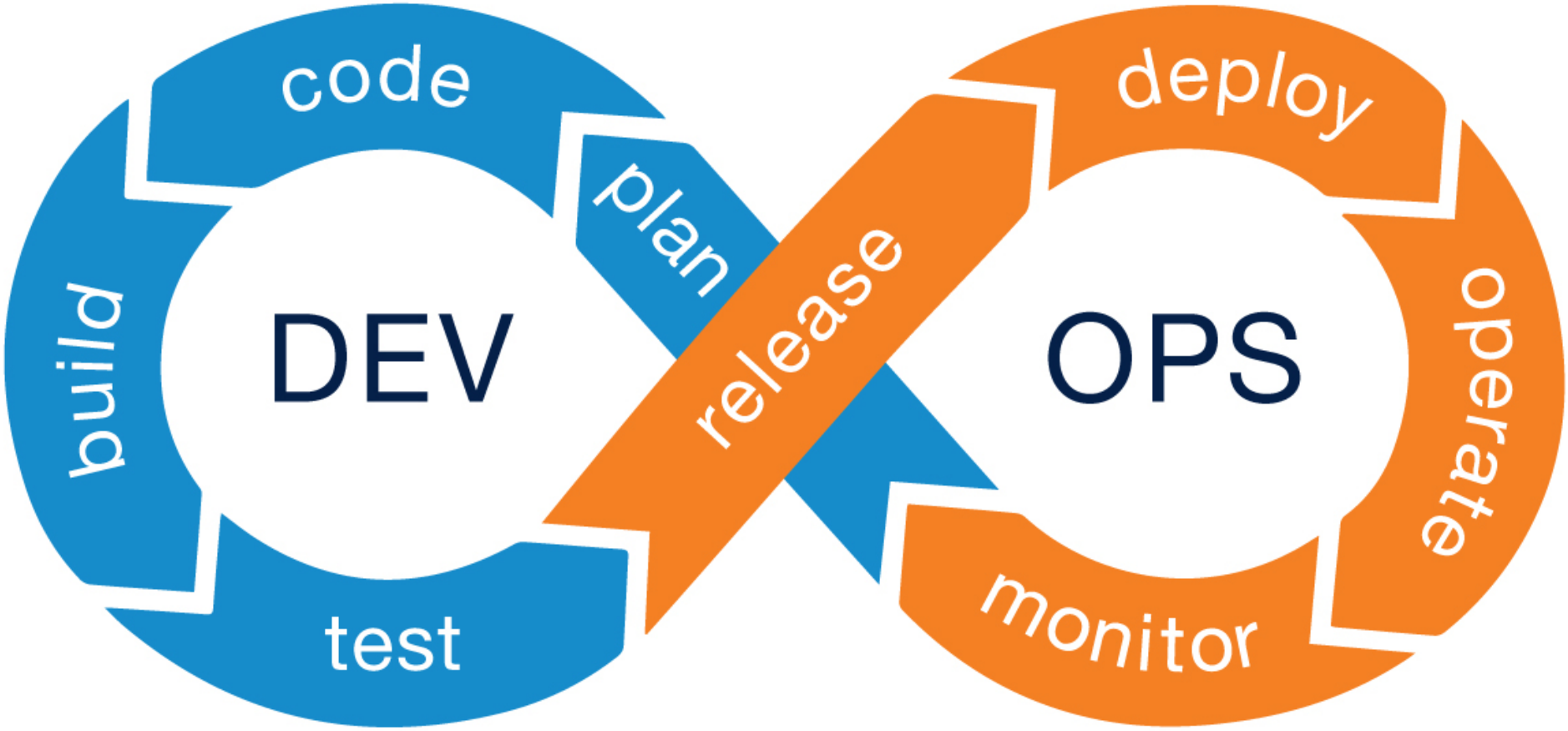
SWEN438 - DevOps

Lecture 15 — Continuous Delivery

James Quilty

School of Engineering and Computer Science
Victoria University of Wellington

The DevOps Lifecycle



Continuous Delivery Part I: Foundations

“It is vital to minimize cycle time so that an effective feedback loop can be established.”

– Jez Humble and David Farley, *Continuous Delivery*

The Problem of Delivering Software

Let's accept, for the moment, that delivering software poses problems.

- What are the issues?
- What problems do they cause?
- What might be the solutions?

Good Configuration Management Checklist

- Can I exactly reproduce any of my environments, including their configuration?
- Can I easily make an incremental change and deploy to any environment?
- Can I easily see and trace each change that occurred?
- Can I satisfy all of the compliance regulations that I am subject to?
- Is it easy for every member of the team to read and write changes?

