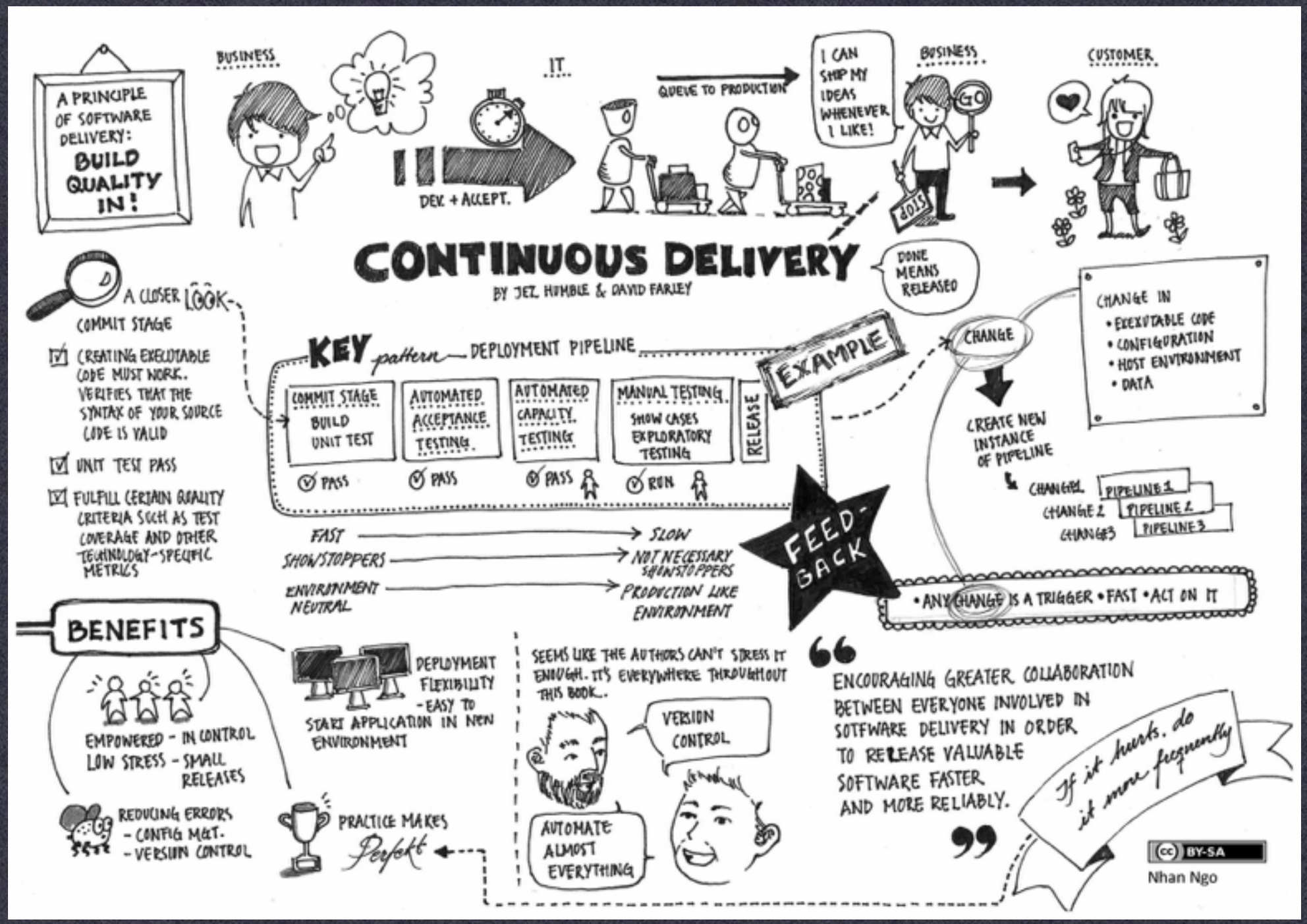
Chris Parnin: https://github.com/CSC-DevOps/Course



DevOps Deployment

Reid Holmes

https://commons.wikimedia.org/wiki/File:Continous_Delivery_by_Jez_Humble_and_David_Farley.jpg

