DT7

DevOps Case Study
Thursday, November 8th, 2018 1:30 PM

Experiences Bringing Continuous Delivery to the DoD and DHS

Presented by:

Gene Gotimer

Coveros, Inc.

Brought to you by:

350 Corporate Way, Suite 400, Orange Park, FL 32073
888-268-8770 · 904-278-0524 - info@techwell.com - http://www.starwest.techwell.com/

Gene Gotimer

Gene Gotimer is a senior architect at Coveros Inc., a software company that uses agile methods to accelerate the delivery of secure, reliable software. As a consultant, Gene works with his customers to build software better, faster, and more securely by introducing agile development and DevOps practices such as continuous integration, repeatable builds, unit testing, automated functional testing, analysis tools, security scanning, and automated deploys. He has successfully brought these techniques to commercial and government clients, including the Department of Defense andÂ Department of Homeland Security. Gene feels strongly that repeatability, quality, and security are all strongly intertwined; each of them is dependent on the other two, which just makes agile and DevOps that much more crucial to software development.

coveros

Agility. Security. Delivered.

Bringing Continuous Delivery to the DoD and DHS

Gene Gotimer

Tale of Two Projects

coveros

U.S. Department of Defense

Exemplar that Agile would work within the DoD

U.S. Department of Homeland Security

Build a Continuous Delivery pipeline for Agile teams

U.S. Department of Defense

The DoD Project

- 4½ Years, September 2009 – March 2014
- High-risk releases every 6 months or so
  - Freeze 2-4 weeks in advance
  - Deploy Friday evening to Sunday afternoon
  - Repair broken functionality Monday and Tuesday (and on)
- Pre-production environments radically different than Production
  - Different compliance rules
- Development risks
  - No unit tests
  - No continuous integration
  - No automated testing

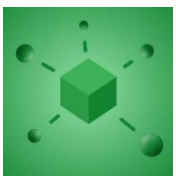
The Approach

- Started with things that were in our control
  - Dev and Test environments
  - Development process
- Made changes behind the scenes
  - Free/open source tools
  - Very few software restrictions
  - Easy to integrate into our CI system
  - Small changes
- Disclose the changes when there was a win
  - “It’s easier to ask forgiveness than it is to get permission” - Admiral Grace Hopper
  - Use as justification for higher environments

Continuous Integration

- Trouble explaining “integration”
  - between two or more developers
  - not between systems
- Set up SecureCI one afternoon

Sonatype



sonarqube

Continuous Integration

- This gave us a strong basis for Continuous Delivery later, although we didn't know it at the time.

Lessons Learned:

Continuous integration is valuable, but outside the dev team it isn't obvious.

The biggest advantage to open-source tools is often acquisition time, not acquisition cost.

Functional Testing

- Functional testing was done manually
- We waited a year before staging a coup
  - we didn't want to encroach on their domain
- Demo of Selenium
  - demonstrated record-and-playback through the Selenium IDE
  - we recorded the first set of tests
  - then turned it back over to the test team
- They argued later that automated testing was ineffective
  - the automated script (singular) only worked one time
  - needed to be re-recorded when any changes got made to the app

Functional Testing

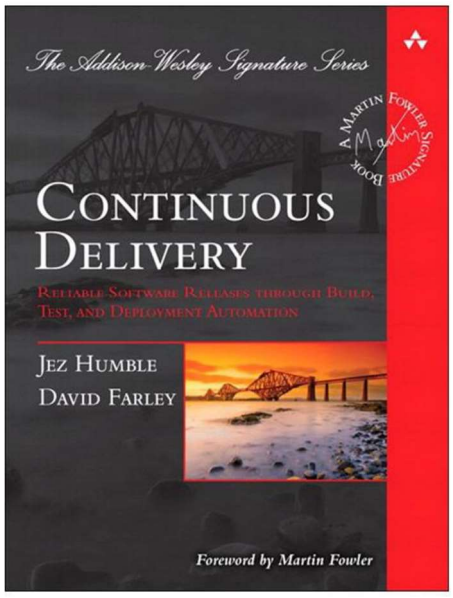
- Development team had more confidence in releases
- Also began testing user roles
  - Security testing = what can this type of user NOT do

Lesson Learned:

Should have focused on demonstrating that there were fewer escaped defects.

