SWEN438 - DevOps

Lecture 16 — Continuous Delivery

James Quilty

School of Engineering and Computer Science
Victoria University of Wellington

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

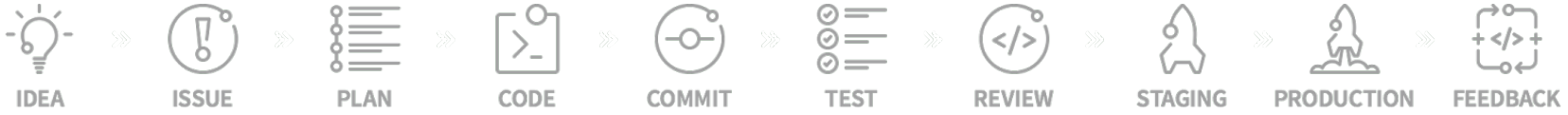
- Increase confidence that the software is fit for purpose
- Provide protection against large-scale changes to the system
- Significantly improve quality through comprehensive automated regression testing
- Provide fast and reliable feedback whenever a defect occurs so that it can be fixed immediately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

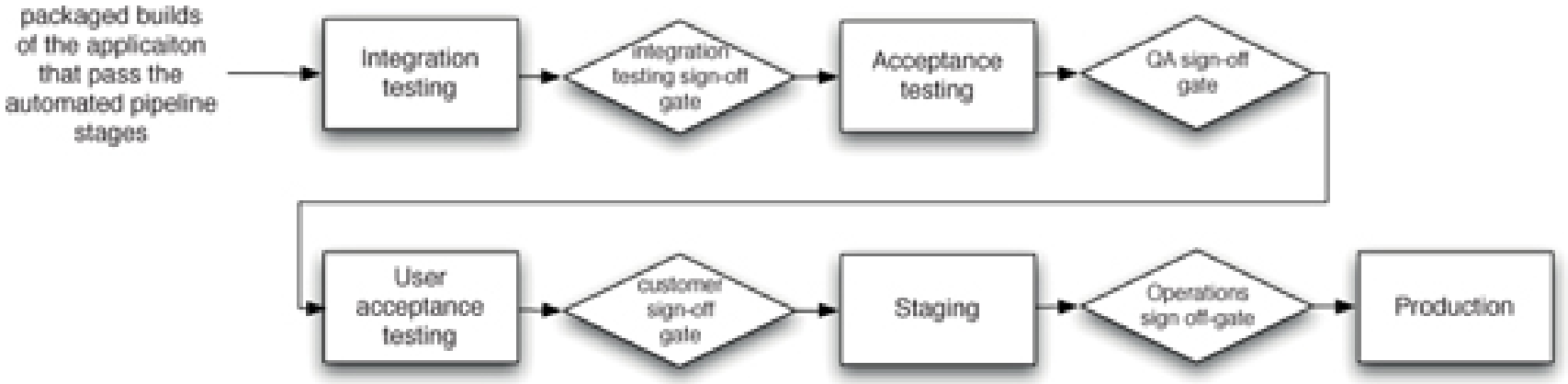
Deploying and Releasing Applications

Strategic considerations

- deployment responsibilities for each environment
- configuration management strategy
- agreed technology for deployment (Dev + Ops)
- deployment pipeline implementation plan
- test environment list and build progression process
- disaster recovery plan
- *etc.*

This is looking very... *planned*.

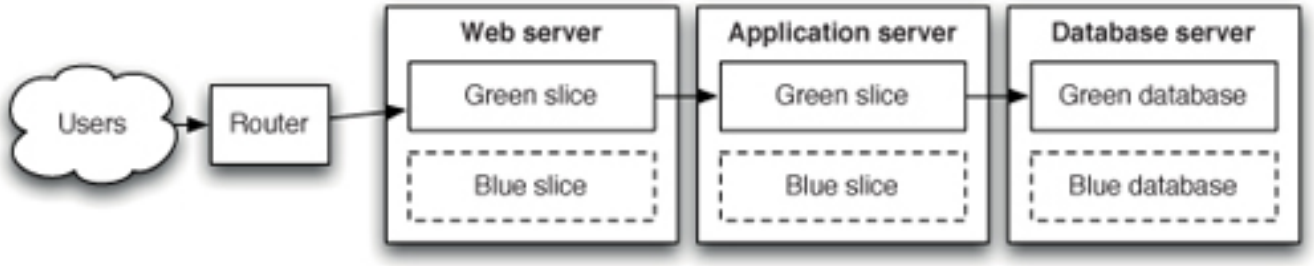

IDEA
ISSUE
PLAN
CODE
COMMIT
TEST
REVIEW
STAGING
PRODUCTION
FEEDBACK


packaged builds of the applicaiton that pass the automated pipeline stages
Integration testing
Integration testing sign-off gate
Acceptance testing
QA sign-off gate
User acceptance testing
customer sign-off gate
Staging
Operations sign off-gate
Production

