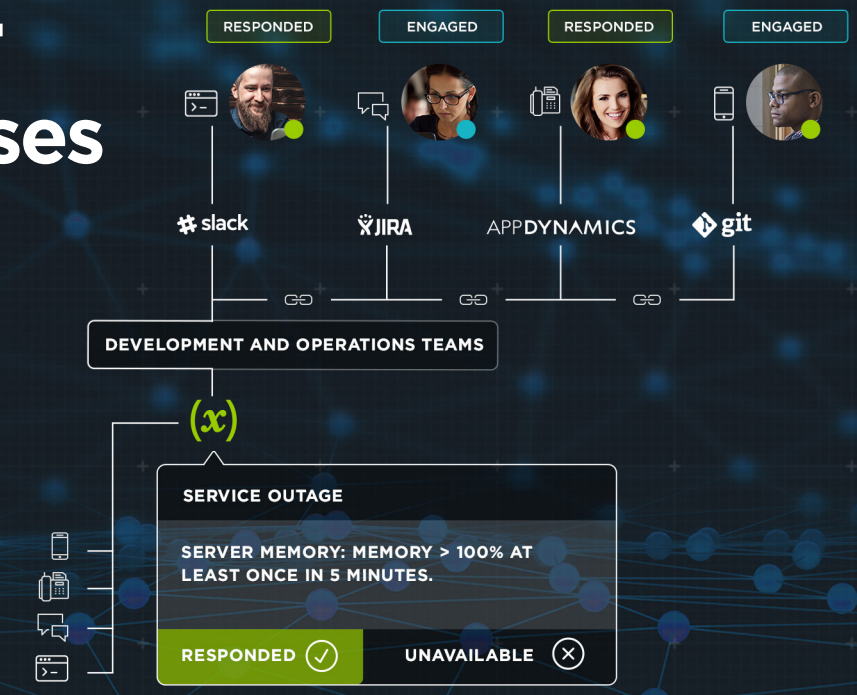
CONCLUSION

DevOps environments can be chaotic enough when everything goes right, especially if you're releasing code multiple times per day. Automation can help replace chaos with repeatable process…until it breaks. And it will break.

And when it does, be prepared to resolve incidents and get your development and software delivery cycles back on track quickly. In today's world of daily (or more) releases, waiting until morning can be disaster.

DAN GOLDBERG is Senior Manager of Content Marketing for xMatters. He has more than 20 years of experience as a writer and editor, beginning in newspapers. He has worked for more than 15 years in the technology industry, where he worked as a front-end web developer before returning to content. Previously he worked at PeopleSoft, Taleo, and CallidusCloud.

Modernize Your DevOps Processes with xMatters



Connect your existing build, deployment, management, monitoring, and collaboration platforms.

RELAY ALERTS BETWEEN SYSTEMS

ENGAGE THE RIGHT PEOPLE TO RESOLVE INCIDENTS

SUPPORT CONTINUOUS DELIVERY PROCESSES

Resolve issues faster with xMatters. xMatters.com/moderndevops

(x) matters®

3 Ways to Take DevOps Processes to the Next Level

Businesses are working at a faster pace to meet customer expectations. Developers have shortened release cycles and ops teams have to support quicker product deployments. Here are 3 keys to keep things humming along:

INCREASE AGILITY AND THROUGHPUT BY CONNECTING YOUR TOOLS

As you adopt DevOps processes the walls between teams naturally break down. The next step is to make sure the walls between your tools are also broken down. You need to share information between tools to engage resources more effectively. With xMatters, you can connect your existing development, deployment, management, monitoring, and collaboration platforms. From HipChat to Slack, ServiceNow to JIRA, Splunk to New Relic, and beyond, you can use xMatters to hand off data between systems while engaging the right people along the way.

DEVELOPERS ON CALL? YIKES. YOU'D BETTER DO NOTIFICATIONS RIGHT.

As you mature your DevOps culture, developers will have to support their own code in production and be on call. So if you don't want to risk waking up a grumpy developer, you'll need to make sure your data is correct so the right person is contacted. With xMatters, you can easily manage users, groups, subscriptions, schedules, and contact information to ensure your alerts and notifications reach the right person. Our powerful integrations allow you to customize response options so developers can start resolving the issue right from the alert, focus on resolution activities, and avoid major incidents.

INCREASE THE VELOCITY OF YOUR CI/CD PROCESSES

Your release pipeline is automated; automate your communications too so you can drive critical processes forward and avoid costly service outages. Instead of copying and pasting (or worse, typing) alert data into tickets or chat rooms, etc. automatically push relevant data from your monitoring alerts to the tools you use to resolve incidents.

WRITTEN BY ABBAS HAIDER ALI
CTO, XMATTERS

PARTNER SPOTLIGHT

xMatters

xMatters delivers integration-driven collaboration that relays data between systems while engaging the right people to proactively resolve issues.

(x) matters®

CATEGORY
DevOps, IT Alerts

RELEASE SCHEDULE
Quarterly

OPEN SOURCE?
No

CASE STUDY

FORTUNE 500 FIRM BUILT A MODERN DEVOPS TOOLCHAIN TO MAXIMIZE VISIBILITY FOR ITS TEAMS

As the leading provider of on-demand, integrated digital solutions designed to enhance the efficiency and profitability for all major segments of the retail industry, this Fortune 500 Firm constantly faces urgency in meeting customer needs faster and with greater stability. Three big challenges they faced was lack of visibility, a small DevOps team, and the requirement to do more faster and with higher quality.

Thanks to a combination of monitoring, management, and communication tools brought together by xMatters integrations, the firm improved their time to resolve issues and product development speed.

With xMatters, this Fortune 500 Firm was able to:

- Reduce time to restore service to business partners
- Reduce business impact of incidents thanks to a streamlined monitoring process
- Automate communications between tools
- Automate processes and information flows to increase visibility

STRENGTHS

- Connect insights from your existing tools to the people, teams, and systems needed to drive business processes forward
- Leverage group on-call schedules and rotations, escalation rules, and user device preferences to automatically engage the right resources
- Customized response options allow teams to quickly get up to speed and take action during incidents
- Robust self-service integration platform with 100s of prebuilt configurations that can be installed in minutes
- 15 years of experience integrating people into toolchains spanning DevOps, Ops, and Service Management solutions

NOTABLE CUSTOMERS

- 3M
- Fiserv
- Fujitsu
- Manpower
- The Kellogg Company
- ViaSat
- Walgreens

WEBSITE xmatters.com/moderndevops | **TWITTER** @xmatters_inc | **BLOG** xmatters.com/blog

are you doing DEVOPS RIGHT?

In this year's audience survey, only 18% of readers said that their organizations had achieved Continuous Delivery across the board, while 28% reported that some of their projects were following Continuous Delivery practices. However, we also know that just because someone says they're following DevOps processes, doesn't mean that they really are. So, before you claim to be following the best practices of Continuous Delivery, let's play a game and see where you land on the DevOps score! For each question you answer "yes" to, give yourself a point, then see where you fall on the score tracker.

THE GAME OF DEVOPS

1 YES = 1 POINT

HAVE YOU ADOPTED A SOURCE CONTROL TOOL LIKE GIT OR SVN?	DO YOUR DEVELOPMENT AND OPERATIONS TEAM COMMUNICATE FREQUENTLY?	ARE YOUR UNIT AND PERFORMANCE TESTS FULLY AUTOMATED?
HAS YOUR TEAM ADOPTED A CONTINUOUS INTEGRATION SERVER?	HAS YOUR TEAM ADOPTED APPLICATION RELEASE AUTOMATION SOFTWARE?	DO YOU HAVE AUTOMATIC CHECKS TO PROCEED THROUGH THE DEPLOYMENT PIPELINE?
ARE YOUR DEV, OPS, AND QA TEAMS ALL USING THE SAME TOOLS?	IS MANAGEMENT FOSTERING A DEVOPS-FRIENDLY CULTURE?	DO YOU PERFORM CODE QUALITY CHECKS SUCH AS CODE REVIEWS?

NOVICE

You're just getting started on your DevOps journey. Maybe you're working with your teammates or management to adopt some of the necessary tooling, or maybe you're in a position where management refuses to support these efforts. Keep trying, you'll get there!

ADOPTER

You're well on your way to becoming a DevOps organization. Maybe you're still working out some cultural kinks, or trying to figure out the best way to automate what you can. Putting in the effort will pay off in the long run.

THOUGHT LEADER

You, or members of your team, must be DevOps pros who can't imagine life any other way. You deploy at least once a day, failures are resolved quickly, and you collaborate on a regular basis with ops and QA teams, right down to using the same tools and pipelines.

You Can't Manage What You Can't Monitor — Why Metrics are Key to Enabling DevOps

The pace of innovation in technology and digital transformation is accelerating as barriers-to-entry for new competitors are collapsing. The shift to cloud and the commoditization of compute and storage infrastructure are driving this shift, with highly scalable and highly available services just a credit card away. Ideas are turned into viable products in days or weeks instead of months or years.

This shift is also being accelerated by the transition to ephemeral infrastructure. Instead of updating and patching systems, it's easier to spawn new instances and deprecate old ones. These on-demand microservices, containers, and functions (including Lambda) allow developers to spend more of their time creating new features and releasing them continuously to production. As customers are constantly exposed to new code, service impacts can be frequent and widespread. The only way to mitigate those impacts is to have real-time visibility into your entire service stack, with the ability to redact and re-release code.

Traditional APM and logging tools don't work in short-lived environments. These heavy-weight agents take minutes to instantiate, which means you're flying blind at the most critical time. Functions such as Lambda last only seconds or minutes, and don't natively provide real-time visibility into their cold-start, processing, and termination.

Real-time metrics are the only way to gain visibility into these short-lived services. The power to see all your services at one second resolution and only a few seconds delay means you can provide your customers with a superior experience that gives your organization competitive advantage.

WRITTEN BY MIKE KLACZYNSKI
PRODUCT MARKETING DIRECTOR, SIGNALFX

PARTNER SPOTLIGHT

SignalFx

signal fx

True real-time metric alerting, automation, and analytics for better operational decisions.

CATEGORY
Advanced Real-time Metrics Monitoring

RELEASE SCHEDULE
Weekly

OPEN SOURCE?
Yes

CASE STUDY

Acquia helps companies build amazing digital customer experiences using Drupal. As their customer base grew, however, the company needed better insight into customer instances and quicker access to operational data it could trust. That's when Acquia turned to SignalFx for monitoring its growing cloud services ecosystem. By providing the same metrics data access to developers, engineers, operations, tech support, and business users, they slashed call times, have shorter time-to-resolution, fewer disruptions, happier customers, and less burden on Acquia's technical team. Employees empowered with data release higher quality code more frequently, and provide a differentiated product for their customer and a competitive advantage for Acquia.

STRENGTHS

- Real-time metric analytics and alerting at 1 second resolution, with 2-3 second latency.
- Consolidate metrics from infrastructure, services and microservices, applications, containers, and functions. Custom instrumented metrics aren't treated differently.
- Unique streaming architecture scales to usage by the largest cloud-native organizations.
- Easy to use for everyone in your organization, for visibility with consistency.
- Pricing based on amount of data ingested, not number of hosts. Optimized for ephemeral microservices, containers, and Lambda.

