An Unlikely Union: DevOps and Audit

Information security and compliance practices

DevOps Enterprise Forum

IT REVOLUTION

An Unlikely Union: DevOps and Audit

Information security and compliance practices

DevOps Enterprise Forum

IT Revolution
Portland, Oregon

An Unlikely Union: DevOps and Audit
Information security and compliance practices

IT Revolution
Portland, Oregon
info@itrevolution.net

For quantity purchases by corporations, associations, and others, please contact the publisher at **orders@itrevolution.net**.

Cover design by Brian David Smith and Mammoth Collective.
Interior layout generated by O'Reilly Atlas.

Table of Contents

Preface

The DevOps Enterprise Forum, facilitated by IT Revolution, brought together 50 technology leaders and thinkers in the DevOps Enterprise community for three days in Portland, Oregon in May 2015, with the goal of organizing in to task teams and creating written guidance on the best-known methods for overcoming the top obstacles in the DevOps Enterprise community.

We tackled the five key areas identified by the community at the 2014 DevOps Enterprise Summit:

- Better strategies and tactics for creating automated tests for legacy applications
- Addressing culture and leadership aspects during transforming
- Top approaches to organizational design, roles and responsibilities
- Information security and compliance practices
- Identifying metrics to best improve performance with DevOps initiatives.

For three days, we broke into groups based on each of the key areas and set to work, choosing teams, sometimes switching between teams, collaborating, sharing, arguing… and writing.

After the Forum concluded, the groups spent the next six months working together to complete and refine the work they started together.

Our goal was to generate content in the form of white papers, articles, or other resources in time to share that guidance with the technology community during the DevOps Enterprise Summit 2015 in October.

We are proud to share the outcomes of the hard work, dedication, and collaboration of this amazing group of people.

—Gene Kim
October 2015

Introduction

Many organizations are adopting DevOps patterns and practices, and are enjoying the benefits that come from that adoption: More speed. Higher quality. Better value.[1] However, many organizations often get stymied when dealing with information security, compliance, and audit requirements. There seems to be a misconception that DevOps practices won't work in organizations which are under SOX or PCI regulations. In this paper, we will provide some high-level guidance on three major concerns about DevOps Practices:

1. DevOps and Change Control
2. DevOps and Security
3. DevOps and Separation of Duties

1 2015 State of DevOps Report, https://puppetlabs.com/2015-devops-report.

1 DevOps & Change Control

Achieve Harmony in a World that Demands Continuous Delivery

Organizations struggle with obtaining buy-in and implementing DevOps methodologies because security, compliance, and audit stakeholders (both internal and external) believe that change control requirements cannot be met. These stakeholders often tend to disrupt adoption of DevOps before an organization can explore its potential for implementation.

The intent of a strict change control process is to reduce the risk of implementing changes that may lead to increased operational failures, poor transaction processing, and unreliability of the system. The most common audit concern about DevOps is that developers can deploy their own code without any formal change approval processes. **The DevOps Audit Defense Toolkit** describes an auditor's perspective in this way:

> The absence of change approval controls creates the risk of untested and unauthorized code being introduced into production.

Small companies can be quite successful in implementing a lean change approval process that allows them to do frequent deployments. Because these companies typically have a small data center footprint, limited applications and small sized teams, there is a higher probability that problems will be detected through their lean change approval process. In a mid- to large-sized company the probability of detecting a problem is potentially reduced due to the volume, complexity, and larger team sizes. Companies that decide to move to a DevOps model find the transition difficult as manual processes become blockers for rapid rates of change.

Looking closely, we find that contrary to the general perception, DevOps practices offer better risk management and ease out change control process.

Smaller batch size delivered faster and more frequently

It is well known that risk is higher for large amount changes. It is also known that delays in delivering changes involve higher risks and higher costs. Changes that have lower risks go through a leaner change control process.

Single set of large amount of changes at once usually is stressful due to too many unknowns and dependencies. It usually involves lot of planning, process overheads, teams of IT associates spending nights over "production release". It also involves a detailed "roll back" plan that is equally complicated.

This is exactly where DevOps practices win. DevOps practices allow teams to deliver smaller change sets to production more frequently with lesser risk and much lesser change control process overheads. If change delivery is more frequent, it will have better repeatability and will make everyone involved more confident about the process. Martin Fowler has described this very eloquently in his pattern titled "**FrequencyReducesDifficulty**."