Deploying Code

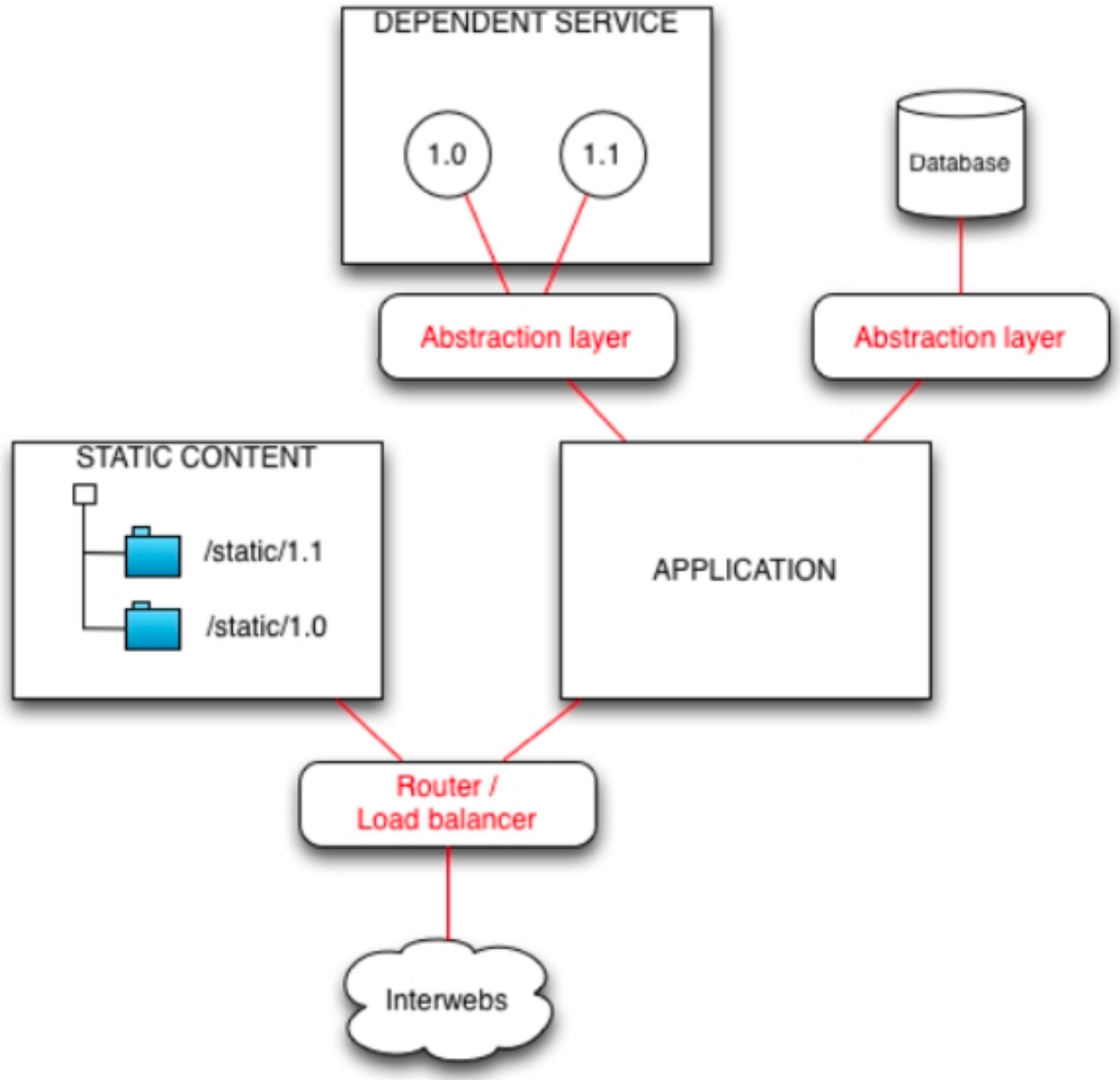
Lowering the risk of change through tools and culture.

CHUCK ROSSI

While tooling automates deploying changes, ultimately this is a human process.