NOTABLE CUSTOMERS

- Square
- Nasdaq
- Kayak
- Lululemon Athletica
- Spotify
- Yelp

WEBSITE signalfx.com | **TWITTER** @signalfx | **BLOG** signalfx.com/blog

ProdDev: the New DevOps?

BY ANDREW PHILLIPS
VP DEVOPS STRATEGY AT **XEBIALABS**

QUICK VIEW

01. While dev and ops teams are doing better at communicating with each other, work is still often "thrown over the wall" between product management and development.

02. Getting the two groups to interact more frequently is a great way to improve understanding of user needs across the entire team.

03. Embedding user-focused practices helps teams go from building software better to building better software.

It's a theme heard regularly when discussing the software development process: "I wish A and B talked to each other more and each had a deeper understanding of what the other was doing. If only A and B were incentivized to work together, rather than 'throwing work over the wall' from one to the other, things would be so much better."

A = "Dev" and B = "Ops" is probably the most well-known current version of this theme. At the level of individual teams, though, I'd contend that the issue of "throwing work over the wall" from developers to Ops is, if not quite a solved problem, then at least a problem with a known set of solutions.

This should not be taken to imply that developers and Ops now necessarily understand each other's work or mentality much better. Rather, the increasing availability and maturity of cloud IaaS offerings and services provided on top of them, means that detailed knowledge of operational "secret sauce" is now required in far fewer cases. At least for straightforward applications that are not (yet) pushing the envelope, scalability, resilience, automated deployment and management, etc. have increasingly become out-of-the-box experiences.

There is a different phase in the software development lifecycle, however, where "throwing work over the wall," low levels of communication, incomplete mutual understanding, and, not infrequently, misaligned incentives are still common. Product Management and Development teams often communicate only when descriptions of features to be built are thrown over the wall (a.k.a. "handed over") in the form of requirements, documents, or items on the product backlog. Ongoing dialogue to explain and fine-tune the motivation for these features is rare, and is additionally complicated by the lack of a shared, up-to-date model of the needs of users and the position of the market in general. More fundamentally, a clear shared understanding of the roles of Product Management and Development in ideation and feature refinement is often missing, and the respective views of how much value each group adds to the overall process and product differ quite substantially.

WHY WE NEED "PRODDEV"

OK, so for argument's sake, let's say there indeed is still is a "wall" between Product Management and Development. Why is this a problem? If DevOps makes the "how" part of building software that runs reasonably well so much easier, what else do we need?

In the first place, it's worth noting that many companies, and certainly most of the leading tech companies, place a strong emphasis in their core values on user-centric thinking, or "putting the customer first," in their core values. That user or customer, in general, doesn't care much about how easy it was to build the software they are using. They may care about how quickly the software can be updated to meet new needs, and this is an area

in which adopting cloud and DevOps practices definitely helps. More than that, however, they care about *what* is being built – what the software they use actually does and how it makes their lives easier. In most cases, this is far more important to users than the speed of development: a clumsy feature delivered a week early causes many more problems than a great feature delivered a couple of weeks late.

Of course, getting the "what to build" question right is genuinely a Very Hard Problem, as anyone who's had to wrestle with product/market fit will attest. One study of a widely-used piece of software showed that about a third of new features genuinely improved the user experience, a further third had a broadly neutral impact, and the remaining third actually made things worse. Whether these proportions are fully representative or not, it seems evident that the more of the team's intelligence and creativity that can be harnessed to tackle the problem of what to build, the better.

One study of a widely-used piece of software showed that about a third of new features genuinely improved the user experience, a further third had a broadly neutral impact, and the remaining third actually made things worse.

Why then have only one person, or a couple of people, work in relative isolation and show up at the beginning of the project or sprint with a list of features or user stories that the team "just needs to implement?" Why discuss the underlying user needs that provide the critical background to the intended features just before work on them is about to start? Why use synthesized and abstracted "user personas" instead of personal interactions with real end users?

Obviously, researching the market, talking to active and potential users, and defining and validating coherent feature sets that address genuine user needs are usually full-time activities. Expecting a development team that's trying to get "into the zone" for maximum productivity to be engaged in all of these tasks would likely be very disruptive. Luckily, there are a number of relatively low-friction activities that can be incorporated into the team's routine fairly easily, and which can start to bridge the "Prod-Dev Gap:" regular user interactions, early backlog insight, review of user-level metrics, and access to user feedback.

REGULAR USER INTERACTIONS

As long as users are represented by synthesized and somewhat "faceless" user personas, creating a sense of empathy within the development team for their applications' users is difficult. Developers, as expert users, are also often at risk of not understanding how or why users fail to "get it" when features turn out to be much less usable or useful than intended.

Arranging in-person contact between developers and users can be difficult, especially if the user base and development teams are geographically separated. Presenting profiles of individual users to the dev team on a regular basis; ensuring the team gets to interact with users during meetings such as Customer Advisory Boards; encouraging attendance at conferences, meetups, and similar gatherings; or even just reporting back to the dev team about meetings with existing or potential users: these are all steps that can hopefully be taken without too much difficulty to increase the team's feeling of being "close to the user."

EARLY BACKLOG INSIGHT

In many organizations, "grooming the backlog," i.e., ensuring feature requirements are appropriately prioritized and are described in sufficient detail to be worked on by developers, is the "upstream task" of Product Management alone. As a result, much of the information about the market, identified user needs, and competitive positioning that goes into determining why a particular feature is needed and why it is considered to be more important than other features, remains largely hidden from the dev team.

Short, regular sessions where one or more developers review the pipeline of future product work can greatly increase the development team's level of insight into the ideation and feature refinement process. For example, a group working in two-week sprints could schedule a bi-weekly session in which a rotating member of dev team meets with the product manager to review what's likely to be coming up two to three sprints down the line.

In addition to providing early input into any technical challenges ahead, such meetings can help ensure that developers have a better picture of which user needs are being investigated, and also gives them the opportunity to suggest possible alternative solutions.

REVIEW OF USER-LEVEL METRICS

It's common for standups and other status meetings to review burndown charts and similar development status metrics, but taking a quick look at how the application is doing from a user perspective is something that usually happens much less frequently, if at all. Often, a necessary prerequisite involves defining and collecting metrics that aactually relate directly to the user experience: for example, checkout processes abandoned, user-experienced load time, or time taken to complete a particular user flow.

Part of the challenge here is that — unlike low-level operational metrics such as CPU usage or API error rates — user-level metrics are, in most cases, not provided out-of-the-box by cloud platforms or monitoring tools. Instead, measuring and publishing these metrics has to be explicitly coded into the application, while the appropriate scripts and dashboards to analyze and visualize the results also have to be created by hand.

Still, dedicating a particular timeboxed amount per iteration to improve user-level metric gathering and reporting for your applications is a great first habit to adopt. This is as much due to the increased focus on user-level experience as it is the result of any particular metric captured or dashboard created.

ACCESS TO USER FEEDBACK

If there is a recognized support channel for the team's applications, exposing developers to support interactions can be a great way of putting them in touch with users' day-to-day questions and experiences. At the same time, inundating the development team with support requests to the point that they're no longer able to work effectively needs to be avoided.

Practices that are worth exploring include regular review sessions of recurring or noteworthy support issues conducted by rotating members of the support and development teams, and "Stack Overflow" or "forum duty," where a rotating member of the development team spends a fixed amount of time responding to user questions.

A chat channel or status board that displays ratings, feedback comments, or similar indicators of user happiness to the development team can also be very useful. Care needs to be taken, however, that such an information channel is not so "in your face" as to become an ongoing distraction — such as a channel that alerts all members with a visual notification for every message.

ON MISALIGNED INCENTIVES

"ProdDev practices" like those described often sound sensible in theory, but from experience we know that much more is needed to actually make them work in real life. One key lesson from numerous attempts to introduce DevOps is that little is likely to happen if the practices conflict with fundamental differences in how the respective groups are motivated.

One way this misalignment can manifest itself between Product Management and Development is if developers are encouraged (through internal attitudes, promotion culture, etc.) to aim for technically complex and "sophisticated" solutions, rather than aiming for The Thing That's Best For the End User. Since this question often goes all the way up the food chain and ultimately revolves around (re-)classifying IT as a business rather than a technical function, it's unfortunately generally not one for which there are straightforward solutions.

It's worth noting, though, that a focus on technical sophistication and complexity is rarely aligned with most organizations' professed focus on the user or customer. Therefore, the question "how does this ensure we build the best solution *for the user?*" can hopefully at least become a talking point when team goals are under discussion. Also, the more the team is able to adopt some of the ProdDev practices presented, and the more regularly they are exposed to users and user-related metrics as a result, the more the team will habitually define and measure success in terms of user satisfaction rather than via technical achievements.

CONCLUSIONS

With DevOps, we have developed a whole bunch of techniques to help enable teams to *build software better*. With the creative capacity that this has freed up, we can hopefully now focus on leveraging the skills, ingenuity, and intelligence of the entire team — not just of an "upstream provider" in Product Management — to tackle a far more challenging problem: how to *build better software*, software that our users really want and need.

Let's hear it for ProdDev!

ANDREW PHILLIPS heads up strategy at XebiaLabs, developing software for visibility, automation, and control of Continuous Delivery and DevOps in the enterprise. He is an evangelist and thought leader in the DevOps and Continuous Delivery space. When not "developing in PowerPoint," Andrew contributes to a number of open-source projects, including the multi-cloud toolkit Apache jclouds.

ARE WE THERE YET?

Ten Milestones on the Way to a Successful DevOps Transition

Adopting a DevOps culture and M.O. in any business, large or small, doesn't happen overnight. But making sure you hit key milestones along the way lays the groundwork for a highly successful evolution to a more agile, less siloed world. If you are contemplating — or in the middle of — a DevOps movement, how many of these milestones have you hit? And which are you still striving for?

1 FIND YOUR CHAMPION

Getting management on your side early in the game is crucial. Toward that end, identify a champion from your executive leadership team to consistently promote your initiative to the C-suite. This leads to greater levels of sustained success and adoption.

2 INITIATE A PILOT PROJECT

A pilot project allows you to test-drive DevOps on a smaller scale so you can quickly demonstrate the potential of a full-scale evolution.

3 DEFINE THE PARAMETERS OF SUCCESS

Clarify the criteria of success, and make sure your team — top to bottom — knows them. This gives everyone something to aim for and removes the guesswork from evaluating the project.

4 EMPOWER YOUR TEAM

Give your team the autonomy to define and fill the roles they need to build a successful DevOps group.

5 TRAIN TO GAIN

Train your team on the new skills they need to perform their new DevOps duties. Consider hiring a coach with DevOps experience who can help guide the team during their initial pilot.

6 GIVE THE TEAM BREATHING ROOM

If possible, clear space for your team during the pilot project. Assigning work outside of the pilot will dilute their focus and possibly lower morale. You've embarked on the DevOps journey to deliver the results your traditional methods could not deliver. Trust the process so the team can do their work.