"Emergency Change Control" may be the norm!

An interesting pattern is often observed in almost all IT organizations. While normal release process have complex, manual change control process; emergency changes go through faster! In fact, controls for emergency changes are designed lean and are compliant with audit and regulatory requirements.

The idea here is not to literally make every change an emergency change, but to prove that small amount of changes can have lean change control process, and be compliant at the same time.

Automated change control

Traditional change control requires manual approvals and verifications. Many of these manual verifications can be achieved programmatically and securely by incorporating business logic, pre-defined thresholds, and automated controls throughout the delivery pipeline.

As examples, approvals may require: proof static code analysis and verification of change sets; proof of successful regression, load, and performance testing; proof of security scan and testing testing of the entire stack etc. Almost all of these can be automated at many different levels depending upon an organization's DevOps maturity.

Incorporating these approvals in an automated fashion allows for continuous deployment, which is a fundamental benefit of the DevOps model. In addition, risk is well managed in this model as the automation of change management incorporates all the security and audit checks that a formal change approval process provides.

There are significant and proven business and operational advantages to implementing DevOps. Organizations should explore transformation and not allow antiquated security and audit concerns to disrupt progress.

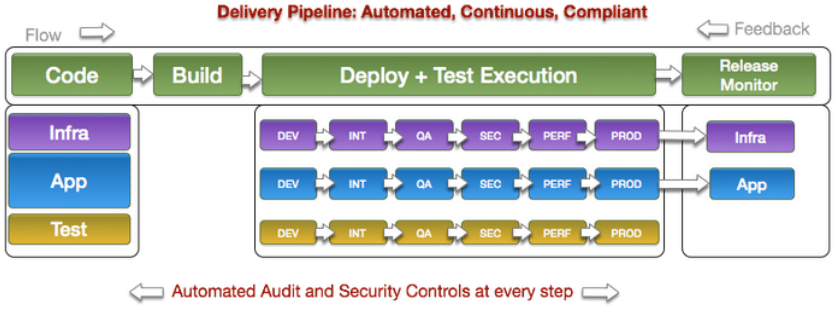
DevOps with Security: Secured Delivery Pipeline 2

Most organizations undertake a good amount of IT spending in the areas of Infrastructure and Network security. However, focus and spend on software security is disproportionately low.[1]

While securing infrastructure and network is absolutely necessary, the focus on software security must be addressed in today's IT organizations.

Implementing a secured delivery pipeline will allow organizations to ensure better security control, less risk, and better compliance.

A delivery pipeline can be described as in the diagram below:



1 The "Continuous Security: 5 Ways DevOps Improves Security" presentation slideshare by David Mortman, http://www.slideshare.net/SonatypeCorp/continuous-security-5-ways-devops-improves-security/20

Code

There are three types of codes: Application, Infrastructure, and Test. Each of these must be stored in a version control system (such as Git). Developers' access to the code base is audited. Every change in the code base is audited.

The code base also can be "scanned" to detect user id, passwords, access keys, etc. in clear text. During code design and peer-review special attention should be given to detect handing of unencrypted/untokenized sensitive data within the code. Application interfaces should be reviewed to ensure secured access to the interfaces.

Developers can also scan their code before checking in changes. Almost all modern day's development tools have built in features/plugins to make this seamless.

Build

During build process, infrastructure code, and application code can be scanned and analyzed thoroughly.

Application code should be scanned for security vulnerabilities and other software defects during a Continuous Integration cycle.

Third-party libraries (including open source libraries) should be scanned for security and legal/licensing vulnerabilities. It is also a good practice to keep third-party libraries up to date.

Some organizations fail builds if there are security vulnerabilities detected during the Continuous Integration builds.

Deployment and Test Execution

Deployment process should be automated and/or scripted for reproducibility. Deployment process should keep track of software versions and their flow across the environments. Each deployment should be followed by automated test execution. Security testing should be a part of test automation. Test results should be stored to ensure traceability and help troubleshoot.

