Evolving Continuous Delivery

Chris Read
What We’re Told
What We’re Told

- Single Source Repository

@ cread   http://chris-read.net
What We’re Told

• Automate Build and Testing
What We’re Told

- Publish Latest Distributable

@cread

http://chris-read.net
What We’re Told

- Every Commit Builds

@cread

http://chris-read.net
What We’re Told

• Test in Production Like Environment

@cread

http://chris-read.net
What We’re Told

• Keep Builds Fast

@cread

http://chris-read.net
What We’re Told

• Use Information Radiators

@cread  http://chris-read.net
What We’re Told

• Automate Deployment

@cread

http://chris-read.net
What We’re Told

- Build Binary *Once*
What We’re Told

• Promote Binary Through Stages
Evolution
Initial State

• New team of talented and impatient developers

• Starting to create trading applications for an established desk
Initial State

- Releasing daily from developer workstations to production
- **No** Continuous Integration!
- Using **Fig** for dependency management
Stage 1

- **Standardise** and Refactor the build scripts
- Add **Continuous Integration** server
- Set up an **Information Radiator**

@cread

http://chris-read.net
Stage II

- Create a standard deployment script
- Turn the scripts into dependencies
- Try a different CI server
Stage III

- **Sideline** the Continuous Integration loop
- **Bake** the Continuous Integration *safeties* into the deployment scripts

@cread

http://chris-read.net
Stage IV

- Automate server builds
- Start to scale services out
Stage V

- Fracture services out into stacks
- Stage the binaries
- Fast rollbacks
Stage VI

• Onward...
How Do We Do It?

- Process
- Principles
- Heresy
Process
Trad Agile Process

Dev
Code
Test
QA
CI
Stage
Prod
User

I week

http://chris-read.net

@cread
Our Team’s Process

30 mins

@cread

http://chris-read.net
Pros and Cons

- Fast
- Low Cycle Time
- Simple Flow
- Narrower Scope
- Smaller Change Delta
- Less Complex Bugs

- More Bugs in Prod

@cread  
http://chris-read.net
Principles

• Requires *architectural vision* & *discipline*
• **Short code half life** & *unix philosophy*
• *Minimise risk* by co-locating customers
• *Prodigious monitoring*
Heresy

• Minimal use of frameworks
• Polycopyism
What Have I Learned?

- The Things We’re Told still hold, but implementation will vary greatly
- Constantly re-evaluate your tools and your processes
- Always question the return on investment

@cread http://chris-read.net
Thank You