7 PROVIDE ADDITIONAL FUNDING

Your team may need additional servers or tools to execute Continuous Integration/Continuous Delivery. This needs to be allocated at the beginning of the project so that funding does not become a bottleneck.

8 EMBRACE AUTOMATION

Encourage your team to automate processes like the provisioning of servers, testing, and code deployment. The more they automate processes, the faster your releases will flow — with fewer issues.

9 UPDATE TEAM MEMBER GOALS

The odds are high that existing goals for individual team members will no longer make sense in their new DevOps roles. Update these goals. For example, an ops person should not be measured only by the number of incidents closed, but on the success criteria defined at the beginning of the project.

10 CELEBRATE EVERY WIN — ESPECIALLY THE SMALL ONES

Your first DevOps team will feel like true pioneers blazing a trail through the wilderness. Keep morale and motivation high by celebrating early and often — especially the small wins.

WRITTEN BY ALLAN LEINWAND, CTO, SERVICENOW

Taking DevOps Monitoring to the Next Level: The 5 Step Guide to Monitoring Nirvana

Download the e-book

We are constantly looking to provide a near-100% uptime... When we looked at our monitoring landscape we realized we needed to refactor our approach in order to meet the success criteria & deliver on ensuring customer success.

Sanket Naik - VP Cloud & Security at Coupa

influxdata®

The Modern Engine for Metrics and Events

Monitoring Maturity Model

Emerging trends such as microservices, containerization, elastic storage, software defined networking, and hybrid clouds all keep pushing the boundaries of what constitutes DevOps Monitoring. Your Monitoring architecture should:

- Improve observability for all by facilitating the collection of metrics and events
- Deliver a real-time pipeline
- Make it simple to capture events
- Provide long-term retention of metrics & powerful search and visualization
- Provide a unified system for monitoring metrics and events
- Deliver high availability, scalability and performance

WHY INFLUXDATA

- **Addresses new workload requirements**: InfluxData can handle a high volume of real-time writes, is purpose-built from the ground up for irregular (events) and regular (metrics) time series data, and offers retention policies for performance and availability.
- **Delivers time-based queries**: Specifically, functions such as aggregation and summation using time-based functions directly in an SQL-like language, and allows for large-range scans of many records very quickly.
- **Provides scalability and availability**: Is a distributed-by-design, multi-instance deployment to allow you to have no single point of failure.

We all aim at that monitoring nirvana that often seems unattainable, but with the relevant conversations and the right tools, it becomes a strategic action plan you can attain.

WRITTEN BY MARK HERRING
CMO AT INFLUXDATA

PARTNER SPOTLIGHT

Cloud Foundry

Cloud Foundry gives you the right tool for the right job with two complementary open source technologies for app developers and operators.

CLOUD FOUNDRY

CATEGORY
Platform-as-a-Service (PaaS)

RELEASE SCHEDULE
Continuous

OPEN SOURCE?
Yes

CASE STUDY

Insurance giant Liberty Mutual knew it needed to make a radical change. CIO Mojgan Lefebvre explained, “We knew we had to become a software company that sells insurance to survive in today’s competitive world.” After adopting Cloud Foundry, the team went from hypotheticals to standing up a minimum viable product (MVP) in just 28 days. Embracing agile methodologies and taking a cloud-native approach by deploying Cloud Foundry, the team created a fully functional portal that was ready within six months and provided:

- 40% strike rate against 20% for industry on-average
- Referral time of only three minutes – more than 3x less than competitor products
- 200 quotes and 60 policies within one month of operation

Cloud Foundry’s flexibility, agility, and scalability enabled Liberty Mutual to move closer to its commitment to digital transformation.

STRENGTHS

- Polyglot and multi-cloud
- Run apps at scale
- Simplify the development lifecycle and container management

NOTABLE USERS

- Allstate
- American Airlines
- Cloud.gov
- Ford
- The Home Depot

WEBSITE cloudfoundry.org | **TWITTER** @cloudfoundry | **BLOG** cloudfoundry.org/blog

User Personas and Pipeline Façades for Effective Release Decisions

BY MANUEL PAIS AND MATTHEW SKELTON
SKELTON THATCHER CONSULTING

QUICK VIEW

01. Delivery pipelines are treasure troves of data for release decisions. But such data needs to be collected, filtered and presented to the right stakeholders.

02. How data is presented is as important as the data itself, caring for the user experience is crucial.

03. User personas are a simple yet effective way to empathize with internal, non-technical users and guide our design decisions.

04. Pipeline façades are analogous to façade design pattern in code, whereby a simplified, filtered view on the relevant data abstracts away tooling complexities.

Continuous Deployment (putting changes live in production as soon as they pass a fully automated delivery pipeline) is a **hard sell in most organizations**. That's partially because release decisions often need to involve different areas of the business (marketing and compliance, to name but a few), even if all the technical bits have been automated.

Continuous Delivery, on the other hand, is really about making timely, well-informed release decisions, unhindered by technical constraints (as the delivery pipeline gives us confidence that all the required technical activities have been performed successfully):

> "Continuous Delivery is the ability to get changes of all types, into production, or into the hands of users, safely and quickly in a sustainable way" — **Jez Humble**

Our interpretation of "sustainable" in this context, and from years of consulting experience, is that we need to put the *right information*, in front of the *right people*, at the *right time*, and in the *right format* for them to consume.

DELIVERY PIPELINES ARE TREASURE TROVES OF DATA FOR RELEASE DECISION-MAKING

Typical usages of pipeline information include activity execution time (in a rather static fashion: "How long is our build? How long are our acceptance tests?") and artifact traceability ("Which code changes were deployed to production between version X and Y?"). Sometimes, information is collected for external auditors ("Who approved this change and when?").

More advanced use cases include historical trends on execution time ("our acceptance tests take 20% longer to run than 6 months ago") or overall cycle time ("on average a user story takes 2.5 days since initial commit until it's in production").

However, there's a lot more data available in our pipeline tooling, especially if we're using a toolchain made up of **single-purpose, API-driven tools**. In fact, a lot of data is logged but not even displayed via the tool's UI, or not with an associated semantic.

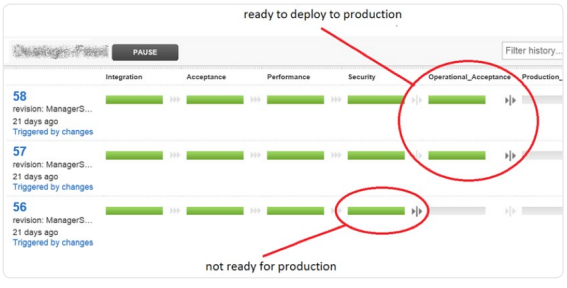
MAKING USE OF PERSONAS TO UNDERSTAND WHAT DATA OUR BUSINESS NEEDS

User personas are a representation of a class of users of our system, personified as an individual with specific needs, skills, goals and frustrations. This concept originated in the UX community and is widely used in design processes.

Thinking of users in terms of personas helps build empathy and design systems that are a better fit for the jobs they need to get done and that avoid known obstacles.

Turns out this is an effective way to understand internal users as well, not only customers. So, we can apply them to understand the information needs of our business stakeholders when it comes to making release decisions.

Let's imagine Amy is a product owner who is waiting for an exciting new feature to be ready to deploy to production. She found out today from the developers in the standup meeting that the changes are ready. After confirming that the related work item has been updated in JIRA, she now finds this scenario in their pipeline tool:



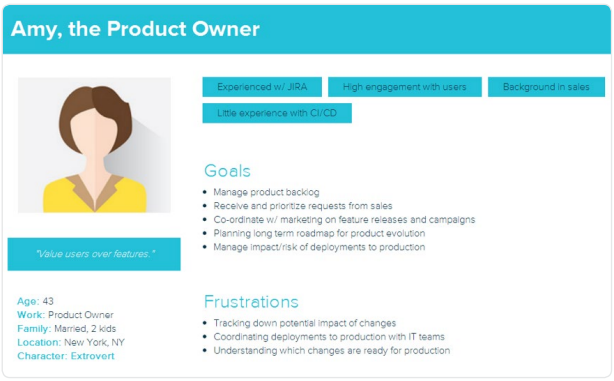
She knows pipeline #56 is not ready for production because it did not reach "Operational Acceptance" stage. But what about pipelines #57 and #58?

Amy reaches out to developers, testers, and ops folks to understand what are the differences and impact of deploying the changes in pipeline #57 versus those in pipeline #58.

By the end of the day, she learns that pipeline #57 is safer to deploy to production, as pipeline #58 includes some backwards compatible database changes needed for a different feature. Tests are passing, but it's peak season for the business and we can't afford downtime or issues during DB migration.

Amy needs to coordinate with marketing for them to launch a social media campaign as soon as the feature is available to customers, so deployment will have to wait for tomorrow.

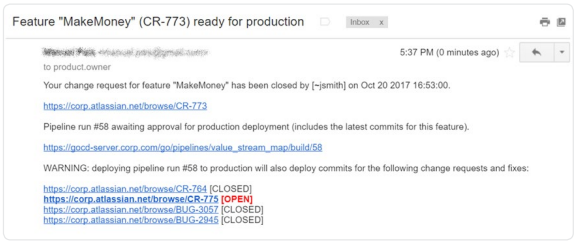
The interesting bit here is that had we mapped out the user persona for Amy (as a representative of the product owner stakeholder), we would have found out she's not familiar with our pipeline tooling, that she often needs to synchronize with marketing and other teams on the timing for a release, and that her frustration level increases when it takes her the better part of a working day to find out the impact of deploying one set of changes versus another.



Amy's user persona highlights common traits, goals and frustrations of product owners in the organization

HOW DATA IS PRESENTED IS AS IMPORTANT AS THE DATA ITSELF

Now, let's imagine Amy gets an email notification like this every time a feature request is closed.



Pipeline façade in the form of a notification email collecting data from JIRA and GoCD

At a glance, without any coordination effort, Amy is informed that pipeline #58 contains the completed feature, but changes for another request have creeped in. She quickly accesses the item under work (CR-775) and asks John, the assigned developer, about the status. After double checking with the DBA team, Amy syncs with marketing and they agree on deploying the previous version instead (pipeline #57).

Because we provided Amy with the data she needed in a format that is familiar and easy to consume, we drastically reduced the time to make a release decision. There is still some coordination and analysis involved, but all the running around and frustration is greatly reduced.

This is just an example. The point is not about this specific use case or implementation, but rather to be aware that **non-technical stakeholders are part of our delivery chain too**. By identifying internal personas, we can better understand how to present them with the information they need in a way that reduces friction and time to decide.

DESIGN PIPELINE FAÇADES TO PUSH THE RIGHT INFORMATION IN THE RIGHT FORMAT

A façade is a software design pattern whereby a single, minimal interface abstracts away the complexity of a larger body of code. The actual implementation might need to make multiple calls to other parts of the code and even merge or transform data between formats in order to provide its consumers what they need and nothing more.