Good Configuration Management Checklist

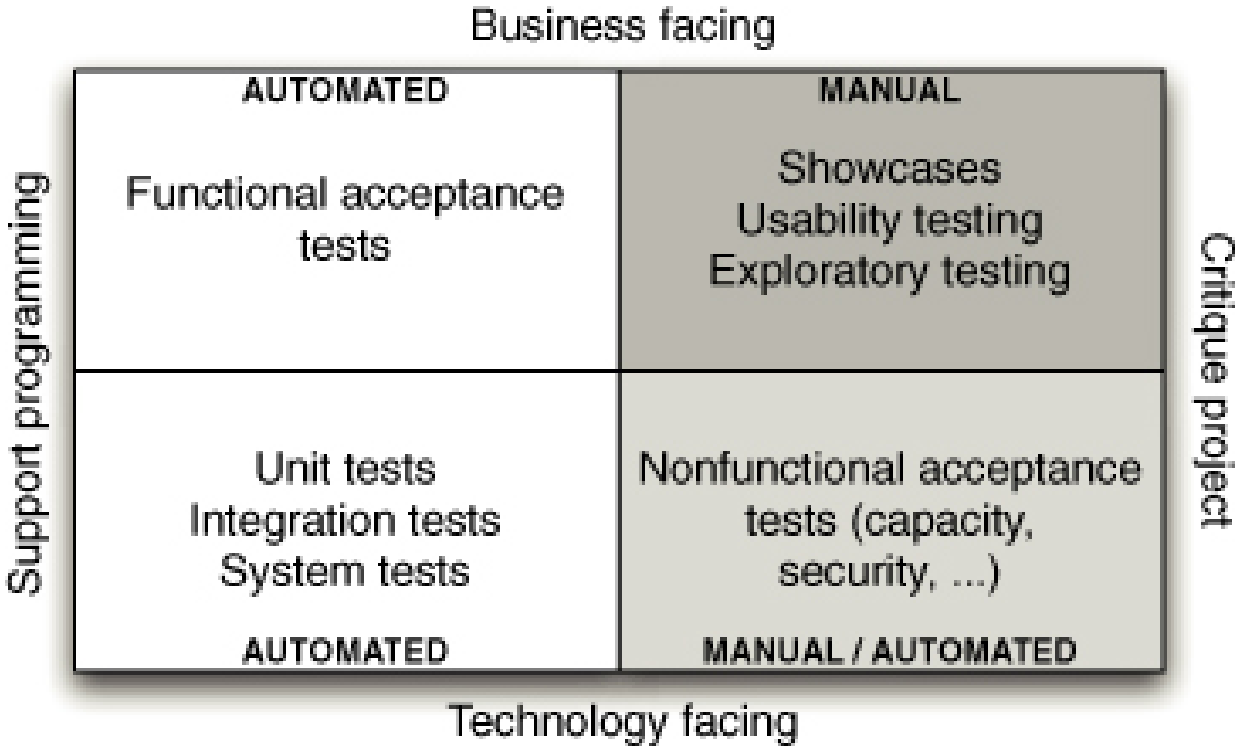
If your configuration management process is sound, you should be able to answer “yes” to the following questions:

- Could you completely re-create your production system, excluding production data, from scratch from the version-controlled assets that you store?
- Could you regress to an earlier, known good state of your application?
- Can you be sure that each deployed environment in production, in staging, and in test is set up in precisely the same way?

Continuous Integration Essential Practices

1. Don’t Check In on a Broken Build
2. Wait for Commit Tests to Pass before Moving On
3. Always Run All Commit Tests Locally before Committing
4. Wait for Commit Tests to Pass before Moving On
5. Never Go Home on a Broken Build
6. Always Be Prepared to Revert to the Previous Revision
7. Time-Box Fixing before Reverting
8. Don’t Comment Out Failing Tests
9. Take Responsibility for All Breakages That Result from Your Changes
10. Test-Driven Development

