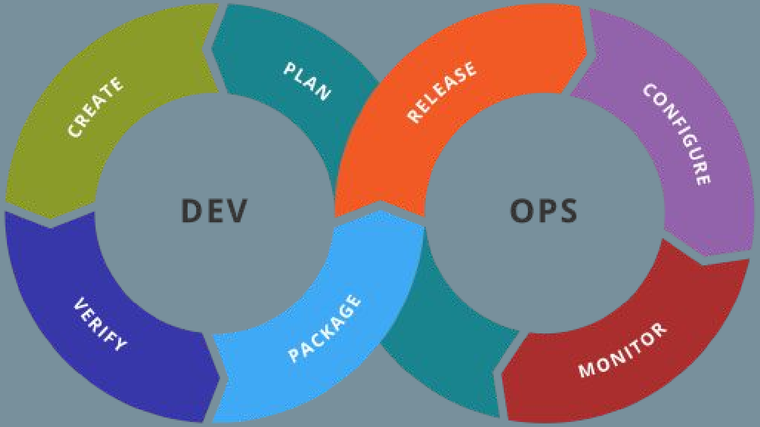
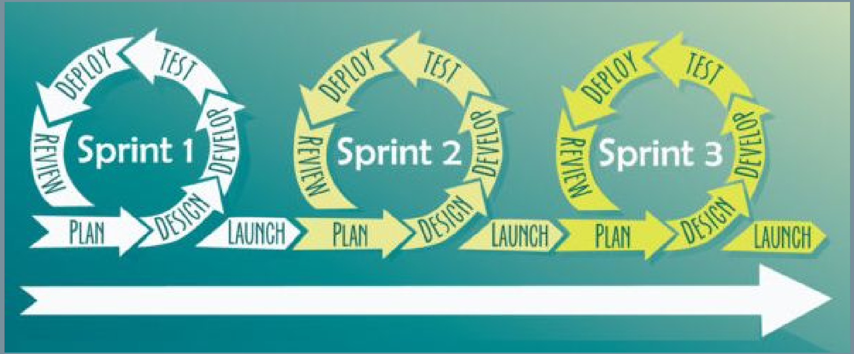
DevOps for Defense
DevOps and Agile + “The First Way”



…



Jan 4, 2018

DevOps for Defense Meetup: Code of Conduct

- UNCLASSIFIED ONLY!!!!
- Treat each other with respect and professionalism.
- Do not talk about private, sensitive, or proprietary work.
- Do talk about your experiences, needs, desires to improve work in our domain.
- Do share your thoughts.
- Do learn from others.
- Do respect & tip your bartenders!

History of DevOps

There is definitely a historical connection between DevOps and Agile. Starting all the way back in 2008 with Patrick Dubois presenting Agile Infrastructure and Operations at the Agile 2008 conference.

See The History of DevOps by Damon Edwards for more information

The Relationship Between DevOps and Agile

The DevOps philosophy is ingrained within the Agile Manifesto, and it could be said that DevOps is just an extension of Agile principles into operations:

- Individuals and interactions over processes and tools
    - DevOps is all about collaboration among everyone in the software supply chain
- Working software over comprehensive documentation
    - DevOps takes this concept to the next level by automating environment provisioning, software installation, and automatic delivery of applications instead of document-centric step-by-step instructions
- Customer collaboration over contract negotiation
    - When fixes to production issues can be automatically tested and redeployed, it’s certainly focusing on customer collaboration and value
- Responding to change over following a plan
    - Small batch sizes and continuous delivery support the ability to change as needed while delivering value

So why are they different?

It is typically how we tend to implement our Agile process

- Declaring victory too early
  - The concept of an “end game” or “hardening sprint” during which all necessary activities for documenting, packaging, deploying, and even finishing up testing activities can make agile look a lot more like a waterfall
  - DevOps strives to move the team toward a continuous delivery process that is capable of producing releasable code as frequently as possible. Ideally, the code produced each sprint is releasable, with many teams moving toward a truly continuous process wherein all code changes can be released immediately
- Critical team members are not interconnected
  - Unfortunately, not delivering releasable code frequently means that system administrators and operational personnel are not part of your cross-functional agile team. This causes communication challenges that agile was built to address!