We can do exactly the same with our delivery pipeline: design façades that collect information from the different tools involved and transform it as necessary to provide non-technical stakeholders an actionable, filtered data view. The pipeline and toolchain technicalities are no longer obstacles, making our delivery more inclusive.

Because we are not restricted to a specific programming language, implementing these pipeline façades can take multiple forms: a simple query, an email notification, a webpage, a database view, etc.

The hard part is not implementing, but understanding which implementation better fits our personas needs, skills and frustrations.

Let's consider another example, one which will increasingly be on the spotlight for any organization doing business in the European Union, as the new **General Data Protection Regulation** (GDPR) comes into effect in May 2018. The GDPR enforces stricter data governance and consent management for user data. With fines up to 20 million euros, organizations will be forced to shift their data security and compliance processes left. Again, pipeline information can massively help in decision making for compliance with GDPR.

Assuming our database changes are stored in version control (either using database migrations or database state versioning approach) and go through a pipeline just like any other code changes, quickly identifying relevant changes for data governance can prove massively helpful for Mark, our compliance officer. But how?

Let's imagine Mark has a technical background and is familiar with Oracle databases and SQL scripts. We're not database experts but it's not a

wild guess if we say he'll be quite interested to know when new tables, columns, or views get created, at the very least.

Because our system is based on the Ruby on Rails framework, we know all our database migration scripts are located under *db/migrate* folder in source control, thus we can simply provide Mark a **GitHub query URL** with all commits in that folder containing the *create_table* instruction, right?