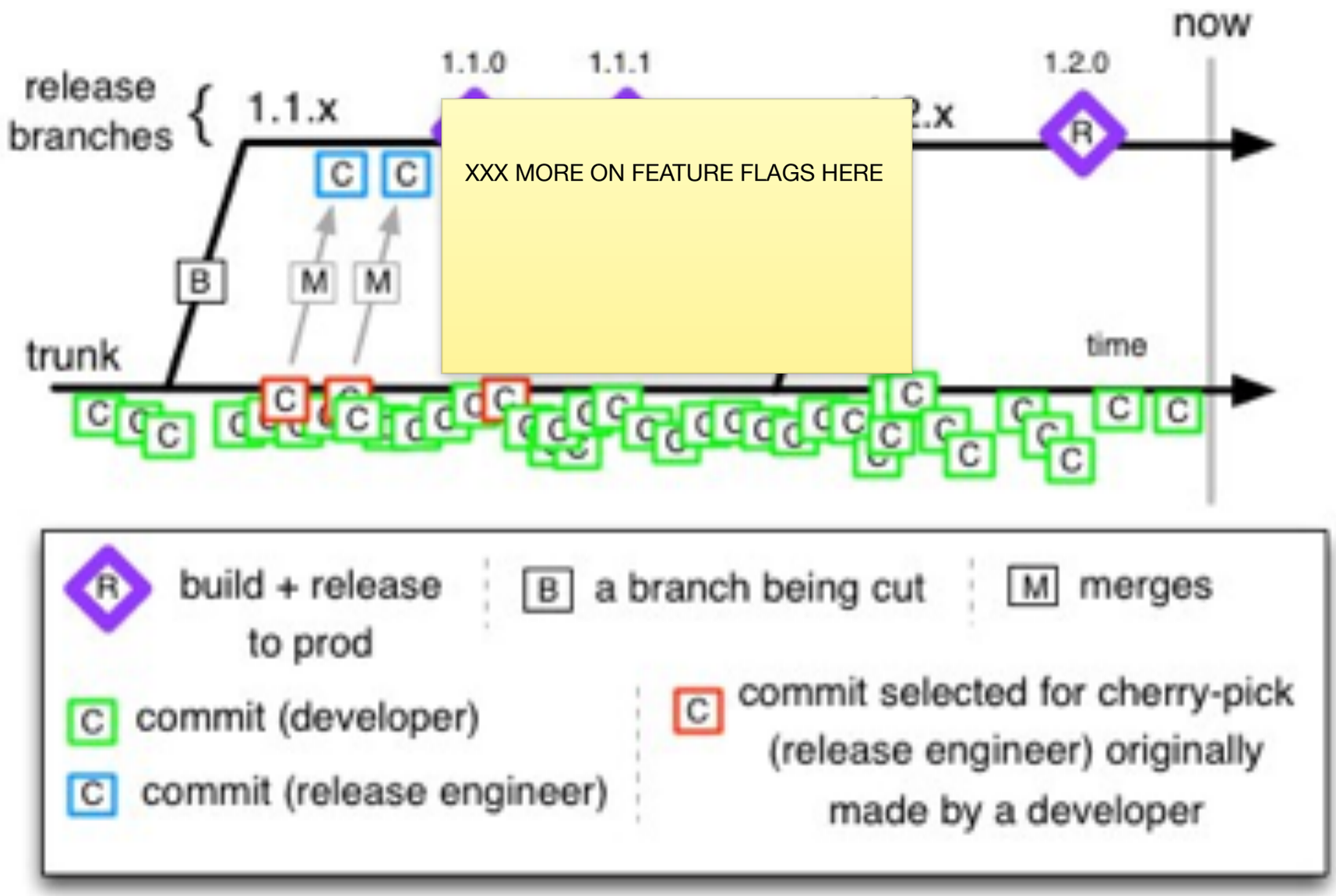
HTTPS://VIMEO.COM/56362484

Incrementalism



Facebook example

- New release branch every Tuesday.
- Revs are cherry-picked to prod as needed.