Release

Release process in a large enterprise typically involves a lot of planning, approval processes and manual deployment of new software into production environment. This traditional release process usually involves high risk. In DevOps world, with smaller and faster deployment cycle with reduced risk, a **BlueGreenDeployment** pattern eases out the release process and reduces risk even further. Below is how Martin Fowler described the pattern:

> The blue-green deployment approach does this by ensuring you have two production environments, as identical as possible. At any time one of them, let's say blue for the example, is live. As you prepare a new release of your software you do your final stage of testing in the green environment. Once the software is working in the green environment, you switch the router so that all incoming requests go to the green environment—the blue one is now idle.

Blue-green deployment gives the ability to perform a rapid rollback in case something goes wrong.

Canary Release is also an effective pattern to release newer version of software in production. Jez Humble describes this pattern quite nicely in his book ***Continuous Delivery***. Canary release pattern may be used for large footprint applications where having two identical large production environment becomes challenging.

DevOps and Separation of Duties 3

Separation of duties is not something that was invented in the IT world. In fact, there are use cases where separation of duties is crucial:

We know that two keys are needed to launch the warhead. The person signing the checks shouldn't also be authorizing the expenses. (Checks are sent to a PO Box that check signer doesn't have access to. Ops administrators don't have write access to logs, so they can't cover their tracks.)

Many organizations implement part of their IT Security Requirements through Separation of Duties. The main idea behind this approach is to reduce chances of fraud and security breaches. The problem is that the analogy of Separation of Duties we see in the physical world doesn't quite work out the same way. In fact, in most cases a strict Separation of Duties implementation is an anti-pattern and is a key identifier for a low-performance IT organization.

Developers write the code and then hand off the code to a system administrator to actually deploy and run it. Along the way there are approval processes wherein a manager or director gives the order for the system administrator to deploy the code.

Separation of Duties & PCI/DSS

The major assumption that many IT organizations have is that they believe that the implementation of Separation of Duties requires separate personnel with completely separate functional roles in order to achieve compliance.

Section 6.4.2 is the only section of PCI/DSS v 3.1 where the phrase 'Separation of Duties' is found:

> 6.4.2 Separation of duties between development/test and production environments[1]

1 Payment Card Industry (PCI) Data Security Standard, v3.1, April 2015, https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf

The guidance for 6.4.2 is quite helpful:

> The intent of this requirement is to separate development and test functions from production functions. For example, a developer may use an administrator-level account with elevated privileges in the development environment, and have a separate account with user-level access to the production environment.
>
> Reducing the number of personnel with access to the production environment and cardholder data minimizes risk and helps ensure that access is limited to those individuals with a business need to know.
>
> The intent of this requirement is to ensure that development/test functions are separated from production functions. For example, a developer may use an administrator-level account with elevated privileges for use in the development environment, and have a separate account with user-level access to the production environment.
>
> In environments where one individual performs multiple roles (for example application development and implementing updates to production systems), duties should be assigned such that no one individual has end-to-end control of a process without an independent checkpoint. For example, assign responsibility for development, authorization and monitoring to separate individual.[2]

Based on the above guidance, the following sections will describe ways to implement Separation of Duties in a better way.

Automation

Automating production deployment process is a good way to ensure that no human being executes the production deployment—it is done by the system. The automation (script, tools configuration, etc.) is implemented both by Dev and Ops then reviewed and tested for correctness. It is a good practice to use the same automation to deploy to non-production environments. This allows the team to test the deployment automation and gain confidence.

Separate Accounts

Many organizations attempt to implement 6.4.2 by making blanket rules like "Only operations staff may have access to production environments" as a method for ensuring compliance.

2 Payment Card Industry (PCI) Data Security Standard, v2.0, October 2010, https://www.pcisecuritystandards.org/documents/navigating_dss_v20.pdf

As alternative approach, developers can use a web form to request temporary access to production systems on which their code runs to troubleshoot production issues. This access can be granted (and logged) automatically and their credentials are revoked automatically in certain time frame.

It is worth noting that the PCI/DSS requirement is to use two different accounts, not necessarily two different human beings.

In a highly automated DevOps environment, the same PCI/DSS requirement can also be implemented via the use of system level accounts that evaluate and promote codes that meet predefined quality standards (code coverage, test coverage, test success rate, etc.).

Independent Checkpoints

PCI/DSS requires no single individual to have complete "end-to-end control of a process without an independent checkpoint." Separation of Duties is also used in most organizations to implement this requirement.