Except that those db migration scripts are written in Ruby, and Mark is only familiar with SQL. So instead of looking for commits, we might need to resort to diffing the **SQL database state file that Rails generates** after the migration scripts are applied. The good news is that all of this is available in our pipeline runs.

```
693  693  /*!40101 SET character_set_client = @saved_cs_client */;
     694  +DROP TABLE IF EXISTS `listing_working_time_slots`;
     695  +/*!40101 SET @saved_cs_client     = @@character_set_client */;
     696  +/*!40101 SET character_set_client = utf8 */;
     697  +CREATE TABLE `listing_working_time_slots` (
     698  +  `id` bigint(20) NOT NULL AUTO_INCREMENT,
     699  +  `listing_id` int(11) DEFAULT NULL,
     700  +  `week_day` int(11) DEFAULT NULL,
     701  +  `from` varchar(255) DEFAULT NULL,
     702  +  `till` varchar(255) DEFAULT NULL,
     703  +  `created_at` datetime NOT NULL,
     704  +  `updated_at` datetime NOT NULL,
     705  +  PRIMARY KEY (`id`),
     706  +  KEY `index_listing_working_time_slots_on_listing_id` (`listing_id`)
     707  +) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Also, Mark doesn't really care about *all* the databases our system handles, only those related to users. So, we need to filter out uninteresting results to make this usable. Mark would also like to see at a glance how and when these database changes have been tested and **which data set was used to test them**. All of these should be available in our pipeline, and thus possible to retrieve and present as well.

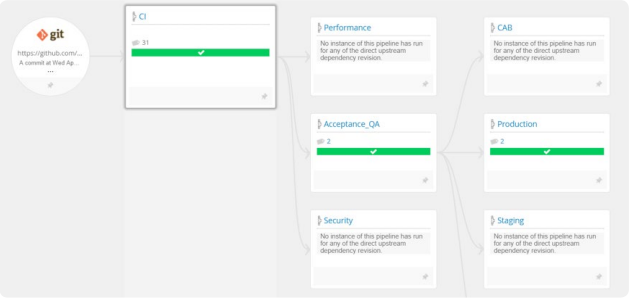
Now imagine instead that Mark had a pure legal background and was not technically savvy at all. We would need an extra data transformation step to translate the filtered-out SQL or Ruby into plain English statements. Possibly we could agree to detail how data will be stored (format, retention policy, etc.) in our commit comments and push them into the release information that Mark will receive.

Hopefully we have highlighted the point that how we make use and present the data in our pipelines for less technical stakeholders is the hard(er) part of making faster, more informed release decisions.

SHORT AND WIDE PIPELINES GO HAND IN HAND WITH EFFICIENT DECISION MAKING

Once you start cutting down the time it takes to make your release decisions, eventually the pipeline itself becomes the bottleneck, especially when all activities are serialized (each activity can only start when the previous has been completed, and all activities in pipeline must be performed for all changes).

This is the time to think about making your pipelines short and wide instead, a pattern **mentioned by Jez Humble**. This means moving your pipeline from a sequential, long path to production, to one where we recurrently make decisions in terms of "for this set of changes, which activities in the pipeline must be performed before release?".



Example of a short and wide pipeline, with three mandatory activities for all changes, and four optional

With a short and wide pipeline, all changes go through a minimum set of activities (for example CI, acceptance tests and deployment to production) but all other activities in the pipeline are optional (for example performance tests or in-depth security tests).

Of course, the pipeline façade pattern can help us again to decide which activities we should perform before deploying to production, based on past deployments and success rate for similar type of changes.

GETTING STARTED

First, identify what kind of decisions need to be made and by who for any non-trivial change to get released to production. If your pipeline already closely maps your value stream then this should be quite straightforward. Pick one, ideally the one causing the greatest delay in time from commit to production.

Second, write up a persona that approximates the characteristics of the group of persons in your organization (or division) fulfilling that decision-making role.

Third, talk to the people in that group and discuss what kind of information would help them make those decisions faster. Ask them to think beyond what they know today (current pipeline tool UI for example).

Fourth, develop a first pipeline façade iteratively. Sketch out (**paper prototyping** is enough) a few different implementation options and assess usefulness (*right data*) and usability (*right format*). Iterate a couple of times at most until you have an MVP. Don't polish this too much, start using it rather quickly to actually validate if it helps your decision-making process.

Finally, once you've proven this approach, reach out to other delivery decision makers and engage with them to design pipeline façades that help them do their job more efficiently.

MANUEL PAIS is an independent DevOps and Delivery Consultant, focused on teams and flow. As member of DevOps pioneers Skelton Thatcher Consulting, Manuel has helped large organizations in finance, legal, and manufacturing adopt test automation and CD, as well as understand DevOps from both technical and human perspectives. Co-curator of DevOpsTopologies.com and co-author of the upcoming book "Team Guide to Software Releasability."

MATTHEW SKELTON has been building, deploying, and operating commercial software systems since 1998. Co-founder and Principal Consultant at Skelton Thatcher Consulting, he specializes in helping organizations to adopt and sustain good practices for building and operating software systems: Continuous Delivery, DevOps, aspects of ITIL, and software operability.

DIVING DEEPER

INTO DEVOPS

#DEVOPS TWITTER ACCOUNTS

 @MartinFowler

 @RealGeneKim

@JezHumble

 @BridgetKromhout

 @ChrisShort

 @daedtech

 @MattWatson81

 @Anders_Wallgren

 @MattStratton

 @PatrickDeBois

DEVOPS ZONES

DevOps Zone

dzone.com/DevOps

DevOps is a cultural movement supported by exciting new tools that is aimed at encouraging close cooperation within cross-disciplinary teams of developers, and IT operations, and system admins. The DevOps Zone is your hot spot for news and resources about Continuous Delivery, Puppet, Chef, Jenkins, and more.

Cloud Zone

dzone.com/Cloud

The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. The Cloud Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

Agile Zone

dzone.com/Agile

In the software development world, Agile methodology has overthrown older styles of workflow in almost every sector. Although there are a wide variety of interpretations and techniques, the core principles of the Agile Manifesto can help any organization in any industry improve their productivity and success.

DEVOPS REFCARDZ

Continuous Integration

Reap the benefits of quality, better testing, and early error detection with proper CI implementation. Using examples from the Git Command Line Interface and Git Plugin hooks for Jenkins, it also walks through build management, build configuration, testing, and code quality.

Continuous Testing 101

There are many misconceptions as to what Continuous Testing is. Let's push the misconceptions aside and learn exactly what Continuous Testing is, the 10 key elements of Continuous Testing methodology, and the benefits of utilizing this concept.

Preparing for Continuous Delivery

In this Refcard you will learn the fundamentals of continuous delivery, including how to set up your delivery pipeline, putting you one step closer to decreased cycle times and faster delivery.

DEVOPS PODCASTS

Arrested DevOps

A podcast that helps you achieve understanding, develop good practices, and operate your team and organization for maximum DevOps awesomeness.

Testing Podcast

A daily podcast that covers test automation, A/B testing, testing in DevOps, and more.

Developer Things

A new podcast that aims to produce high-quality weekly podcasts covering software development best practices, site reliability, developer tools, and more.

DEVOPS RESOURCES

The DevOps Handbook:

How to Create Word-Class Agility, Reliability, and Security in Technology Organizations

Practicing Continuous Integration and Delivery on AWS:

Accelerating Software Delivery With DevOps

Leading the Transformation:

Applying Agile and DevOps Principles at Scale

THE HUB OF ENTERPRISE JENKINS AND DEVOPS

Software at the speed of ideas.

Transform the software that fuels your business with CloudBees Jenkins Solutions.

Your business has amazing, world-changing ideas - the only thing standing in your way is the time, energy and processes it takes to turn code into finished product, to transform ideas into impact. CloudBees® - the only secure, scalable and supported Jenkins®-based DevOps platform - lets you focus on the ideas you want to bring to life, not the challenge of building, testing and deploying them, so you can make an impact sooner.

Get enterprise DevOps automation from CloudBees and allow your team to focus on building great software!

DevOps at scale:
Your business is critical. CloudBees takes the frustration of having to wait for build systems, making them ready and available.

End to End Visibility:
Get better at predicting what features will ship by the drop date. Capture metrics across the pipeline to identify bottlenecks.

Build and Deploy in Minutes:
Jumpstart your DevOps efforts by sharing templates across teams and deploy effortlessly.

Enterprise-grade security:
CloudBees provides a vetted version of Jenkins so you can lock down sensitive projects and ensure compliance.

To learn more about CloudBees Jenkins Solutions, visit: **www.cloudbees.com.**

CloudBees, Inc.
2001 Gateway Place, Suite 670W | San Jose, CA 95110 | United States
www.cloudbees.com | info@cloudbees.com

0118v01

Six Steps to CD Success

You've made the decision to adopt Continuous Delivery (CD). Now how do you get started? CD doesn't just happen. You need a plan for CD success:

1. START SMALL

A common mistake is to try to do too much, too soon. Identify a small project that lets a team try out CD, get used to new processes and experience success.

2. SYSTEMIZE A PROCESS

You can buy a set of tools to help you get started. But until you map out a process, understand the steps and assign roles, you can't get rolling.

3. CREATE A NO-BLAME CULTURE

Successful DevOps teams accept failure to promote learning and risk-taking. Moving to CD is hard and your team needs to work together to achieve shared goals.

4. METRICS FOR SUCCESS

You can't improve what you don't measure. You need to work with the business to determine how you define success. Then, set baselines and track metrics early and often so issues are raised and addressed.

CONFIGURATION AS CODE

A key aspect of CD is to adopt configuration as code. This DevOps practice eliminates problems that result from disparate environmental or tool configurations and improves quality and decreases the mean time to resolution (MTTR) for issues.

ORCHESTRATE AND AUTOMATE

You've defined the process. Now orchestrate it, in order to:

- Ensure reproducible builds
- Share build artifacts
- Deconstruct steps to identify bottlenecks in each phase of the pipeline
- Ensure the pipeline process is visible and transparent to stakeholders

CONCLUSION

CD can seem daunting, but it's a journey worth taking. With CD, you will generate tangible benefits for your company and improve the experience for your customers.

WRITTEN BY SACHA LABOUREY
CEO,CLOUDBEES

PARTNER SPOTLIGHT

CloudBees DevOptics

Provides live insights into the end-to-end application delivery stream by aggregating data from software pipelines and jobs to create a holistic view of software delivery.

CATEGORY
CD Optimization

RELEASE SCHEDULE
Monthly

OPEN SOURCE?
No

CASE STUDY
Capital One Invests in Continuous Delivery to Automate Software Development Pipelines

SUMMARY

Capital One brings highly-customized financial products to market faster with the power of CI/CD and CloudBees.

CHALLENGE

Accelerate delivery of business applications while maintaining the highest quality and security standards

SOLUTION

Use CloudBees Jenkins Platform to provide a stable, scalable CI infrastructure, automate repeatable build processes, and manage CD pipelines from commit to deployment

RESULTS

- 90% of pipeline automated
- Deployment frequency increased 1,300%
- Engineers focused on application development, not infrastructure
- Quality and security ensured through repeatable processes

"We see numerous advantages to CI and CD with CloudBees Jenkins Platform, including shorter time-to-market, improved quality through repeatable processes and a reduction in the cognitive load on our developer community, who are now focused more on the software they are creating and less on the pipeline that creates it." —
- **BROCK BEATTY**, DIRECTOR OF SOFTWARE ENGINEERING, CAPITAL ONE

STRENGTHS

- Identifies improvements in SDLC
- Single source of truth on delivery status
- Quantifies productivity

NOTABLE CUSTOMERS

- ABN AMRO
- Accenture
- Allianz
- Capital One
- WatchGuard

WEBSITE cloudbees.com | **TWITTER** @CloudBees | **BLOG** cloudbees.com/blog

An Introduction to DevOps Principles

QUICK VIEW

01. An explanation of the three ways that underpin DevOps.

02. DevOps is the result of applying the best-practices from the most trusted principles of the physical manufacturing world to IT (Lean, ToC, and TPS).

03. DevOps is all about people, tools are only a mere consequece.

BY MICHELE FERRACIN
CTO AT **MADLAB SRL**

DevOps is a set of practices (not only tools) that aim to apply the lean concepts of the manufacturing world to the software world. At the heart of DevOps are the **Three Ways**:

- *The principles of Flow*, which accelerate the delivery of work from Development, to Operations, to our customers;
- *The principles of Feedback*, which enable us to create ever safer systems of work;
- *The principles of Continual Learning and experimentation*, which foster a high-trust culture and a scientific approach to organizational improvement as part of our daily work.

If we follow these Ways, change will happen. I'm not saying that's easy. We're going to face every kind of difficulty: from the technical to the psychological. Our team has to strive every day to make this change happen. What you have to do is to commit to the Three Ways with small changes that provide quick feedback (think about days or weeks, not months) over and over again. This process will never stop, your team will be always improving in some aspects of their daily work.

A BRIEF HISTORY

DevOps and its resulting technical, architectural, and cultural practices represent the sum of many management methodologies. DevOps resulted from a number of movements and this shows an amazing progression of thinking and unlikely connections. There are decades of hard learned-lessons from manufacturing, high-reliability organizations, high-trust management models, and others that have contributed to the DevOps state of the art today.

DevOps is the result of applying the best-practices from the most trusted principles of the physical manufacturing world to the IT value stream. DevOps has its foundation on Lean, the Theory Of Constraints, the Toyota Production System, and many others. Other valuable contexts that DevOps draws from include management culture and organizational change management. The result is world-class reliability, stability, and security at lower cost and effort; and accelerated flow and reliability throughout the tech value stream, including Product Management.

THE LEAN MOVEMENT