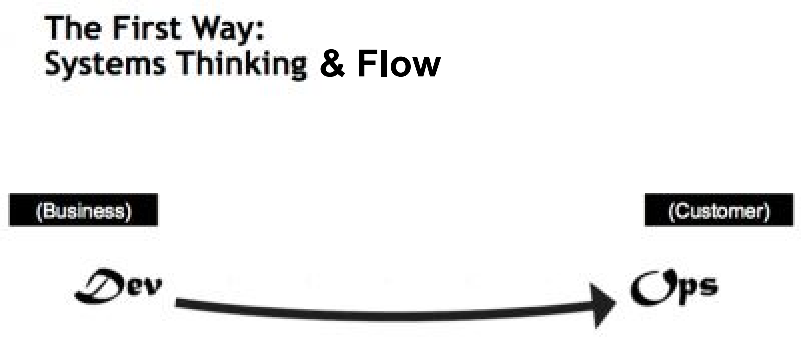
In Summary

Agile nor DevOps are business goals in and of themselves

- Both are cultural movements that can inspire your organization with better means for achieving your goals
  - Encourage a Learning Organization
  - Experimentation, Fail Fast, Limit the Blast Radius
- Agile and DevOps work better in combination
  - Small Batch Planning & Execution
  - Regular cadence that delivers deployable products
- It is about communication and involving anyone in the development of the product
  - Attack ambiguity and uncertainty - write good stories!

DevOps: The First Way - Systems Thinking & Flow

- Systems Thinking: “Emphasize the performance of the entire system, as opposed to the performance of a specific silo of work or department” - Gene Kim (2012)
  - Interpretation: Optimize the delivery of value to your customer across the organization recognizing that your delivery process is a system itself
- Flow: “Accelerate the delivery of work from Development to Operations to our customers” - Gene Kim (2016)
  - Interpretation: Build a Continuous Delivery pipeline! Feed it constantly!

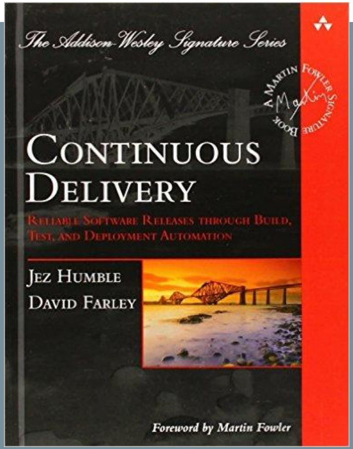

The First Way:
Systems Thinking & Flow

(Business)

(Customer)

Dev

Ops

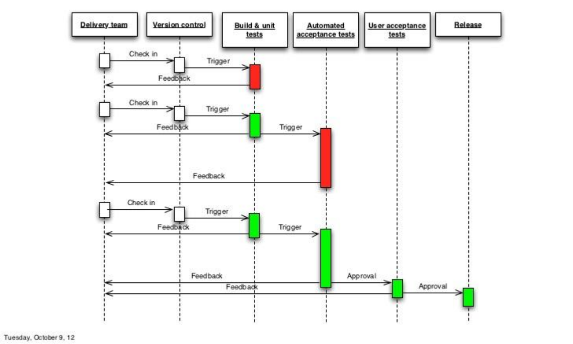

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

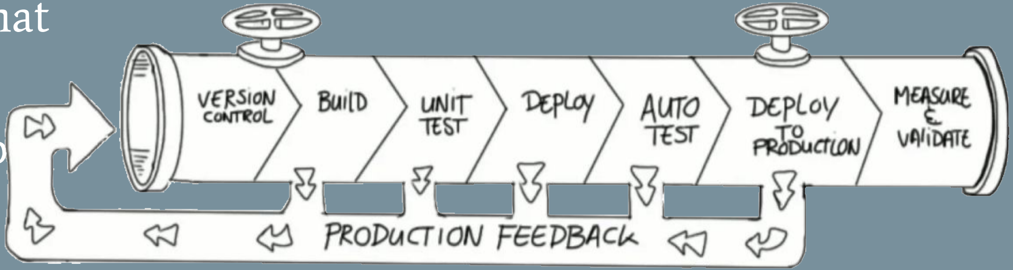
Jez

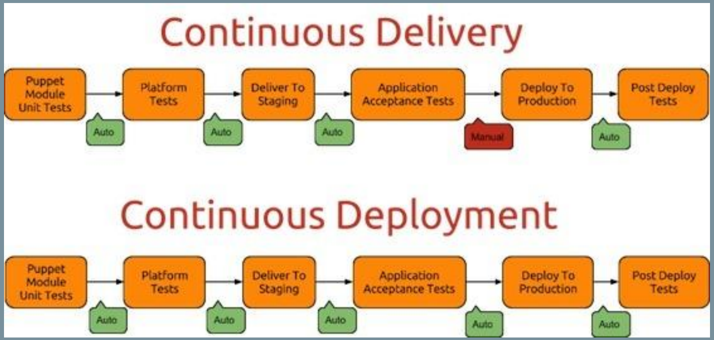
Dave

What is a Continuous Delivery Pipeline?