It was hard to point to a clear benefit.

The Book



- Project Manager came across the book in a book store
- Everything made so much sense
- Logical extension of what we were trying to do
- Addressed a lot of the issues we were running into
- No money or time for an effort, so we adopted it as our long-term goal

Automated Deploys

- Started with automating a greenfield web app install
  - Ignored the database install, since “it was easy”
- Then automated the manual COTS install
- Then started reverse engineering the broken COTS installer

Automated Deploys

- Down the road, realized we could automate everything
  - Doesn’t just reduce risk, also speeds up the process

Lesson Learned:

Automate everything- even the easy stuff.

When it is easier to install, you’ll stumble across more reasons to install it.

Go from Why? to Why not?

Security Testing

- Implemented OpenSCAP in Jenkins for STIG
  - immediately found issues
  - started adding Puppet manifests for remediation

- Started running weekly scans of dev and test using OpenVAS
  - no immediate issues, but started seeing package security updates before they became IAVMs

OpenVAS
Open Vulnerability Assessment System

OpenSCAP

Security Testing

Found 0 vulnerabilities!

Security Testing

- Lost a lot of faith in us when we were hacked
- Information Assurance isn't the same as Security
- You can't assume someone else will take care of it.
  - No matter what "it" is.

Lessons Learned:

Protect every system, everywhere.

Many hacks are just for the system, not the data.

The team can't have "their" problems.
They are all our problems.

Culture Clash

- Continuous Delivery was being openly discussed
  - PMO had just started thinking of it as a clear plan
  - Kept asking when "continuous delivery" would be delivered, and how it would be packaged
- Test and Integration started complaining
  - We were pushing them too hard
  - Moving too fast
- People on test and integration team started leaving
  - including "Burt"

Culture Clash

- Benefits were growing clear
- Effort was minimal
- No active resistance

Lesson Learned:

Do not underestimate cultural inertia.

Some will not or cannot ever make the mental shift.

DevOps is…

"The goal of DevOps is not just to increase the rate of change, but to successfully deploy features into production without causing chaos and disrupting other services, while quickly detecting and correcting incidents when they occur."

Gene Kim

Top 11 Things You Need to Know About DevOps

The Aftermath

- Releases every 2 weeks
  - Soft freeze Thursday for Friday release
  - Deploy Friday evening
  - 20-minute deploy, practiced dozens of times each sprint
  - 100% working functionality Friday evening
  - Non-event
- Dev and Test environments very closely matched Production
- Development
  - Unit tests
  - Continuous integration
  - Automated testing

U.S. Dept. of Homeland Security

The DHS Project

- On-going, March 2018 - present
- Agile development teams
- No automated deploys
- Limited continuous integration
- Working in a government data center and cloud

Continuous Integration

- Teams use Jenkins
  - Primary build is still the button in Visual Studio
- Moving to Jenkinsfile
- Migrating to new infrastructure
  - Exposing many broken, but still passing, builds

Non-Cloud Data Center

- No dynamic provisioning
- Provisioning takes time
  - Create a Visio diagram in a particular format
  - Submit for approval
  - Once approved, provisioning begins
  - Provisioning doesn't necessarily match request
  - Correcting take time, approvals, resubmissions
- From request to ready for use takes days to weeks

Access but not permission

- In our integrated test environment, the team
  - Submitted forms to request access
  - Planned to install software using Chef
  - Software was approved and installed in development
  - Coordinated with other teams to provision box
  - Announced on daily status call that we were going to install
  - Installed by us, with other contract watching
- Then
  - Got caught
  - Software was uninstalled
  - More forms submitted, including build instructions
  - Software was reinstalled by other contract, with us watching

That's the way we've always done it

- The most dangerous words for any transformation

Lesson Learned:

Culture is the hardest, and most important, trait to change for DevOps adoption.

#Coveros5

Adopt change incrementally, in processes and technology.

The team can't have "their" problems. They are all our problems.

Agile teams are better than non-agile teams, but are no substitute for an agile organization.

Culture is the hardest, and most important, trait to change for DevOps adoption.

"That's the way we've always done it" can be a clear danger sign.

Questions?

Join me on the TechWell Hub
http://starslack.techwell.com/
#devops