Lean principles focus on how to create value for the customer through systems by creating flow and pull processes (versus push), moving quality closer to the source, leading with humility, and respecting every individual.

THE AGILE MOVEMENT

Agile software development describes a set of values and principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams.

The term agile became famous thanks to the manifesto for agile software development that was created in 2001 by seventeen of the leading thinkers in software development. They wanted to create a lightweight set of rules and principles against other software development processes like waterfall development. One of the main principle is "delivering working software frequently, from a couple of moths to a couple of weeks."

THE CONTINUOUS DELIVERY MOVEMENT

Continuous Delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. Built upon the discipline of continuous build, testing, and integration, the concept of Continuous Delivery defines the role of a deployment pipeline to ensure that code and infrastructure are always in deployable state, and that all code checked in to master can be safely deployed into production.

THE MANUFACTURING VALUE STREAM

One of the fundamental concepts in Lean is the value stream. To better

understand, we'll use some examples for the manufacturing world and then we'll extrapolate the concepts for our tech ecosystem.
The value stream can be defined as the sequence of activities an organization undertakes to deliver upon a customer request, including the flow of information and material.

In manufacturing operations, the value stream is often easy to see and observe: it starts when a customer order is received and the raw materials are released onto the plant floor. In any value stream, there is usually a constant focus on creating a smooth flow of work (the first way of DevOps), using techniques such as small batch sizes, reducing work in process, and reducing rework.

THE TECHNOLOGY VALUE STREAM

The same principles apply to the tech world. With DevOps, we typically define our technology value stream as the process required to convert a feature or change request into a technology-enabled service that delivers value to the end-user.

The input of our process is the formulation of a business concept or a customer request and starts when we accept the work in Development, writing it to our backlog.

Then, Development teams that follow a typical Agile or iterative process will transform that idea into some sort of specification, which is then implemented in source code. The code is then checked in to the version control where each change is integrated and tested (with automation in the process) with the rest of the software system.

Because value is created only when our services are live in production, we must ensure that we are not only delivering fast flow, but also that our deployment can also be performed without causing service outages or security or compliance failures.

KEY CONCEPTS AND METRICS OF DEVOPS

FOCUS ON DEPLOYMENT LEAD TIME

The deployment lead time is a subset of the whole value stream. It starts when any engineer in our value stream pushes the change in to version control and ends when that change is successfully running in production, providing value to the customer and generating useful feedback and telemetry.

Instead of large batches of work being processed sequentially through design and development and then through the test and operations value stream, our goal is to have testing and operations happening simultaneously with design and development, enabling fast flow and high quality. This method succeeds when we work in small batches and build quality into every part of our process.

LEAD TIME AND PROCESS TIME

In the Lean philosophy, lead time is one of two measures commonly used to measure performance, with the other being processing time. The lead time clock starts when the request is made and ends when it is fulfilled, the process time clock starts only when we begin work on the customer request – it omits the time that the work is in a queue, waiting to be processed. Because lead time is what the customer experiences, we typically focus our process improvement attention there instead of on process time. However, the proportion of process time to lead time servers as an important measure of efficiency.

THE DEVOPS IDEAL: DEPLOYMENT LEAD TIMES OF MINUTES

In the DevOps ideal, developers receive fast, constant feedback on their work, which enables them to quickly and independently implement, integrate, and validate their code, and have the code deployed into the production environment.

Teams achieve this by continually checking small code changes into a version control repository, performing automated and exploratory testing against it, and deploying it into production (automatically or with manual approval). This enables us to have a high probability that our changes will operate as designed in production and that any problems can be quickly detected and corrected.

In this ideal DevOps process, our deployment lead time is measured in minutes or, in the worst case, hours.

OBSERVING %C/A AS A MEASURE OF REWORK

Another key metric in the technology value stream is percent complete and accurate (%C/A). This metric measures the quality of the output of each step in the process. It can be obtained by asking the customers that sit at our left what percentage of the time they receive work that is "usable as-is", meaning that they can do their work without having to correct the information that was provided, add missing information that should have been supplied, or clarify information that should have and could have been clearer.

CONCLUSION

There's no DevOps enterprise edition to install in our company that turns the things around in a few days. DevOps is not about perfection, is about consistency. But that's the key: everything that really matters and lasts for long periods of time requires patience and hard work.

The world is full of examples where DevOps principles are winning and improving companies in every aspect. Bringing DevOps into an organization is no small task. It can create risk and starts a process of change that someone might not like. But if we start small, there's nothing to fear. When we start we must demonstrate and broadcast early wins. We must start with small goals and incremental steps. This creates trust and as we generate successes we earn the right to expand the DevOps initiative.

MICHELE FERRACIN is the CTO at MadLab Srl. He studied software engineering at the University of Padua and proceeded to work in a fast-growing startup where there was the need to build a dev team, which he built with his passion for software technologies and management. He cares about creating a great place to work, and he applies the DevOps principles and best practices to build a self-organizing, reliable, and efficient development team. In his daily activities he writes code, helps colleagues, and mentors students at a local high-school with the Agile@Work project.

Nine Principles of DevOps Made Possible with Low-Code

Rather than being a skill, a position you can hire in your IT department, or a specific tool you can purchase, DevOps is a culture-shifting paradigm. The goals are to add increased value for the customer and to enable organizations to react faster to change in the business.

These nine principles help you develop a DevOps mindset in your organization:

1. Before you start developing, define and validate the expected impact on IT.
2. Automate infrastructure provisioning to boost your development and testing speed.
3. Mimic production as closely as possible in all non-production environments.
4. Anticipate failure by testing scenarios on top of a self-healing infrastructure.
5. Continuously integrate your code during app development.
6. Register your configurations in a centralized, shared repository and track every application change.
7. Orchestrate deployment with automation that defines all steps of the process for each application.
8. Measure app health and performance and the impact of your apps on your business and customers once they reach production.
9. Make sure all collected feedback is shared with all stakeholders.

Ultimately, DevOps is all about bridging the gap between development and operations. Low-code development platforms provide a way to cross that chasm with capabilities that enable these nine principles. Examples include a unified application lifecycle management console, embedded version control and build server with continuous integration, built-in monitoring and analytics, high-availability configuration, and centralized configuration management. Combining the speed of low-code visual development with the agility of DevOps can greatly accelerate digital transformation.

WRITTEN BY RUI MENDES
DEVOPS EXPERT, OUTSYSTEMS

PARTNER SPOTLIGHT

OutSystems Low-Code Development Platform

OutSystems is the #1 low-code platform for digital transformation — build mobile apps, web portals, mission-critical systems, and more.

CATEGORY

High productivity, low-code application development and delivery platforms

RELEASE SCHEDULE

Annual major releases and monthly feature releases.

OPEN SOURCE?

No

CASE STUDY

Van Ameyde Insurance - Providing complete claims management services to over 1000 businesses and insurance companies in over 16 countries, Van Ameyde used OutSystems to develop ECHO, a front-end system that delivers customized service for each of its clients. With an off-the-shelf installation of SAP, Van Ameyde needed a layer of flexibility in order to customize process flows for its customers, while maintaining the sophisticated fiscal, legal, and security requirements of the SAP financial system. With its deep integration to SAP, ECHO provides a fast, flexible, secure, and agile way to implement different process flows based on each client's needs.

STRENGTHS

- Visual, low-code full-stack web and mobile application development
- Change tracking and automatic change impact analysis
- One-click deployment and Continuous Delivery for even the most complex apps
- High availability, scalability, and enterprise-grade security for running apps
- Integration with everything: Easily connect your apps to any system
- Advanced ALM and monitoring capabilities for DevOps

NOTABLE CUSTOMERS

- FICO
- AXA Insurance
- Vodafone
- Volkswagen
- Randstad

WEBSITE outsystems.com **TWITTER** @OutSystems **BLOG** outsystems.com/blog

Executive Insights on the State of DevOps

BY TOM SMITH
RESEARCH ANALYST AT **DZONE**

QUICK VIEW

01. The most important elements of a successful DevOps implementation is a combination of people (culture), processes, and automation, with culture being the most important.

02. The most significant change in the evolution of DevOps is the speed at which it is being implemented across the board at start-ups and old-school enterprises.

03. DevOps is about speed: time to market, mean-time-to-resolution, feedback loops, getting new features to customers, and less development time.

To gather insights on the state of DevOps, we spoke with 22 executives at 19 companies implementing DevOps for themselves and helping clients to implement a DevOps methodology. Here's who we talked to:

- **Gil Sever,** CEO, Applitools
- **Mike Tria**, Head of Infrastructure, Atlassian
- **John Trembley**, CMO, Atmosera
- **Scott Harvey**, V.P. Engineering, Atmosera
- **Aruna Ravichandran**, VP DevOps Products and Solutions Marketing, CA Technologies
- **Flint Brenton**, CEO, CollabNet
- **Tom Hearn**, Data Center Architect, Datalink
- **Shehan Akmeemana**, CTO, Data Dynamics
- **Robert Reeves**, Co-founder and CTO, Datical
- **Anders Wallgren**, CTO, Electric Cloud
- **Job van der Voort**, Vice President of Product, GitLab
- **Ben Slater**, Chief Product Officer, Instaclustr
- **Ilya Pupko**, Chief Architect, Jitterbit
- **Tom Joyce**, CEO, Pensa
- **Stephanos Bacon**, Chief of Product, Portfolio Strategy for Application Platforms, Red Hat
- **Michael Mazyar**, CTO, Samanage
- **Eric Wahl**, IT Director, Scribe Software
- **John Joseph**, Vice President of Marketing, Scribe Software
- **Manish Gupta**, CEO and Founder, ShiftLeft
- **Martin Loewinger**, Director of SaaS Operations, SmartBear
- **Jonathan Parrilla**, DevOps Engineer, SmartBear
- **Chris McFadden**, V.P. Engineering and Operations, SparkPost

And, here's what they told us:

1. The most important elements of a successful DevOps implementation are a combination of **people, culture, processes, and automation**. Culture is the most important of these, and continues to be a huge issue. There's a people problem with IT and security, development, and operations which must be brought together from the top down through the CTO or CIO. You cannot force developers to do operations; however, you must break the silos of development, operations, and testing to form one team with the same goals and objectives.

Identify the tools and processes that you will need for your organization. You need agile velocity for development cycles. CD as a process requires automating the entire value delivery lifecycle. Automate everything you can to reduce risk, secure code more thoroughly, and have more confidence in the entire pipeline.

2. Keys to addressing security with a DevOps methodology are to implement it **early in the SDLC, use continuity, and to use automation**. You must shift left and bring security into the SDLC. Security should be built into products when they are designed. Security models are part of the original plan. Security is integrated or layered with training, coding standards, testing, and peer reviewed code. Involve security in the DevOps process and figure out how to automate security checks. Automate as much as you can, including penetration tests and static code analysis. Check versions of dependencies for known vulnerabilities.

3. The most significant change our respondents have seen in the evolution of DevOps is t**he speed at which it's being implemented across the board – at start-ups and old-school enterprises**. There's greater awareness of DevOps and what it can do, people understand its benefits, it's becoming a hard requirement for companies to be

successful and to keep IT professionals who want to remain relevant during the digital transformation.

Three years ago, the State of DevOps Report showed the market cap growth for DevOps companies outperformed the S&P 500. Now finance is asking why organizations can't release every 11.7 seconds like Amazon. The C-suite has realized the value DevOps brings to the organization and how it ultimately improves the customer experience.

4. Speed is where the greatest value is seen: time to market, mean-time-to-resolution, feedback loops, getting new features to customers, and less development time. Cultural value is also seen in breaking down the silos between developers and operations so they are attacking problems as a team rather than as adversaries.

DevOps accelerates the process of core software development. We ship quickly, have more positive feedback loops, and pick up new tools and technology faster. One company we spoke with decreased their time to market by 42% and improved the quality of their code by 48%, while also improving employee retention, efficiency, and productivity. Automation removes overhead and calendar time from the development process.

The majority of examples of real-world problems being solved by DevOps were in **financial services and ran the gamut from rolling releases at greater speed with better security**.

