

# The Why and How of Continuous Delivery

Nigel McNie

[getyourgameon.co.nz](http://getyourgameon.co.nz)

Continuous Delivery is a **strategy** for releasing software

# The Development Landscape

- Business Demands
- Team instability
- Near-realtime dev requirements
- Technical Debt

# The Deployment Landscape

- Waterfall
- Agile
- Business As Usual (BAU)
- Hotfixing/Firefighting
- Marketing Release
- 5am Release
- Friday Afternoon Release
- Scheduled Downtime
- Unexpected Downtime
- Merge Hell
- String Freeze
- QA Approval
- Management signoff

# What's wrong with them?

- Even nightly releases are not fast enough
  - Vulns
  - Urgent/Important fixes
- The less you deploy, the worse you get at it
  - Process subverted for quick fixes
  - Process rot
- Delays cause inefficiency
  - Merge hell
  - Wasted code & time

# What's wrong with them?

Releases are a source of **friction**.  
The larger the friction, the more a project struggles.

# What's wrong with them?

Releases are **feared**.

The less you confront your fear, the more a project struggles.

# The Development Landscape (again)

- Business Demands
- Team instability
- Near-realtime dev requirements
- Technical Debt

# Continuous Delivery

- The ability to deploy at any time
- Two requirements:
  - Deployment process should be automated
  - Mainline branch should always be deployable
- Mechanics: Commit, Test, Deploy

# Commit

- Small commits
- If possible, all commits!
- Need a tactic to hide feature development

# Feature Branches

- Well understood and easy to use
- Be careful - Merge Hell needs to be managed
- GitHub uses this model

# Feature Flags (Flippers/Toggles)

- "Feature branching in code"
- Can be as complex as you want
  - "Dark launch"
  - Enable for admins only, (random?) list of users, user cohort, QA team...
  - Enable after a certain date
- Built-in kill switches

# Test

- Continuous Integration (poorman: manual)
- Should complete quickly
- When they fail, the impending deployment is abandoned

# Deploy

- Triggered by a successful run of the tests (poorman: manual)
- As few commands as possible! Should be frictionless
- **Should be no cause for fear**

# What happens if things break?

- Should always be possible to roll back to the previous commit
  - The smaller the changes are, the easier this is
- Write a test to cover the break
- Strategy assumption: **"Every Defect Once"**

# "It sounds too risky!"

- CD is actually a risk **reduction** strategy
- Presupposes what you're doing now is working ;)
- Going slower decreases **frequency** of risk, but massively increases **magnitude**

# Benefits

- More responsive
- Solid incentive to develop good testing habits
- Less pressure
- Less waste
- Improved developer attitude

# Core Benefit

CD lets you **go fast with confidence**

# "But what about... ?"

- "I work for a financial institution"
  - Your problem is the test suite, not CD
- "You're outsourcing QA to your customers!"
  - You can still involve QA - deploy to a staging environment simultaneously, or use feature flags
  - You already outsource QA to your customers to some extent anyway ;)
- "Batching up changes is safer"
  - Waiting longer does **not** correlate with higher quality
  - Other eyes checking **does** - which you can do without batching

# But what about... ?

- "We'd never get signoff for this"
  - "We're losing millions" never goes through signoff either
- "Legislation forbids us from..."
  - Devs change stuff all the time, in violation of legislation!  
See previous point
  - Some dev always has root access
- "GST goes up to 15% on XXX..."
  - Manage dependency issues with feature flags
- "Our tests now take hours to run!"
  - Test cluster
- "That's all good for you SaaS guys, but..."
  - Google Chrome, Moodle, IOS

# Final Thoughts

- CD has significant mindshare in the tech startup community
- Major players are already adopting it
- You will be left behind - nobody who switches to CD switches back
- Try it on a small/new project to test the concept if you need

# EOF

T: @nigelmcnie

W: nigel.mcnie.name

E: nigel@mcnie.name