Test Automation in DevOps

MOVING TOWARDS CONTINUOUS TESTING
About Me…

• CTO of Utopia Solutions – QA/testing focused service provider
• Involved in software quality and testing for over 25 years
• Most of that time focused on test automation
• Passionate about learning and helping organizations benefit from doing automation the right way

Email: lee.barnes@utopiasolutions.com
Twitter: @USI_LeeBarnes
LinkedIn: linkedin.com/in/leebarnes
Blog: utopiasolutions.com/blog1
Agenda

• Intro
  – What is DevOps
  – Testing in DevOps
  – Continuous Testing
  – Test Automation in DevOps

• Effective Test Automation
  – Scope and Approach
  – Test Environment and Data Management

• Testing in Production
What is DevOps?
DevOps

Software engineering culture and practice aimed at unifying software development (Dev) and software operation (Ops)

- Shorter development cycles
- Increased deployment frequency
- Higher quality releases
- Close alignment with business objectives

The DevOps Cycle

https://danashby.co.uk/2016/10/19/continuous-testing-in-devops/

- Dan Ashby
A Delivery Pipeline

- Developer Workstation
- Integration Environment
  - Performance / Load Environment
  - Deploy
- Production
  - Deploy
### DevOps Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Integration</td>
<td>Frequent integration of developers’ code into a shared repository</td>
</tr>
<tr>
<td>Continuous Delivery</td>
<td>Develop, build, test and release software in short cycles</td>
</tr>
<tr>
<td>Continuous Deployment</td>
<td>Automated deployment of qualified application features to production</td>
</tr>
<tr>
<td>Continuous Testing</td>
<td><em>Stay tuned!</em></td>
</tr>
</tbody>
</table>

**Continuous Processes ≠ DevOps**
# State of DevOps

<table>
<thead>
<tr>
<th>Aspect of Delivery Performance</th>
<th>Elite</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Frequency</td>
<td>On-demand (multiple deploys per day)</td>
<td>Once per hour to once per day</td>
<td>Once per week to once per month</td>
<td>Once per week to once per month</td>
</tr>
<tr>
<td>Lead Time for Changes (commit to production)</td>
<td>Less than one hour</td>
<td>One day to one week</td>
<td>One week to one month</td>
<td>One month to six months</td>
</tr>
<tr>
<td>Time to Restore Service (outage or impairment)</td>
<td>Less than one hour</td>
<td>Less than one day</td>
<td>Less than one day</td>
<td>One week to one month</td>
</tr>
<tr>
<td>Change Failure Rate (%)</td>
<td>0 to 15%</td>
<td>0 to 15%</td>
<td>0 to 15%</td>
<td>46 to 60%</td>
</tr>
</tbody>
</table>

2018 Accelerate State of DevOps Report
Testing in DevOps
Traditional Testing in DevOps?

Testing is PART of the dev cycle
Testing in DevOps

Where does testing fit?

[Diagram showing the DevOps lifecycle with arrows pointing to various stages: Plan, Branch, Code, Merge, Build, Ops, Release, Deploy, Operate, Monitor.]
Testing in DevOps

Testing spreads THROUGHOUT the development cycle

... and becomes everyone’s responsibility

- Dan Ashby

https://danashby.co.uk/2016/10/19/continuous-testing-in-devops/
Testing Across the Delivery Pipeline

- Code analysis
- Unit tests
- Integration tests

- End to end tests
- Exploratory tests

- Performance tests
- Security tests

- Deploy

- Integration Environment

- Deploy

- Production Environment

- Production tests
DevOps testing activities must provide:

- Fast feedback
- Accurate feedback
- Actionable feedback
Continuous Testing
Continuous Testing

Running the right tests at the right time to assess the risk of promoting a build through the delivery pipeline
Successful Continuous Testing

• Provides fast, accurate and actionable feedback
• Avoids creating a bottleneck in the delivery pipeline
• Is embedded throughout the development process (shift Left AND shift Right)
• Includes effective test automation practices
• Promotes effective manual testing activities
• Is executed in stable test environments with valid test data
• Continually reviews and refines test suites to ensure relevance and efficiency
Continuous Testing vs. Test Automation

Test Automation
Pass/fail results associated with user stories and business requirements

Continuous Testing
Business risk associated with releasing the software

Primary Differences

Risk
Tests built with business risk in mind

Breadth
Tests that detect issues that impact user experience

Time
Tests that provide immediate feedback and are executed as close as possible to app element they are testing

- Adapted from Continuous Testing vs Test Automation by Wayne Ariola
Test Automation in DevOps
Importance of Effective Automation

Continuous Integration requires three capabilities:

• A comprehensive and reliable set of automated tests that validate we are in a deployable state
• A culture that “stops the entire production line” when our validation tests fail
• Developers working in small batches on trunk rather than long-lived feature branches

The DevOps Handbook (Kim, Humble, Debois & Willis)
## Automation Impact on DevOps Performance