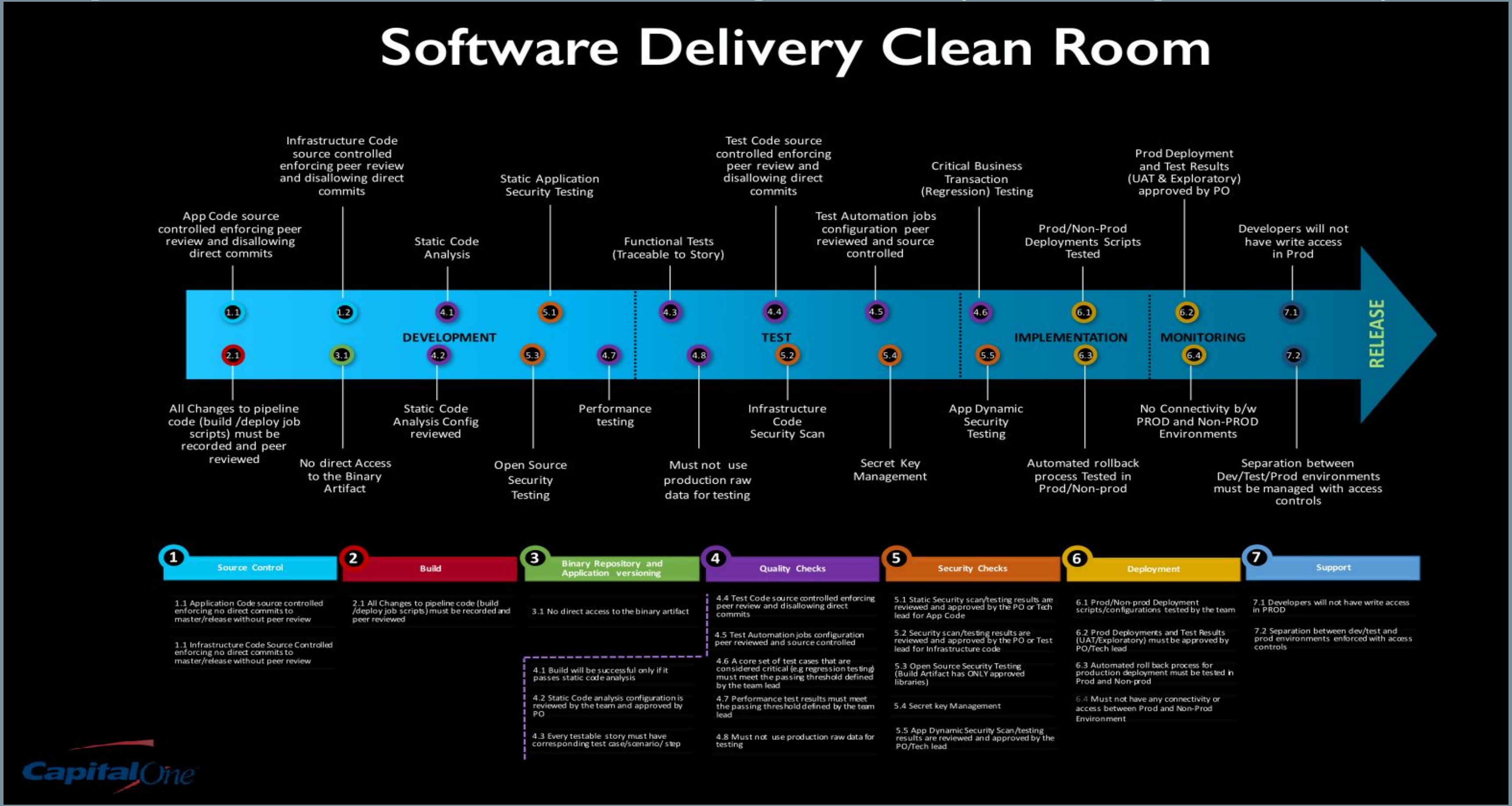
1. Automated suite of tools, triggered by an update to your product's configuration managed baseline, that performs all the build, test, QA, packaging, (and deployment) of yo system.
2. Rapid feedback to engineers of issues introduced by a change
3. Continuous Integration…on steroids!
4. Your product's immune system
5. Your source of confidence