Examples of actual use cases included:

- An online trading company used to be able to deploy only after trading hours which meant long nights and weekends for employees. DevOps tools enabled them to deploy in 45 seconds.
- A large financial services firm was able to reduce loan funding time by 5X while reducing regression testing by 93%.
- A large U.S. bank had an existing app that stood the test of time but could not move to mobile. They did a "lift and shift" to the respondent's platform exposing what they needed to as REST services and built the user interface on top.
- A large U.S. airline changed their continuous testing paradigm, saving $500,000 while increasing code coverage by 85%.
- A large telecom company in China needed to reduce cycle times for development, building and testing. The respondent's company built a backbone of automation to get a small team of internal customers on board. Once the rest of the company saw the reduction in the time required to build, test, and deploy, all the other teams were anxious to deploy the new automated methodology.

5. The biggest obstacle to the success of DevOps implementation is **the inability to embrace change on several fronts** – culture is first and foremost, eliminating silos and fiefdoms, security, business practices, tools, and technology. Management and team members need to embrace change, embrace the mentality of fail early, fail fast, and quit worrying about the security of their current jobs because they fail to realize DevOps will take them away from mindlessly pushing buttons and put them in revenue-generating tasks that will enable them to add more value to the organization. Companies need to start with one project and a "tiger team" to implement it, then watch the organization change as their members see the impact the DevOps methodology can have once walls come down and automation is implemented.

6. Slightly more than half of the respondents had concerns regarding DevOps – **most revolved around culture and change, as well as security**. People try to cut corners without adopting the right culture or architecture. This will lead to failure and the responsibility will be laid at the feet of the methodology rather than the feet of the people responsible for the lack of proper implementation. Companies fail to realize this is a long-term initiative that begins with culture change. Because of the speed to CD, there's a negative impact on security with obvious vulnerabilities getting through production. Companies need to be implementing security from the beginning of the process.

7. Respondents' visions for the future of DevOps are wide ranging, with **ubiquity and security** being the only topics mentioned more than once. According to the feedback we received, everyone will be doing DevOps in the near future. People need to be properly trained in the functions, activities, tasks, and responsibilities. DevOps will be fully adopted along with the strategy, tools, processes, and culture necessary to create value for the consumer. DevOps will continue to expand its footprint so infrastructure becomes generic and accessible by APIs. There will be continued growth of more sophisticated APIs.

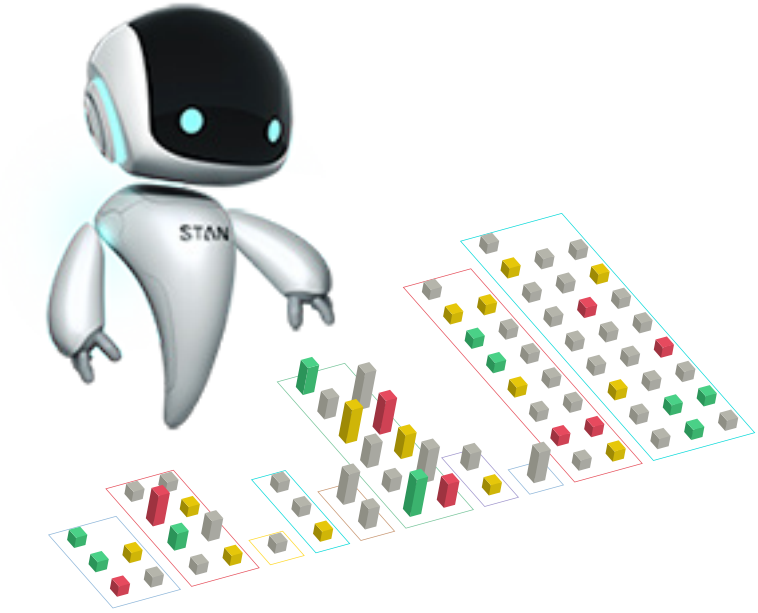
The rate of code changes and deployments will continue to increase. As such, security and DevSecOps will continue to grow out of necessity. Growth in the security space will be massive as organizations strive to improve the security of all code and applications.

8. In order to be successful with DevOps, developers need to b**e mindful of collaborating with all other team members, own their new responsibilities, and learn operations** since they will ultimately be responsible for it. Take an operations team member and a DBA to lunch. Quit thinking you're superior to your team members and get to know what you can do to make their jobs easier since you will ultimately be responsible for what they are doing. If you built it, you run it. Now is the time to reach out and learn why things need to be done a certain way. Build relationships across teams. Partner with teams to be monumentally successful. Work together to find ways to improve the process – that's the way things get dramatically better. Be part of the DevOps solution, not part of the cultural problem hindering its success.

TOM SMITH is a Research Analyst at DZone who excels at gathering insights from analytics—both quantitative and qualitative—to drive business results. His passion is sharing information of value to help people succeed. In his spare time, you can find him either eating at Chipotle or working out at the gym.

INSTANA

Stan Has Two Hobbies: Automation and AI



Any tool trying to monitor dynamic, containerized microservice applications must have AI. The environments are simply too complex to manually manage (or even configure the monitoring tool).

AI requires comprehensive automatic visibility into the full technical stack, coupled with application modeling to deliver automatic root cause determination.

Here are six fundamental skills Stan possesses around Automated Visibility and AI to help manage the performance of dynamic applications.

AUTOMATED VISIBILITY

AUTOMATIC, CONTINUOUS DISCOVERY & MAPPING

To apply an AI approach to performance management, the core model and data set must be up-to-date and impeccable, providing real-time visibility and an accurate picture of your application's structure and dependencies — all with no human configuration.

PRECISE HIGH FIDELITY VISIBILITY

AI requires precise data. For all discovered components, Instana collects the industry's most accurate monitoring data (streamed at 1 second granularity) and every request in a Trace. The data is the source for AI machine training and the basis for the deep microservice visibility.

CLOUD, CONTAINER & MICROSERVICE NATIVE

Instana is built to operate in the modern world. With zero configuration, Instana aligns with the infrastructure, clouds, containers, orchestrators, middleware and languages to accurately model and visualize dynamic microservice applications — wherever they are running.

ARTIFICIAL INTELLIGENCE

FULL STACK APPLICATION DATA MODEL

The core technology at the heart of Instana is the internal data model, called the Dynamic Graph. The Graph models all physical and logical components, the underlying technologies, dependencies and configuration. The Graph also understands logical components like traces, applications, services, clusters and tablespaces.

REAL-TIME AI-DRIVEN INCIDENT MONITORING & PREDICTION

Instana aligns alerts with business impacts and can predict impending service outages — using multiple AI methods to understand and predict application behavior. Predictive algorithms are applied to four derived KPIs (Transaction Rate, Error Rate, Latency and Saturation), leveraging the Dynamic Graph model to understand context.

AI-POWERED PROBLEM RESOLUTION AND TROUBLESHOOTING ASSISTANCE

Instana's AI-assisted troubleshooting leverages full visibility, the Dynamic Graph and AI-driven Incident management. Instana automatically identifies the most likely trigger of an Incident. Reports aggregate metrics, changes, traces and probable root cause on one screen. And predictive analysis identifies performance problems before they happen

Democratization of APM

The complexity of containerized microservices and the need for faster release cycles has led to a stronger and faster adoption of DevOps.

The DevOps transformation is a big cultural change. A fundamental paradigm of DevOps is the **Democratization** of IT (the process by which technology access rapidly becomes more accessible to more people).

Cloud and Containers have helped accelerate this trend. They require less expertise to operate. A good example is the widespread databases used by developers today.

The same is true for APM. In the past, an APM tool was used by only a small percentage of Ops members, but today ***every DevOps team member*** needs to use APM.

There are 3 drivers for Democratization of APM:

AUTOMATED VISIBILITY

A key APM function is application visibility. The complexity & dynamism of cloud native applications, and the need to support all team members, means APM must provide visibility out-of-the-box without configuration.

ARTIFICIAL INTELLIGENCE

AI enables autonomous operation, but already makes APM easier to use through automation with:

- Dynamic thresholds
- Service Quality KPIs and alerting
- Problem Prediction and Root Cause Analysis

APM USER EXPERIENCE

With more APM users (execs, PMs, SREs), the UX must change. Expert data should expand to include cost & business metrics to understand the value and cost of changes.

Non-expert users need easy access to data, presented in a new way and supported by AI to find the needle in the haystack and quickly see optimization opportunities.

Regardless of your DevOps plan, APM democratization will impact 2018. Ensure a successful year by including an APM solution with built-in automation and Artificial Intelligence.

WRITTEN BY MIRKO NOVAKOVIC - CEO AND CO-FOUNDER OF INSTANA

PARTNER SPOTLIGHT

Instana APM for Microservice Applications

AI-Powered APM for Microservice Applications

INSTANA

CATEGORY
Application Performance Management

OPEN SOURCE?
No

RELEASE SCHEDULE
V17.2 (Major release 2 or 3 times / year)

CASE STUDY

In just 2 years, Fintech company ClearScore had grown beyond the capabilities of its Java application. They migrated to microservices hosted in Docker containers, but their APM tool failed to match the efciency and agility they valued. Simply maintaining the monitoring system was like a full-time job. The tool lacked native support for microservices and containers, or Scala. ClearScore chose Instana's microservices-native APM. Instana delivered more detailed, context-aware insights while eliminating tool conguration by DevOps, it also identied and xed problems quicker. The team loves that Instana's automated visibliity and articial intelligence doing everything from monitoring setup and threshold setting to troubleshooting.

STRENGTHS

- Full-stack application mapping and visibility of performance
- Supporting 80+ technologies: middleware, database, orchestration, containers, and 9 languages
- Predictive service incident monitoring and automatic root-cause analysis
- AI-assisted troubleshooting

NOTABLE CUSTOMERS

- Audi
- ClearScore
- Douglas
- Conrad
- Follett

WEBSITE instana.com | **TWITTER** @InstanaHQ | **BLOG** instana.com/blog

Solutions Directory

This directory contains software configuration, build, container, repository, monitoring, and application performance management tools, as well as many other tools to help you on your journey toward Continuous Delivery. It provides the company name, product information, open source data, and product category information gathered from vendor websites and project pages. Solutions are selected for inclusion based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
Amazon	Amazon ECS	Container Management	Free Tier Available	aws.amazon.com/ecs
Apache Software Foundation	Apache Ant	Build Management	Open Source	ant.apache.org
Apache Software Foundation	Apache Archiva	Repository Management	Open Source	archiva.apache.org
Apache Software Foundation	Apache Maven	Build Management	Open Source	maven.apache.org
Apache Software Foundation	Apache Subversion	Software Configuration Management	Open Source	subversion.apache.org
Apache Software Foundation	JMeter	Web and Java Testing	Open Source	jmeter.apache.org
Appium	Appium	Automated Web and Mobile Testing	Open Source	appium.io
Applitools	Applitools	Visual Testing and Monitoring	Available By Request	applitools.com
Atlassian	Bamboo	Continuous Integration, Application Release Automation	30 Days	atlassian.com/software/bamboo
Attunity	RepliWeb for ARA	Application Release Automation	Free Tier Available	attunity.com/products/repliweb
BigPanda	Altert Correlation Platform	Monitoring Alert Software	21 Days	bigpanda.io
BMC	Release Lifecycle Management	Application Release Automation	Available By Request	bmc.com/it-solutions/release-lifecycle-management.html
Buildbot	Buildbot	Continuous Integration	Open Source	buildbot.net
CA	Automic Modern Software Factory	Application Release Automation, APM, CD Pipeline Security	30 Days	automic.com/products/application-release-automation

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
CA	CA Release Automation	Application Release Automation	N/A	ca.com/us/products/ca-release-automation.html
CenturyLink	Cloud Application Manager	Application Lifecycle Management	N/A	ctl.io/cloud-application-manager/application-lifecycle-management
Chef	Chef Automate	Continuous Deployment Platform	Demo Available By Request	chef.io/automate
Chef	Chef	Application and Infrastructure Release Automation	Open Source	chef.io/chef/
CircleCI	CircleCI	Continuous Integration	Free Tier Available	circleci.com
Cisco	AppDynamics	Application Performance Management	15 Days	appdynamics.com
CloudBees	CloudBees Jenkins Enterprise	Continuous Integration, Application Release Automation, continuous delivery	14 Days	cloudbees.com
Codeship	Codeship	Continuous Integration	14 Days	codeship.com
CollabNet	TeamForge ALM	Application Lifecycle Management	30 Days	collab.net/products/teamforge-alm
Cucumber	Cucumber	Automated Rails Testing	Open Source	cucumber.io
Datadog	Datadog	IT Stack Performance Management	14 Days	datadoghq.com
Datical	Datical Database Release Automation	Database Release Automation	Demo Available By Request	datical.com
Docker	Docker	Container Management	Open Source	docker.com/community-edition
Eclipse	Hudson	Continuous Integration	Open Source	eclipse.org/hudson
Electric Cloud	ElectricFlow	Application Release Automation	Free Tier Available	electric-cloud.com/products/electricflow
FitNesse	FitNesse	Acceptance Testing Framework	Open Source	fitnesse.org
Free Software Foundation	Concurrent Versions Systems (CVS)	Software Configuration Management	Open Source	savannah.nongnu.org/projects/cvs
Git	Git	Software Configuration Management	Open Source	git-scm.com
Gitlab	GitLab	Code Review, Continuous Integration, Continuous Delivery	Open Source	about.gitlab.com
Gradle	Gradle	Build Automation	Open Source	gradle.org

Modern Application Platform and Lifecycle: The Future of DevOps

We are seeing an explosion of data, as well as advances in artificial intelligence (AI) and machine learning, that are driving another wave of digital transformation and reimagination. What began with tech companies is now becoming commonplace inside traditional companies trying to reinvent themselves to suit modern consumers and business realities. But beneath the headlines and WSJ articles, the unsung hero is the design and delivery of modern software applications.

Software is the language of how you engage customers, reach new users, understand their data, promote products or services, and process orders. To do this well, today's software is going bespoke, broken down into small pieces of software called microservices that are designed for very specific jobs. Delivering high-quality, modern applications requires a top-to-bottom embrace of modern application lifecycle management tools and processes. With the right DevOps tools, developers and IT practitioners can streamline continuous integration and delivery (CI/CD) and get innovative applications into the hands of customers more quickly.