<table>
<thead>
<tr>
<th>Type of Manual Activity</th>
<th>Elite</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>5%</td>
<td>10%</td>
<td>30%</td>
<td>30%</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td>10%</td>
<td>20%</td>
<td>50%</td>
<td>30%</td>
</tr>
<tr>
<td>Deployments</td>
<td>5%</td>
<td>10%</td>
<td>30%</td>
<td>30%</td>
</tr>
<tr>
<td>Change Approvals</td>
<td>10%</td>
<td>30%</td>
<td>75%</td>
<td>40%</td>
</tr>
</tbody>
</table>

- 2018 Accelerate State of DevOps Report
Current State of Test Automation

Some alarming statistics

- The level of test automation is below 20%
  - 2017-2018 World Quality Report

- The value generated by test automation is largely unchanged
  - 2017-2018 World Quality Report

- Almost 80% of test automation implementations fail
  - Unknown
Obstacles to Test Automation Success

Common Automation Traps

• Too much / not enough automation
• Automating the wrong tests
• Focusing on UI tests
• Unstable test environments and data
• Not treating automation as code
Effective Automation
Pillars of Effective Automation

Scope

Approach / Method

Environment & Data
SCOPE AND APPROACH
Pipeline Tradeoffs

Increasing Confidence in Build’s Production Readiness

Environments Become More Production-like

Developer Workstation → Deploy
Integration Environment → Deploy
Performance / Load Environment → Deploy
Production

Faster Feedback

- Continuous Delivery by Jez Humble and David Farley
# Which Test Suites Do We Need?

**Test Suite Canvas for evaluating test suites**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Dependencies</th>
<th>Constraints</th>
<th>Pipeline / Execution</th>
<th>Environment / Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>What business risk does this test suite mitigate?</td>
<td>What systems or tools must be in place for this test suite to run successfully?</td>
<td>What can prevent us from implementing this test suite in an ideal way?</td>
<td>Will this test suite be part of a pipeline? When will it be triggered? How often will it run?</td>
<td>What environment will the test suite run in? How will test data be managed?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ownership and Response</th>
<th>Maintainability</th>
<th>Effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who will create the test suite?</td>
<td>What will the process be to review code?</td>
<td>How will we know the test suite is effective?</td>
</tr>
<tr>
<td>Who should own it?</td>
<td>What documentation needs to exist?</td>
<td></td>
</tr>
<tr>
<td>Who will address test failures and how?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Ashley Hunsberger (adapted from original concept by Katrina Clokie)
Test Suites Mapped to the Pipeline

<table>
<thead>
<tr>
<th>Suite</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Tests</td>
<td>Is the change being pushed ready for a code review?</td>
</tr>
<tr>
<td>Integration Tests</td>
<td>Are the integrated system components ready for further testing?</td>
</tr>
<tr>
<td>End-to-End Tests</td>
<td>Is the system functionally ready for deployment?</td>
</tr>
<tr>
<td>Performance</td>
<td>Does the system meet performance SLAs?</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Test Scope: Risk vs. Speed

Coverage vs. Speed

Feedback Loop
Which Tests Do We Need?

Using **FITR** to assess your automated tests for inclusion

- **Focused** Test is tied as close to the functionality being tested as possible ▶ UI, API, unit, etc.
- **Informative** Test clearly and concisely communicates its intent and result
- **Trustworthy** Test executes reliably and doesn’t provide false negatives / positives
- **Repeatable** Test can be executed on demand ▶ environment and data dependencies

- Bas Dijkstra

"All models are wrong, but some are useful"

George Box, Statistician
Test Pyramid... Re-imagined

- Noah Sussman

- Katrina Clokie
Or Not a Pyramid at All...

- Gives equal importance to each layer
- Removes the focus on solely automation
- Highlights that tools and skills must be used appropriately throughout

- John Stevenson
Continuous Testing Improvement

- If an automated acceptance test fails -> investigate writing a unit test to catch the problem
- If a defect is discovered in exploratory testing -> write an automated unit or acceptance test as appropriate
- If an automated test generates a false positive error -> fix the test or remove it

⚠️ If tests are unreliable, they will lose their privilege of being in the build pipeline and eventually decay into uselessness
ENVIRONMENT AND DATA
Test environment and data availability continue to be top challenges to achieving desired levels of automation

- 2018-19 World Quality Report

Well... it worked on my machine!
Test Environment and Data Spectrum

Environment / Data Issues
- Data staleness and instability
- Scheduling conflicts
- Not production-like

Automation Averse
- Unreliable execution
- Long/complex setup
- False positives
- Prolonged analysis

Automation Friendly
- Create clean environments on demand
- Dedicate environments to single purpose
- Production-like (if needed)
- Load with clean/fresh data