Much like in the "two keys" scenario as discussed earlier, there can be "cryptographic keys" for software to progress down the pipeline from development to production. When a developer checks in their code they sign it with their personal cryptographic signature. When the code is reviewed by their peer reviewer, the code is again signed. During each stage of deployment as the code goes through the CI pipeline, the code can be signed giving a mathematical assurance that the code that was reviewed was not tampered with the entire way to production.

Furthermore, when that code is in production, it can be verified to be the desired bits.

Conclusion

As DevOps practitioners, we do acknowledge that there is a perceived notion of friction between DevOps and Audit/Compliance. Taking a closer look at the main friction points, we find that the conflict is just a perception. In this paper, we attempted to go one step further and prove that DevOps in fact can be considered as a practice that offers better Audit/Compliance as compliance.

Collaborators A

- James DeLucia, Director and Leader for Certification Services, EY CertifyPoint
- Paul Duvall, Chairman and CTO at Stelligent, Author of *Continuous Integration and DevOps in AWS*
- Mustafa Kapadia, DevOps Service Line Leader, IBM
- Gene Kim, Author and Researcher
- Dave Mangot, Director of Operations, Librato, Inc.
- Tapabrata "Topo" Pal, Director, Next Generation Infrastructure, Capital One
- James Wickett, Sr. Engineer, Signal Sciences Corp
- Julie Yoo, Vice President, Information Security Compliance at Live Nation

Acknowledgments

IT Revolution wishes to thank the following sponsors for making the DevOps Enterprise Forum possible.

We wish to thank all the participants of the 2015 DevOps Enterprise Forum

- Steve Barr, Executive Director, Operations at CSG International
- Ross Clanton, Senior Group Manager, Target
- Jason Cox, Director of Systems Engineering, The Walt Disney Company
- Dominica DeGrandis, Director, Learning & Development, LeanKit
- James DeLuccia, Director and Leader for Certification Services, EY CertifyPoint
- Jason DuMars, Senior Director of Technical Operations, SendGrid
- Paul Duvall, Chairman and CTO, Stelligent, Author of Continuous Integration and DevOps in AWS

- Damon Edwards, Managing Partner DTO Solutions, Inc
- Nicole Forsgren, PhD, Director Organizational Performance and Analytics, Chef
- Jeff Gallimore, Partner, Excella Consulting
- Gary Gruver, President, Practical Large Scale Agile LLC
- Sam Guckenheimer, Product Owner, Microsoft
- Mirco Hering, DevOps Lead APAC, Accenture
- Christine Hudson, Solutions and Product Marketing, Rally Software
- Jez Humble, Owner, Jez Humble & Associates LLC
- Mustafa Kapadia, DevOps Service Line Leader, IBM
- Nigel Kersten, CTO, Puppet
- Gene Kim, Author and Researcher
- Courtney Kissler, Vice President of E-Commerce and Store Technologies, Nordstrom
- Dave Mangot, Director of Operations, Librato, Inc.
- Mark Michaelis, Chief Technical Architect, IntelliTect
- Heather Mickman, Senior Group Manager, Target
- Chivas Nambiar, Director DevOps Platform Engineering, Verizon
- Steve Neely, Director of Software Engineering, Rally Software
- Tapabrata “Topo” Pal, Product Manager, CapitalOne
- Eric Passmore, CTO MSN, Microsoft
- Mark Peterson, Sr. Director, Infrastructure Engineering & Operations, Nordstrom
- Scott Prugh, Chief Architect, CSG International
- Terri Potts, Technical Director, Raytheon IIS Software
- Walker Royce, Software Economist
- Jeremy Van Haren, Director of Software Development, CSG International
- Jeff Weber, Managing Director, Protiviti
- James Wickett, Sr. Engineer, Signal Sciences Corp
- John Willis, Director of Ecosystem Development, Docker
- Tim Wilson, Solution Architect, IBM
- Elizabeth Wittig, Field Solutions Engineer, Puppet
- Julie Yoo, Vice President, Information Security Compliance, Live Nation`

And we would also like to acknowledge the organizers, scribes, editors, and designers who lent their support and attention to make the event and these artifacts possible:

Alex Broderick-Forster, Alanna Brown, Robyn Crummer-Olson, William Hertling, Aly Hoffman, Todd Sattersten, and Brian David Smith.