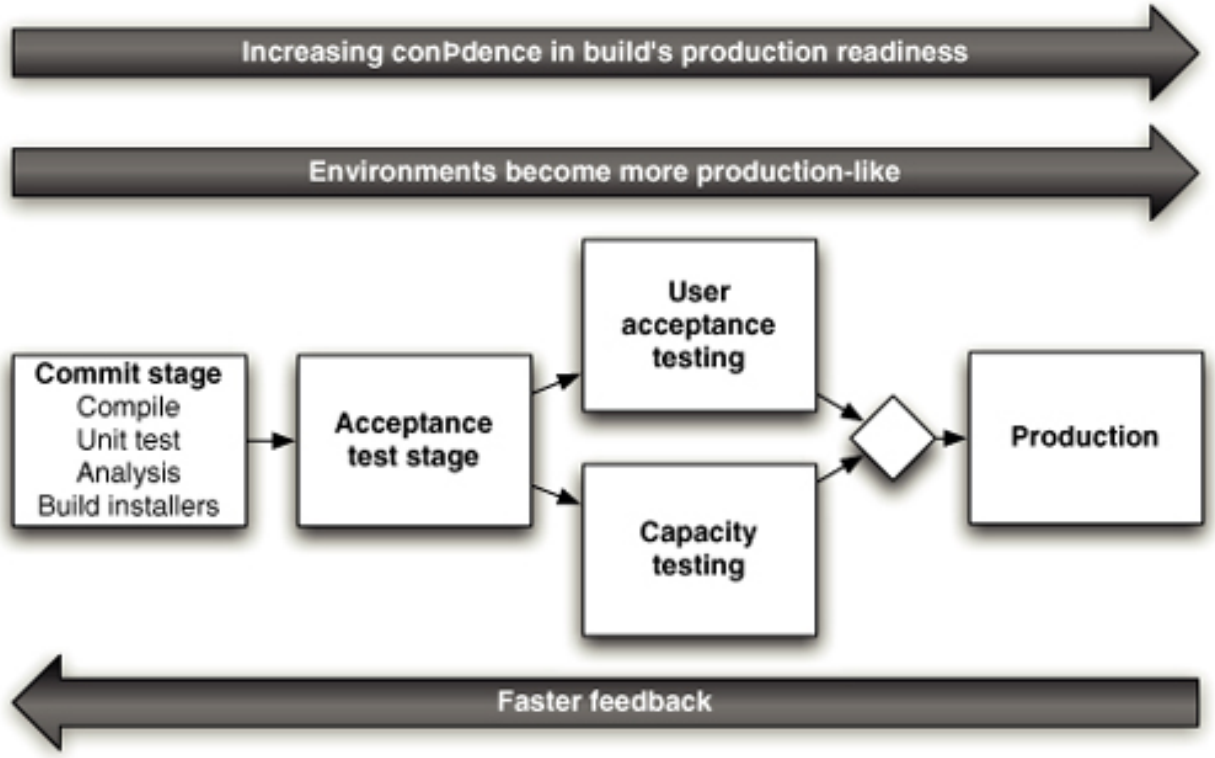
Implementing a Testing Strategy



"Testing is primarily concerned with establishing feedback loops that drive development."

Continuous Delivery Part II: The Deployment Pipeline

CI mainly focuses on development teams, CD focuses on the business and the user.



Deployment Pipeline Practices

- Only Build Your Binaries Once
- Deploy the Same Way to Every Environment
- Smoke-Test Your Deployments
- Deploy into a Copy of Production
- If Any Part of the Pipeline Fails, Stop the Line

Preparing to Release

- Automating Deployment and Release
- Backing Out Changes
- Building on Success

“On no account should you have a different process for backing out than you do for deploying, or perform incremental deployments or rollbacks. These processes will be rarely tested and therefore unreliable. They will also not start from a known-good baseline, and therefore will be brittle. Always roll back either by keeping an old version of the application deployed or by completely redeploying a previous known-good version.”

Principles and Practices of Build and Deployment Scripting

1. Create a Script for Each Stage in Your Deployment Pipeline
2. Use an Appropriate Technology to Deploy Your Application
3. Use the Same Scripts to Deploy to Every Environment
4. Use Your Operating System's Packaging Tools
5. Ensure the Deployment Process Is Idempotent
6. Evolve Your Deployment System Incrementally

Deployment Scripting Tips and Tricks

- Always Use Relative Paths
- Eliminate Manual Steps
- Build In Traceability from Binaries to Version Control
- Don’t Check Binaries into Version Control as Part of Your Build
- Test Targets Should Not Fail the Build
- Constrain Your Application with Integrated Smoke Tests

Commit Stage Principles and Practices

pre-commit!

1. Provide Fast, Useful Feedback
2. What Should Break the Commit Stage?
3. Tend the Commit Stage Carefully
4. Give Developers Ownership

Automated Acceptance Testing

- Increase confidence that the software is fit for purpose
- Provide protection against large-scale changes to the system
- Significantly improve quality through comprehensive automated regression testing
- Provide fast and reliable feedback whenever a defect occurs so that it can be fixed immediately
- Free testers to devise testing strategies, develop executable specifications, and perform exploratory and usability testing
- Reducing cycle time and enabling continuous deployment