https://vimeo.com/56362484

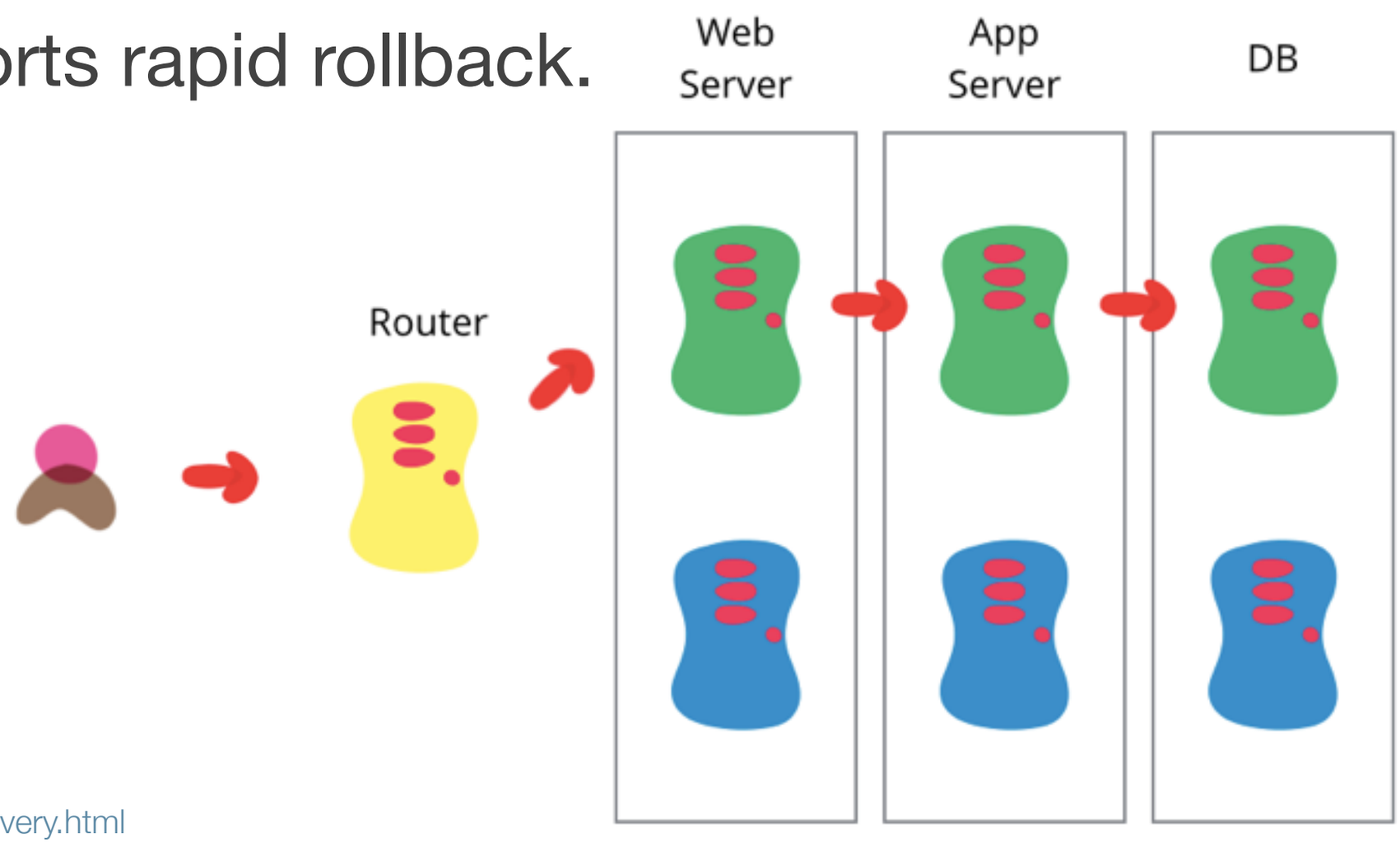
Low-Risk Release Principles

1. Favour incremental changes

http://www.informit.com/articles/article.aspx?p=1833567

Green/Blue Deploy

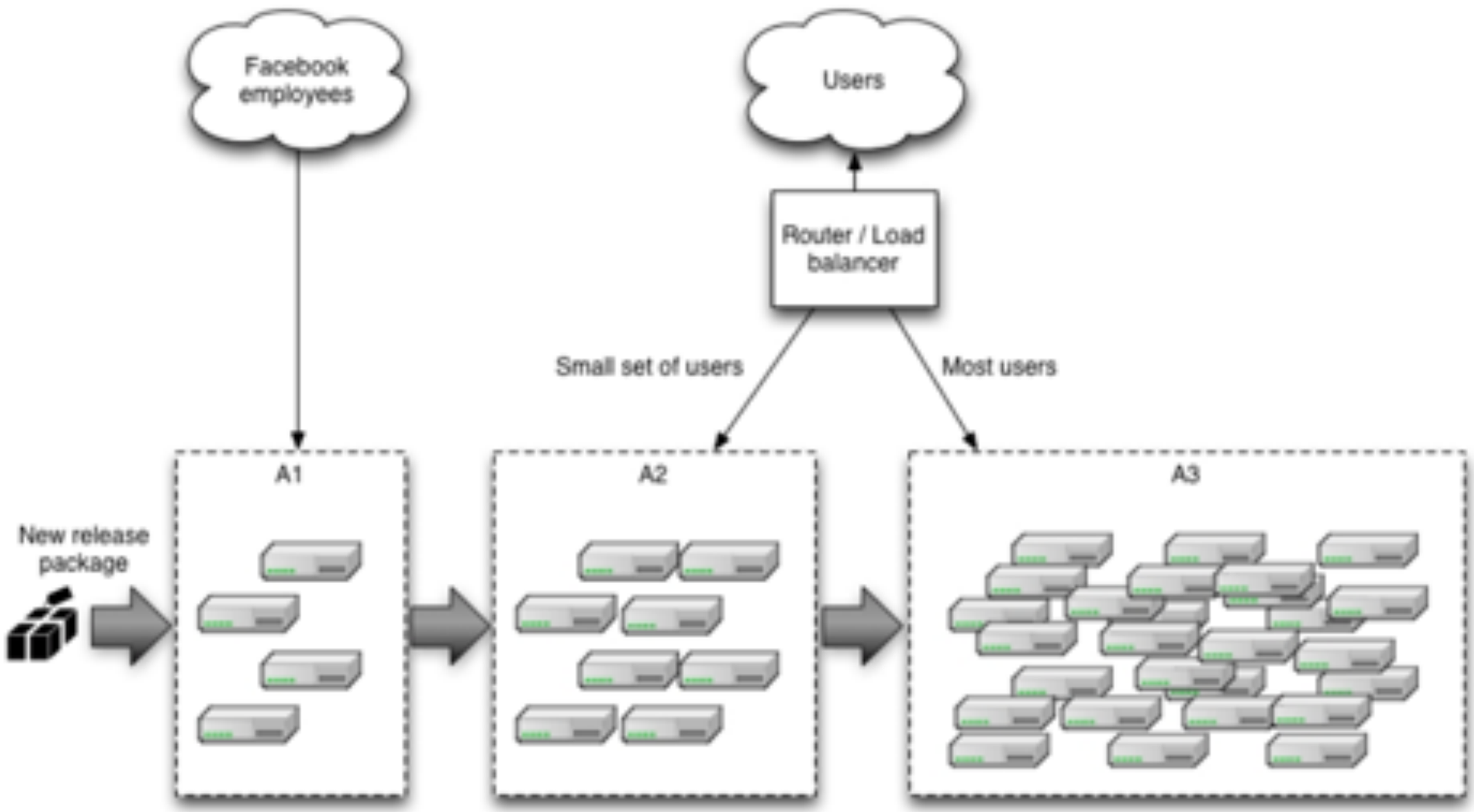
- Two production environments.
- Mechanism for completing deploy on full stack.
- Router then manages user migration to new bits.
- Supports rapid rollback.



http://martinfowler.com/delivery.html

Canary Releases

- Quickly signal bad builds to halt rollouts.