Static
- Shared
- Stale

Dynamic
- Dedicated
- Fresh
DevOps Environment Technologies

Evolved Platforms

Virtual Machines

Containers

Cloud

Infrastructure as Code

Application 1
Application 2
Application 3

Bins/Lib
Guest OS
Hypervisor
Infrastructure

App 1
App 2
App 3

Bins/Lib
Guest OS
Container Engine
Operating System
Infrastructure

Cloud

Ansible

puppet

CHEF
Benefits to DevOps Testing / Automation

Cloud / Virtual Machines / Containers + Infrastructure as code

• Environment configurations can be version controlled
• Environment state can be included with test results
• Manual process becomes automated ➤ repeatable, scalable and reliable
• Ability to develop and test in more production-like environments
• Environment creation can become part of the delivery pipeline if desired
• Infrastructure management processes/code can be tested via automation
• Destructive testing
Test Data Challenges

Three of the top test data challenges according to the World Quality Report 2017-18

- Keeping test data in sync with tests
- Lack of test data for complex integration testing
- Maintaining consistent test data across systems under test

Inability to reliably execute automated tests on demand
Test Data Management Strategy

Key Strategy Input Questions

What data do your tests require?

What are the dimensions of the required data?

What are the data sources?

- **Type**: is data structured or unstructured?
- **Value**: what specific values or class of data are required?
- **Time**: does data change over time?
- **Reuse**: Can data be used more than once?
- **Volume**: how much do data you need?
Common Data Management Techniques

- **System / application techniques**
  - Data refresh
  - Data partitioning
  - Dedicated automation environments

- **Test suite techniques**
  - Pre-test / test suite data creation
  - Post-test / test suite data cleanup
  - On-demand environment creation / destruction
  - Dynamic data handling

- **External techniques**
  - Commercial test data management solutions
Dynamic Data Handling Techniques

Handle application/system data that changes periodically

Static metadata tags used in test cases
Login with <ADMIN_CREDENTIALS>

Metadata tags mapped to current data values
ADMIN_CREDENTIALS = Administrator / $Yyfht%$as

Test suite replaces tags with current data at runtime

Centralized test data repository
Dynamic Data Handling Techniques

Handle test input data that must be dynamic or unique

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstName</td>
<td>&lt;AUTO_FirstName&gt;</td>
</tr>
<tr>
<td>LastName</td>
<td>&lt;AUTO_LastName&gt;</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>&lt;AUTO_DOB</td>
</tr>
<tr>
<td>Email</td>
<td>&lt;AUTO_Email</td>
</tr>
</tbody>
</table>

Test suite replaces parameter tags at runtime

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstName</td>
<td>John</td>
</tr>
<tr>
<td>LastName</td>
<td>Smith</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>06-29-1977</td>
</tr>
<tr>
<td>Email</td>
<td><a href="mailto:john.smith@mailinator.com">john.smith@mailinator.com</a></td>
</tr>
</tbody>
</table>
Service Virtualization

Supports stable execution for tests that require access to application dependencies including:

- Dependencies that have not yet been developed
- Dependencies that don’t have stable/appropriate data
- 3rd party dependencies that don’t allow testing
- Any other situation where the dependency can’t be interacted with in a stable and predictable manner
Testing in Production
Testing vs. Monitoring in Production

Known Knowns  /  Known Unknowns  /  Unknown Unknowns

Testing

- Production users
- Test cases and monitors

Production System

Monitoring

- Telemetry data
- Analysis
- Quality Assessment
Key Practices

Production Testing
• Not a replacement for upstream testing
• Tests should evolve from heartbeats to user impacting scenarios
• Systems should “know” they are being tested

Production Monitoring
• Development needs to be aware of monitoring info
• Monitoring and alerting should be tested

• Production testing and monitoring strategy requires broad IT input
• Feedback (analytics/logging) should be clear and concise
• Testability of the system should be continuously reviewed
Key Takeaways

• DevOps is a culture, not just processes and tools
• Effective test automation is required to achieve Continuous Testing in DevOps
• Focus on the three pillars to ensure fast, reliable and accurate feedback
  – Scope
  – Approach
  – Environment and Data Management
Questions...

?? .... and Answers!

Direct future questions to:

Lee Barnes
CTO
Utopia Solutions, Inc.

Email: lee.barnes@utopiasolutions.com
Twitter: @USI_LeeBarnes
LinkedIn: linkedin.com/in/leebarnes
Blog: utopiasolutions.com/blog1
References


• Ashby, D. (2016). *Continuous Testing in DevOps*  
  https://danashby.co.uk/2016/10/19/continuous-testing-in-devops/

• Clokie, K. (2017). *Test Automation Canvas*  


• Eliot, S. *The Future of Software Testing Part Two*  

• Humble, J., Farley, D. (2010) *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*

  https://www.youtube.com/watch?v=RBrAj9jKgX0


• Stevenson, J. (2015). Blog Post based on MEWT4 Session titled *Sigh, It’s That Pyramid Again* by Richard Bradshaw  
  http://steveo1967.blogspot.com/2015/10/mewt4-post-1-sigh-its-that-pyramid.html

• Wikipedia – DevOps  

• World Quality Report 2017-18

• World Quality Report 2018-19