Example Continuous Delivery Pipelines (#1 - Capital One)



Software Delivery Clean Room

App Code source controlled enforcing peer review and disallowing direct commits

Infrastructure Code source controlled enforcing peer review and disallowing direct commits

Static Code Analysis

Static Application Security Testing

Functional Tests (Traceable to Story)

Test Code source controlled enforcing peer review and disallowing direct commits

Test Automation jobs configuration peer reviewed and source controlled

Critical Business Transaction (Regression) Testing

Prod/Non-Prod Deployments Scripts Tested

Prod Deployment and Test Results (UAT & Exploratory) approved by PO

Developers will not have write access in Prod

All Changes to pipeline code (build /deploy job scripts) must be recorded and peer reviewed

No direct Access to the Binary Artifact

Static Code Analysis Config reviewed

Open Source Security Testing

Performance testing

Must not use production raw data for testing

Infrastructure Code Security Scan

Secret Key Management

App Dynamic Security Testing

Automated rollback process Tested in Prod/Non-prod

No Connectivity b/w PROD and Non-PROD Environments

Separation between Dev/Test/Prod environments must be managed with access controls

1 Source Control

1.1 Application Code source controlled enforcing no direct commits to master/release without peer review

1.1 Infrastructure Code Source Controlled enforcing no direct commits to master/release without peer review

2 Build

2.1 All Changes to pipeline code (build /deploy job scripts) must be recorded and peer reviewed

3 Binary Repository and Application versioning

3.1 No direct access to the binary artifact

4.1 Build will be successful only if it passes static code analysis

4.2 Static Code analysis configuration is reviewed by the team and approved by PO

4.3 Every testable story must have corresponding test case/scenario/ step

4 Quality Checks

4.4 Test Code source controlled enforcing peer review and disallowing direct commits

4.5 Test Automation jobs configuration peer reviewed and source controlled

4.6 A core set of test cases that are considered critical (e.g regression testing) must meet the passing threshold defined by the team lead