Containers are a key component of DevOps, microservices, and modern software engineering. Over the past decade, web-scale companies, including Google, have shown a dramatic commitment to containers. In 2014, the tech giant released Kubernetes – an open source incarnation of the famous Borg system which is now the leading container orchestration engine. Kubernetes provides an abstraction layer that allows you to schedule and deploy container applications in physical or virtual environments.

Container-based microservices are an attractive DevOps pattern because they enable speed to market. With each microservice being developed, deployed and run independently (often using different languages, technology stacks, and tools), microservices allow organizations to "divide and conquer" — while scaling teams and applications more efficiently. When the pipeline is not locked into a monolithic configuration — of either toolsets, component dependencies, release processes, or infrastructure — there is a unique and portable ability to better scale development and operations.

In order to better support full-stack cloud-native environments, the CNCF has taken all these best practices and implemented training and certifications targeting both developers and operators. The CNCF approach ensures that developers don't adopt restrictive platforms, and that application teams can maintain the right levels of freedom and control for their application environments. Kubernetes and Prometheus are just 2 of 14 high-velocity open source projects that CNCF hosts.

Delivering high-quality, modern applications requires a top-to-bottom embrace of modern application lifecycle management tools and processes.

"As cloud native technologies make cloud portability possible, the broader story is that Kubernetes and containers are rapidly displacing virtualization as the dominant software development paradigm," said Dan Kohn, Executive Director of the Cloud Native Computing Foundation.

The shifts in cloud computing, applications, and data have changed the technology and business conversation from just "How are you reducing my costs?" to also "How are you accelerating my business?" As you embark on your journey, CNCF provides the flexibility of an open ecosystem of DevOps tools and technologies to deliver agility, portability, and operational efficiencies at lower costs and with no lock-in.

WRITTEN BY DEE KUMAR
VP MARKETING, CNCF

CLOUD NATIVE COMPUTING FOUNDATION

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
Gridlastic	Gridlastic	Automated Web Testing	Available By Request	gridlastic.com
Hashicorp	Vagrant	Configuration Management	Open Source	vagrantup.com
IBM	Rational	Software Configuration Management	N/A	ibm.com/software/rational/strategy
IBM	Urbancode Build	Continuous Integration, Build Management	Available By Request	developer.ibm.com/urbancode/products/urbancode-build
IBM	Urbancode Deploy	Application Release Automation	Available By Request	developer.ibm.com/urbancode/products/urbancode-deploy
Inedo	Buildmaster	Application Release Automation	Free Tier Available	inedo.com/buildmaster
Inflectra	Rapise	Automated Web Testing	Available By Request	inflectra.com/Rapise
InfluxData	InfluxData	Monitoring and Analytics Program	14 Days	influxdata.com/products
Instana	Instana	Automated APM Platform	Available By Request	instana.com
Jenkins	Jenkins	Continuous Integration	Open Source	jenkins-ci.org
JetBrains	TeamCity Enterprise	Continuous Integration, Application Release Automation	60 Days	jetbrains.com/teamcity
jFrog	Artifactory	Repository Management	Available By Request	jfrog.com/artifactory/
Junit	JUnit	Unit Testing Framework	Open Source	junit.org
Librato	Librato	Monitoring Alert Software, APM	14 Days	librato.com
Linux Foundation	CF Application Runtime	PaaS	Open Source	cloudfoundry.org/why-cloud-foundry
Mercuiral	Mercurial	Software Configuration Management	Open Source	mercurial-scm.org
Micro Focus	ALM Octane	Application Lifecycle Management	Available By Request	software.microfocus.com/en-us/products/alm-octane/overview
Micro Focus	AccuRev	Software Configuration Management	30 Days	microfocus.com/products/change-management/accurev
Micro Focus	Deployment Automation	Application Release Automation	N/A	microfocus.com/products/deployment-automation
Microsoft	Team Foundation Server	Software Configuration Management	Free Solution	visualstudio.com/en-us/news/releasenotes/tfs2017-relnotes
MidVision	RapidDeploy	Application Release Automation	Demo Available By Request	midvision.com/rapiddeploy-overview

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
Nagios	Nagios Core	Infrastructure Monitoring	Open Source	nagios.org/projects/nagios-core
New Relic	New Relic	Application Performance Management	Demo Available By Request	newrelic.com
NuGet	NuGet	Repository Management	Open Source	nuget.org
Nunit	NUnit	Unit Testing Framework	Open Source	nunit.org
OpsGenie	OpsGenie	Monitoring Alert Software	Available By Request	opsgenie.com
Outsystems	Outsystems	Application Development and Deployment Platform	Free Tier Available	outsystems.com
PagerDuty	PagerDuty	Monitoring Alert Software	Available By Request	pagerduty.com
Panaya	Release Dynamics (RDx)	Application Lifecycle Management & Continuous Delivery	Demo available by request	panaya.com
Parasoft	Parasoft Development Testing Platform	Automated Web and API Testing	Available By Request	parasoft.com/product/development-testing-platform
Perforce	Helix	Software Configuration Management	Free Tier Available	perforce.com/helix
Plutora	Plutora Release	Application Release Automation	Available By Request	plutora.com/platform/plutora-release
Portworx	Portworx	Database Release Automation	Demo Available By Request	portworx.com
Puppet Labs	Puppet	Configuration Management	Free Tier Available	puppetlabs.com
Rake	Rake	Build Automation	Open Source	github.com/ruby/rake
Ranorex	Ranorex	Automated Web and Desktop Testing	Available By Request	ranorex.com
Red Gate Software	SQL Toolbelt	Database CI and Release Automation	14 Days	red-gate.com/products/dlm/dlm-automation
Red Hat	Ansible Tower	Configuration Management, Application Release Automation	Open Source	ansible.com/
Rocket Aldon	ALM and DevOps	Application Release Automation	Demo Available By Request	rocketsoftware.com/product-categories/application-lifecycle-management-and-devops
Rogue Wave Software	Zend Server	Application Release Automation for PHP	30 Days	zend.com/en/products/zend_server
Sahi	Sahi	Automated Web Testing	30 Days	sahipro.com
SaltStack	Salt	Configuration Management	Open Source	saltstack.com

COMPANY	PRODUCT	CATEGORY	FREE TRIAL	WEBSITE
Sauce Labs	Sauce Labs	Automated Web and Mobile Testing	14 Days	saucelabs.com
Selenium	Selenium WebDriver	Automated Web Testing	Open Source	seleniumhq.org
ShiftLeft	ShiftLeft	Security Management Platform	Demo Available By Request	shiftleft.io
SignalFX	SignalFX	Analytics Dashboards, Collaboration Tools, Alerting	14 Days	signalfx.com
SmartBear Software	SoapUI	Automated Web and API Testing	Open Source	soapui.org
Solano Labs	Solano Labs	Continuous Integration	14 Days	solanolabs.com
Sonatype	Nexus	Repository Management	Open Source	sonatype.com/nexus-repository-sonatype
Tellurium	Tellurium	Automated Web Testing	Free Tier Available	te52.com
TestingBot	TestingBot	Automated Web Testing	Available By Request	testingbot.com
TestNG	TestNG	Unit Testing Framework	Open Source	testng.org
The Linux Foundation	Kubernetes	Container Management	Open Source	kubernetes.io
Thoughtworks	Go	Application Release Automation	Open Source	gocd.io/
TravisCI	TravisCI	Continuous Integration	Open Source	travis-ci.org
VictorOps	VictorOps	Monitoring Alert Software	Demo Available By Request	victorops.com
Watir	Watir	Automated Web Testing	Open Source	watir.com
WeaveWorks	Weave Cloud	Continuous Deployment Platform	Available By Request	weave.works/product/cloud
Windmill	Windmill	Automated Web Testing	Open Source	github.com/windmill
Xebia Labs	XL Deploy	Application Release Automation	Demo Available By Request	xebialabs.com/products/xl-deploy
XMatters	XMatters IT Management	Alerting Software, Collaboration Platform	14 Days	xmatters.com/products
xUnit	xUnit	Unit Testing Framework	Open Source	xunit.github.io

GLOSSARY

ARTIFACT
Any description of a process used to create a piece of software that can be referred to, including diagrams, user requirements, and UML models.

AUTONOMY
The ability to make changes with the resources currently available, without the need to defer to something or someone higher up in the hierarchy.

BRANCHING
The duplication of an object under review in source control so that the same code can be modified by more than one developer in parallel.

COMMIT
A way to record the changes to a repository and add a log message to describe the changes that were made.

COMPLEX-ADAPTIVE SYSTEMS
Any system made of a collection of similar, smaller pieces that are dynamically connected and can change to adapt to changes for the benefit of a macro-structure.

CONTAINERS
Resource isolation at the OS (rather than machine) level, usually (in UNIX-based systems) in user space. Isolated elements vary by containerization strategy and often include file system, disk quota, CPU and memory, I/O rate, root privileges, and network access. Much lighter-weight than machine-level virtualization and sufficient for many isolation requirement sets.

CONTINUOUS DELIVERY
A software engineering approach in which continuous integration, automated testing, and automated deployment capabilities allow software to be developed and deployed rapidly, reliably, and repeatedly with minimal human intervention.

CONTINUOUS DEPLOYMENT
A software development practice in which every code change goes through the entire pipeline and is put into production automatically, resulting in many production deployments every day. It does everything that Continuous Delivery does, but the process is fully automated, and there's no human intervention at all.

CONTINUOUS INTEGRATION
A software development process where a branch of source code is rebuilt every time code is committed to the source control system. The process is often extended to include deployment, installation, and testing of applications in production environments.

CONTINUOUS QUALITY
A principle that preaches the continuous quest for quality across the entire SDLC, starting from requirements definition, code development, testing, and operations. Another key area of focus for Continuous Quality is the application code pipeline orchestration. There are many opportunities to negatively impact the quality of an application when code is being manually moved across environments.

CONTINUOUS TESTING
The process of executing unattended automated tests as part of the software delivery pipeline across all environments to obtain immediate feedback on the quality of a code build.

DEPLOYMENT PIPELINE
The process for deploying software from development to production, often including tools such as source control and CI servers, as well as testing checks.

DEVOPS
An IT organizational methodology where all teams in the organization, especially development teams and operations teams, collaborate on both development and deployment of software to increase software production agility and achieve business goals.

EVENT-DRIVEN ARCHITECTURE
A software architecture pattern where events or messages are produced by the system, and the system is built to react, consume, and detect other events.

MODEL-BASED TESTING
A software testing technique in which the test cases are derived from a model that describes the functional aspects of the System Under Test (SUT). Visual models can be used to represent the desired behavior of a SUT, or to represent testing strategies and a test environment. From that model manual tests, test data, and automated tests can be generated automatically.

PAIR PROGRAMMING
A software development practice where two developers work on a feature, rather than one, so that both developers can review each others' code as it's being written in order to improve code quality.

PRODUCTION
The final stage in a deployment pipeline where the software will be used by the intended audience.

SOURCE CONTROL
A system for storing, tracking, and managing changes to software. This is commonly done through a process of creating branches (copies for safely creating new features) off of the stable master version of the software and then merging stable feature branches back into the master version. This is also known as version control or revision control.

SPRINT
A set period of time in which a development team works on a predetermined set of features to be completed by the end of the sprint. Any activities not completed at the end of a sprint may be moved over to the next sprint, or re-evaluated before work continues on them.

STAGING ENVIRONMENT
Used to test the newer version of your software before it's moved to live production. Staging is meant to replicate as much of your live production environment as possible, giving you the best chance to catch any bugs before you release your software.

TECHNICAL DEBT
A concept in programming that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution.

TEST AUTOMATION
The use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes.

WHAT'S SLOWING DOWN EVERY STEP IN YOUR BUILD PROCESS?

Slow feedback can hold back even great development teams. By testing every build and gathering fast feedback, you can unleash the potential of continuous integration & delivery.

Execute optimal test scope (smoke, functional, non-functional) at the earliest stages - and every step along the way.

Perfecto - Continuous Testing for Devops.

Perfecto

perfectomobile.com/ditchtheparachute

The Importance of Continuous Testing

As DevOps goes mainstream, software quality is shifting from a single role to a shared software development team responsibility. Most DevOps conversations focus on culture or CI/CD as a core practice. However, there's something that lacks enough emphasis in the conversation. The missing element is a central reason why many enterprises fail to realize the time-to-market benefits promised-slow feedback.

It's too simple to point out that slow feedback drags down potential velocity. Automating everything and especially testing is hard. Test automation efforts too often focus on unit, smoke and regression. Important, but not enough!

Teams realizing fast feedback are shifting towards automating testing of new features within each sprint rather than waiting for subsequent sprints to catch up test scripts for new features. To achieve this, teams progressing towards faster time to deployable code are adopting Acceptance Test Drive Development along with BDD. Combined, they pull test automation efforts to the earliest development phase, enabling testing to keep pace with new features.

Continuous Testing – Executing automated tests in every phase of development attacks the slow feedback problem. Three components enable teams to accelerate feedback loops:

1. LAB: A stable test environment integrated into the delivery pipeline.
2. AUTOMATION: Reliable tests built as new features are developed and executed across the platforms typically used by a customer.
3. Fast feedback: Tests that return complete data for effective troubleshooting.

Cut the slow feedback cord by implementing continuous testing.

INTRODUCING THE

AI Zone

What's new with Predictive Analytics, Natural Language Processing, and Neural Nets? Check out DZone's new AI Zone – a central place for AI resources and tutorials.

Keep a pulse on the industry and go beyond the hype. This Zone gives you access to practical applications and use cases for AI technologies like: Machine Learning, Cognitive Computing, and Chatbots.

Visit the Zone

MACHINE LEARNING

COGNITIVE COMPUTING

CHATBOTS

DEEP LEARNING