Low-Risk Release Principles

1. Favour incremental changes
2. Decouple deploying from releasing

http://www.informit.com/articles/article.aspx?p=1833567

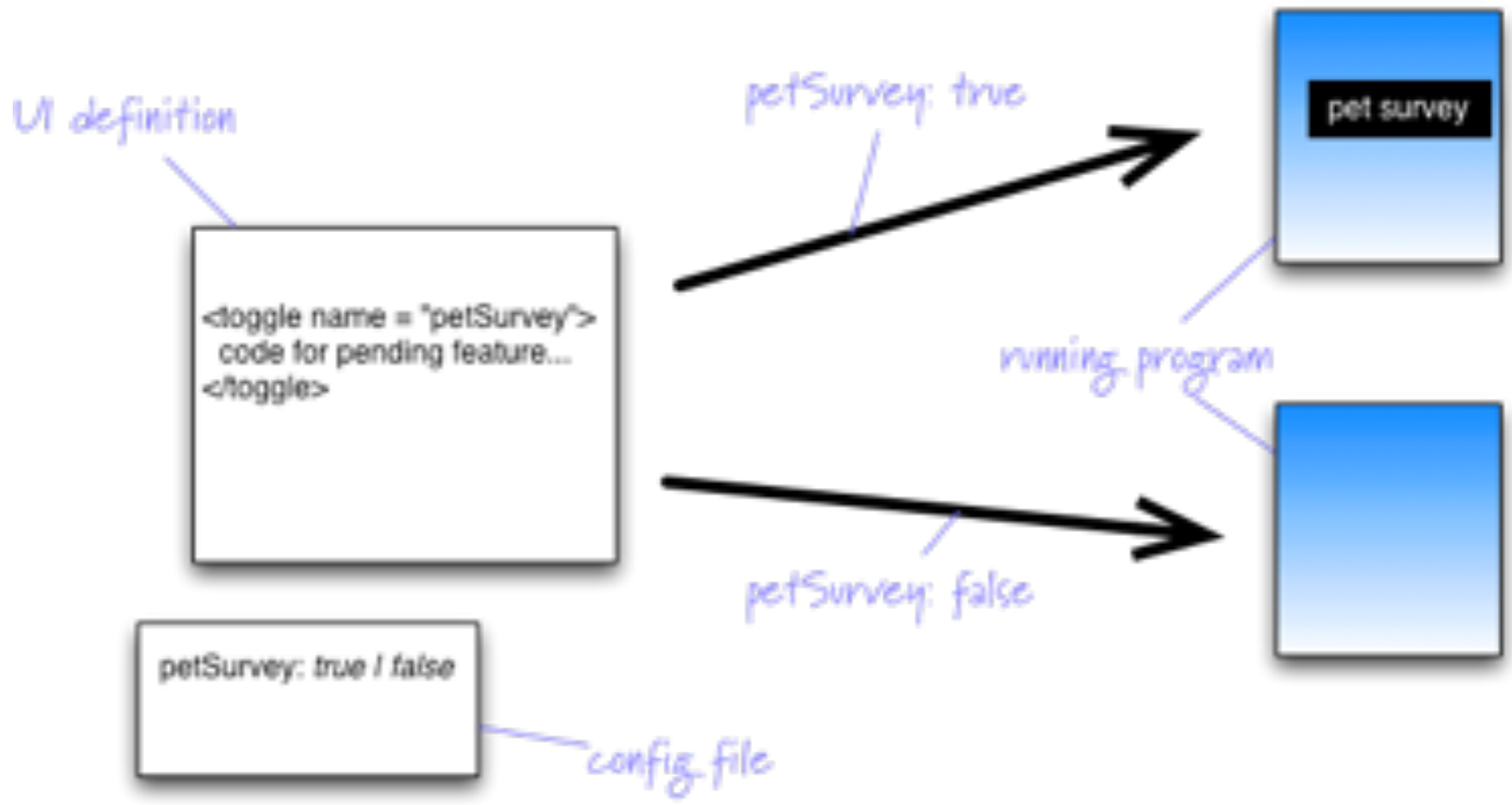
Dark Launch

- Release changes without their user-facing elements.
- Can simultaneously direct traffic through previous and dark-launched code for load testing.
- Feature flags can enable/disable dark code.
- Shortens release branch duration.
- Improves disaster recovery.

http://swreflections.blogspot.ca/2014/08/feature-toggles-are-one-of-worst-kinds.html
http://martinfowler.com/bliki/FeatureToggle.html

Feature Flags

- Gatekeeper can direct specific subsets of traffic to newly-launched code to gather data/feedback.