4.7 Performance test results must meet the passing threshold defined by the team lead

4.8 Must not use production raw data for testing

5 Security Checks

5.1 Static Security scan/testing results are reviewed and approved by the PO or Tech lead for App Code

5.2 Security scan/testing results are reviewed and approved by the PO or Test lead for Infrastructure code

5.3 Open Source Security Testing (Build Artifact has ONLY approved libraries)

5.4 Secret key Management

5.5 App Dynamic Security Scan/testing results are reviewed and approved by the PO/Tech lead

6 Deployment

6.1 Prod/Non-prod Deployment scripts/configurations tested by the team

6.2 Prod Deployments and Test Results (UAT/Exploratory) must be approved by PO/Tech lead

6.3 Automated roll back process for production deployment must be tested in Prod and Non-prod

6.4 Must not have any connectivity or access between Prod and Non-Prod Environment

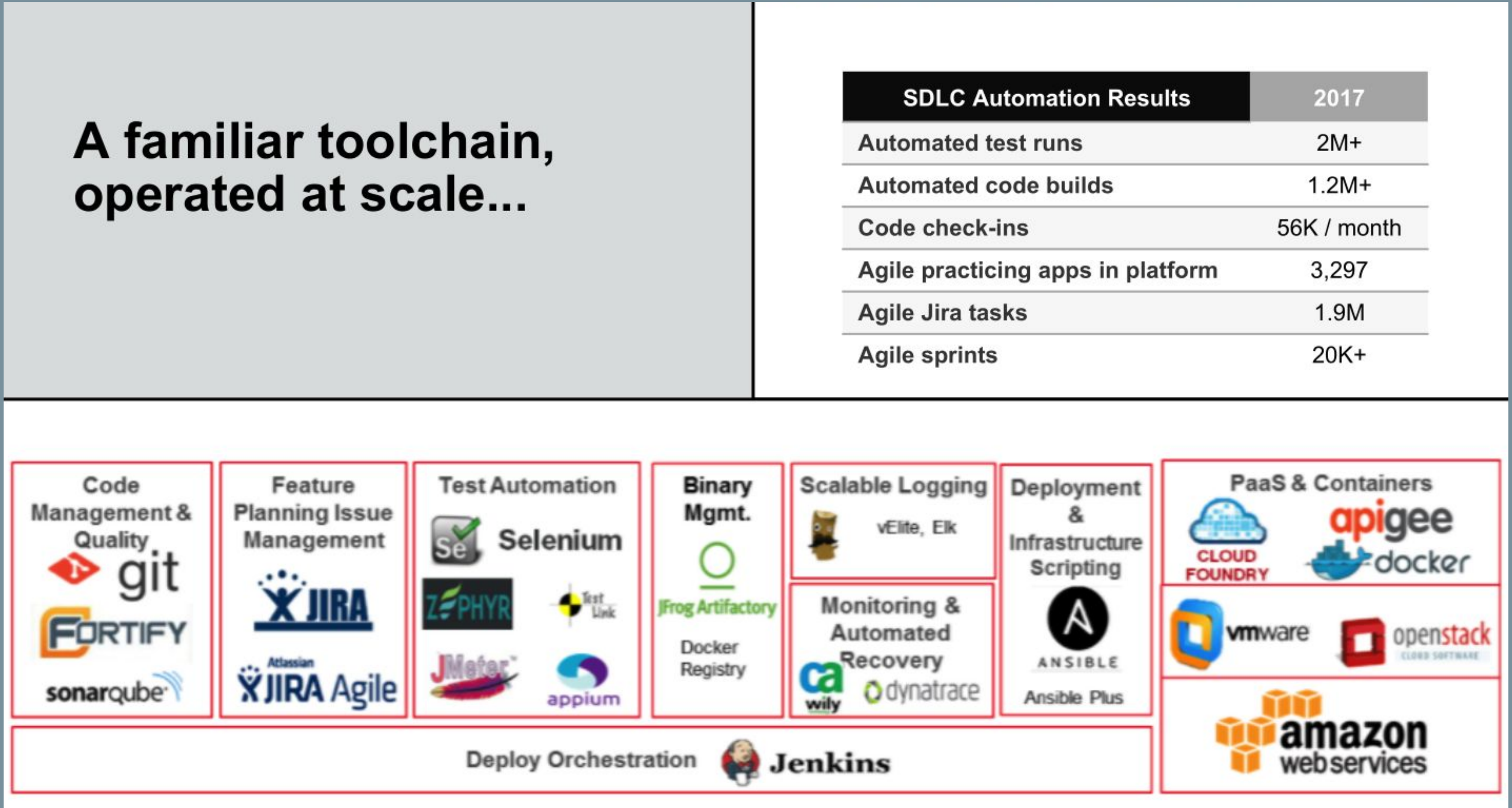
7 Support

7.1 Developers will not have write access in PROD

7.2 Separation between dev/test and prod environments enforced with access controls

CapitalOne

Example Continuous Delivery Pipelines (#2 - Verizon)



A familiar toolchain, operated at scale...

SDLC Automation Results	2017
Automated test runs	2M+
Automated code builds	1.2M+
Code check-ins	56K / month
Agile practicing apps in platform	3,297
Agile Jira tasks	1.9M
Agile sprints	20K+

Code Management & Quality: git, FORTIFY, sonarqube

Feature Planning Issue Management: JIRA, Atlassian JIRA Agile

Test Automation: Selenium, ZEPHYR, Test Link, JMeter, appium

Binary Mgmt.: JFrog Artifactory, Docker Registry

Scalable Logging: vElite, Elk

Monitoring & Automated Recovery: ca wily, dynatrace

Deployment & Infrastructure Scripting: ANSIBLE, Ansible Plus

PaaS & Containers: CLOUD FOUNDRY, apigee, docker, vmware, openstack CLOUD SOFTWARE, amazon web services

Deploy Orchestration: Jenkins

“Here at ________, we build _______ into our products!”

- Use your pipeline to build in quality, security, and the “ilities”.
- Quality:
  - Automated Unit Test - xUnit
  - Automated Acceptance Tests - Cucumber
  - Static / Dynamic Analysis - (sooo many)
  - Metrics (Sonar Qube, Jenkins, etc.)
- Security:
  - Klocwork, Fortify, Coverity, Arachni, etc.
- Deployability:
  - Puppet, Chef, Ansible
- Reliability:
  - Simian Army (Chaos Monkey, etc.)
- ______ility
  - Find/Build the right tool, Read Jez’s Book

Group Exercise

1. Divide up into 2 groups
2. Assign roles
   a. Production unit 1
   b. Production unit 2
   c. Production unit 3
   d. Release QA
3. Each role has specific instructions, do not share instructions outside your role
4. Build and release 15 products
   a. Group 1: Batch size = 15
   b. Group 2: Batch size = 1

Thank You!

Stick around and chat. Meet somebody new.

Suggest a topic or exercise for next time!

Volunteer to present a subject!!!

DevOps for Defense Meetup - Founding Members!