Rolling Back Deployments and Zero-Downtime Releases

Techniques:

- Hot deployment and redeployment of the previous good version
- Canary Releases
- Blue-Green deployments

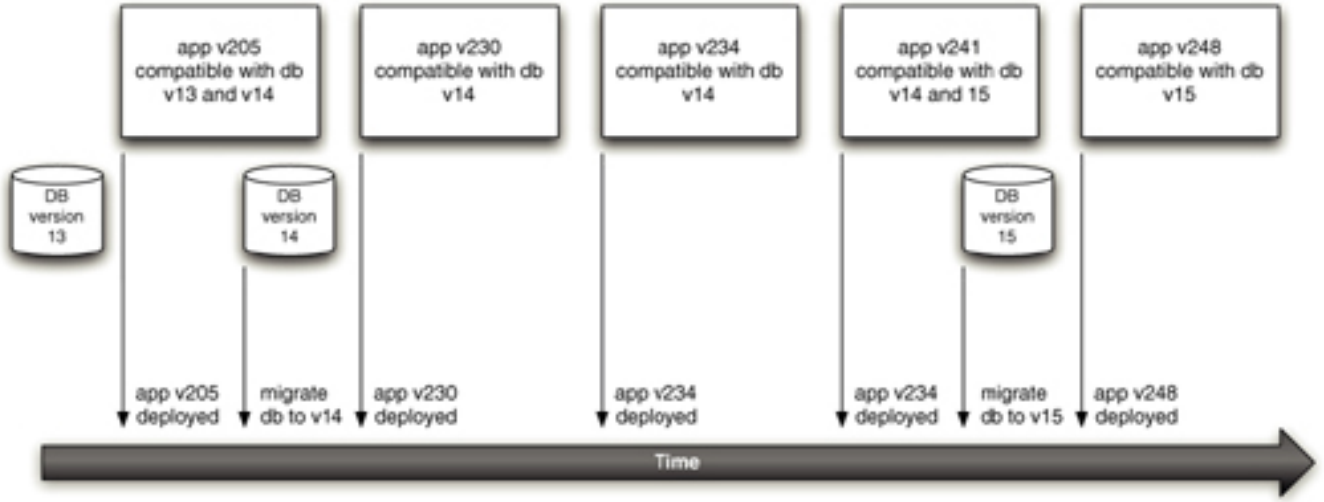


But what about user-installed software?

Managing Data

Data is difficult!

- Database scripting is a must-have!
- Databases should be versioned (but not version-controlled)
- Test data in the deployment pipeline is special data!
- Rolling-back databases and Zero-Downtime releases
- Decouple application deployment from database migration



Components and Dependencies

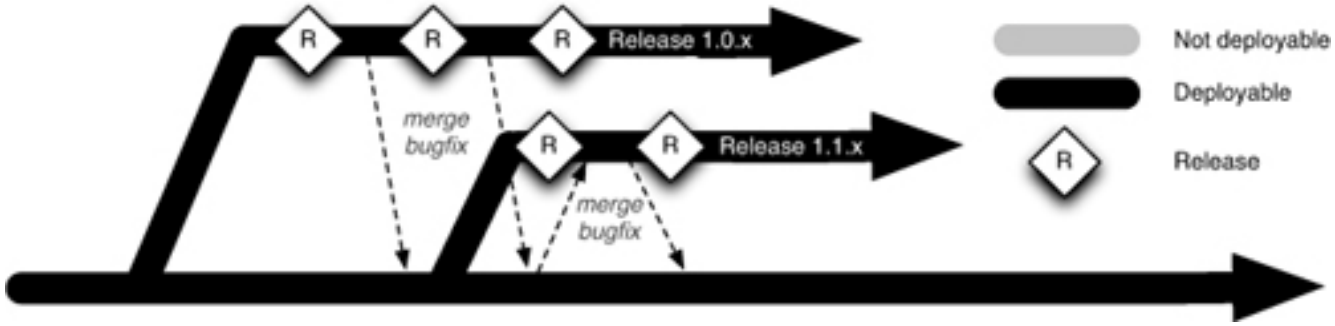
Applications typically grow. . . into *monoliths*; contemporary response: componentisation!

- Use feature flags to maintain releasability during change
- Make changes incrementally
- Manage libraries. . . carefully!
- Component pipelines and artefact repositories are must-haves!

Advanced Version Control

Advanced Version Control — a historic view.

- Advice published 10 years ago
- Advocates *always* committing to main
- With the benefit of hindsight, what do we think about this?



Managing Continuous Delivery

Common delivery problems — their symptoms and causes

- Infrequent or Buggy Deployments
- Poor Application Quality
- Poorly Managed Continuous Integration Process
- Poor Configuration Management

Conclusion

The last part of CALMRS is *Sharing* and the focus has been on the practice of *Continuous Delivery* as an embodiment. Common themes:

- Feedback
- Transparency
- Collaboration
- Retrospectives

Final thought: *defund the CAB!*