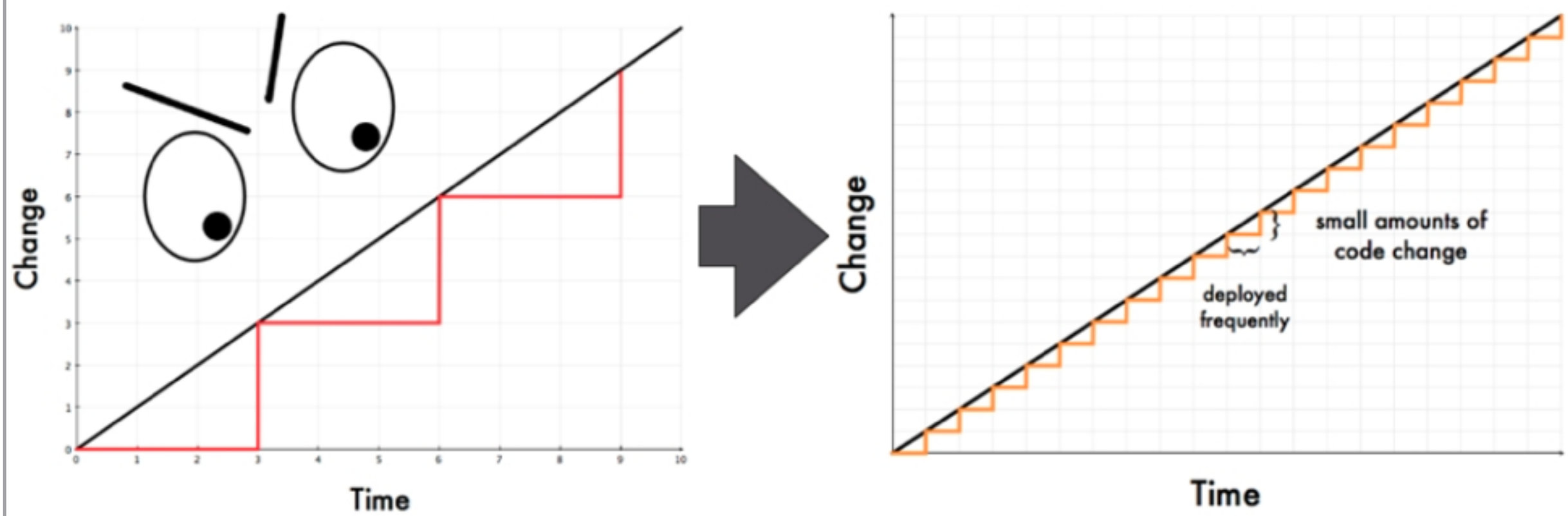
Low-Risk Release Principles

1. Favour incremental changes
2. Decouple deploying from releasing
3. Focus on reducing batch size

http://www.informit.com/articles/article.aspx?p=1833567

High Release Cadence

- Deploys new features to users quickly.
- Enables responsive defect resolution.
- Smaller delta == smaller faults.
- Releasing loses ‘dark art’ status.



http://www.slideshare.net/jallspaw/ops-metametrics-the-currency-you-pay-for-change

deploymonitor

logged in as: ekastner [switch] QA: 25145-trunk-20100518-220720-UTC Princess: 25145-trunk-20100518-220720-UTC
Production: 25145-trunk-20100518-220720-UTC

Deploy to QA (Trunk)

deploying revision: 25159

Message:

Push to QA →

Princess is in the other castle

deploying version: 25145-trunk-20100518-220720-UTC

Save the Princess

Deploy to Production

deploying version: 25145-trunk-20100518-220720-UTC

PROD!!! →

Important links:

Log

!! Commits since last prod deploy !!

☑ Auto scroll command output?

Add an arbitrary log message:

PROJECT-### to link to jira and add a comment log

- [web] | 2010-05-18 22:20:50 | PRODUCTION | sandrews | Production Deploy: old 25134, new: 25145 diff
- [web] | 2010-05-18 22:18:00 | PRINCESS | sandrews | Princess Deploy: old: 25144, new: 25145 diff
- [web] | 2010-05-18 22:17:22 | QA | sandrews | kyles bug fix old: , new: 25145 diff
- [web] | 2010-05-18 22:12:03 | QA | sandrews | pushing again -- banned user cache busting old: , new: 25144 diff
- [web] | 2010-05-18 22:06:39 | PRINCESS | sandrews | Princess Deploy: old: 25134, new:
- [web] | 2010-05-18 22:02:36 | QA | sandrews | [redacted] old: 25134, new: 25144 diff
- [web] | 2010-05-18 20:56:50 | PRODUCTION | cmunns | Production Deploy: old 25134, new: 25134 diff
- [web] | 2010-05-18 20:49:02 | PRODUCTION | cmunns | Production Deploy: old 25134, new: 25134 diff
- [web] | 2010-05-18 20:44:43 | PRODUCTION | ahashim | Production Deploy: old 25030, new: 25134 diff
- [web] | 2010-05-18 20:41:17 | PRINCESS | ahashim | Princess Deploy: old: 25030, new: 25134 diff
- [web] | 2010-05-18 20:40:38 | QA | ahashim | peanut butter jelly time!!!!! old: 25030, new: 25134 diff
- [web] | 2010-05-17 16:23:26 | PRODUCTION | sandrews | Production Deploy: old 24951, new: 25030 diff
- [web] | 2010-05-17 16:12:12 | PRINCESS | sandrews | Princess Deploy: old: 24951, new: 25030 diff
- [web] | 2010-05-17 16:08:05 | QA | sandrews | kissmetrics amongst others old: 24951, new: 25030 diff
- [web] | 2010-05-14 20:16:47 | PRODUCTION | zgarrett | Production Deploy: old 24884, new: 24951 diff

https://codeascraft.com/2010/05/20/quantum-of-deployment/

Low-Risk Release Principles

1. Favour incremental changes
2. Decouple deploying from releasing
3. Focus on reducing batch size
4. Optimize for resilience

http://www.informit.com/articles/article.aspx?p=1833567

Have a Plan B

- No. Really.
- Culture is fundamental to identifying, fixing, and recovering from large distributed faults.
  - Do you know you there's a problem?
  - Can you figure out what it is?
  - Can you find out who should fix it?
  - Fix it. (we're good at this)
  - Do we know how to deploy it?
  - Can we validate the problem is fixed?

Testing plan B

NETFLIX

- Fail often, tolerate failure.
- Learn with scale, not models.
- Netflix Simian Army:
  - Latency
  - Conformity
  - Doctor
  - Janitor
  - Security
  - i9n, l18n

http://techblog.netflix.com/2011/07/netflix-simian-army.html
http://techblog.netflix.com/2010/12/5-lessons-weve-learned-using-aws.html

Admin Stuff - D2 Rubric

Architecture (/5):
1) Is there a component diagram? Can the architecture of the system be understood from it?
2) Have architectural styles been applied?
3) Have architectural decisions been justified at all?
4) Are external and phone-based services included on diagram / in description.
5) Is an explanation of how NFPs are supported and are measurable included?

Design (/7):
1) Is a rationalization given for the chosen design?
2) Is there a description of the class organization? (even tying back to arch components is fine here).
3) Are key patterns / abstractions / data structures documented?
4) Is there a mapping of the architectural components to the design components?
5) Is the ‘class’ diagram clear, capture physical location of classes, and external elements?
6) Is there a description of how coupling was minimized?
7) Is a future point of evolution provided?

Admin Stuff - Testing

Testing is the most fundamental approach for measuring software quality.

Testing is not easy.

Testing is not optional.

You control your architecture and can refactor your system as needed to effectively test its internals.