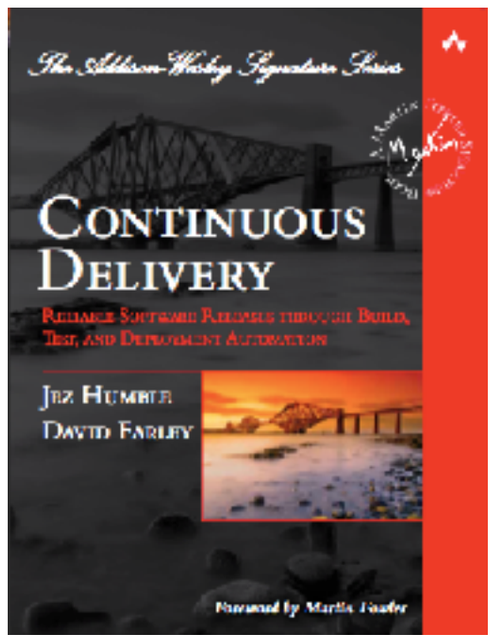
architecting for continuous delivery

@jezhumble

devops enterprise | 20 october 2015



microservices

what is continuous delivery?

The ability to get changes—features, configuration changes, bug fixes, experiments—into production or into the hands of users *safely* and *quickly* in a *sustainable* way.

golden rules

software (trunk) is always deployable

everyone is checking into trunk daily
(at least)

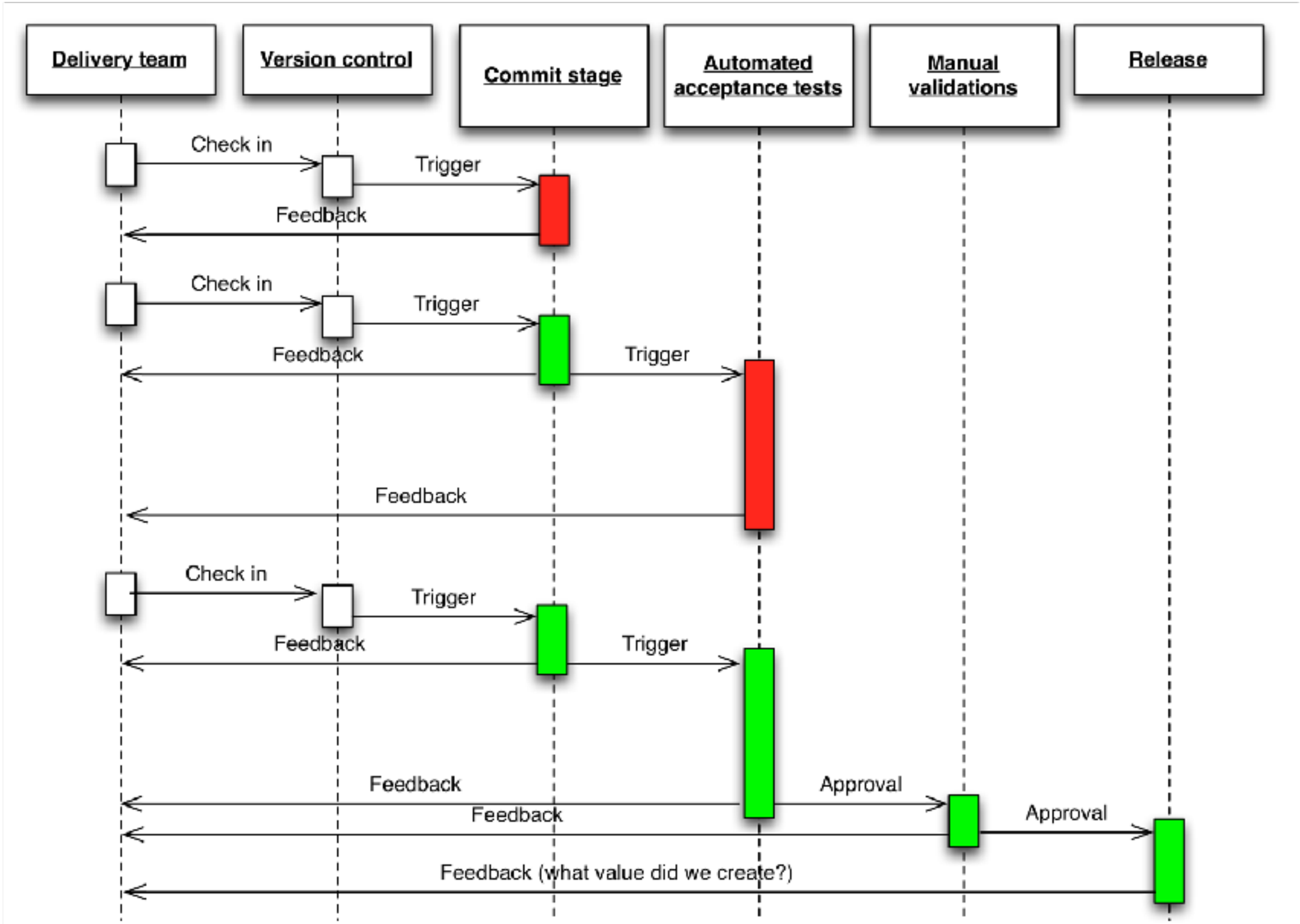
ingredients

configuration management

continuous integration

automated testing

deployment pipeline



to validate golden rules

testability

"works on my machine"

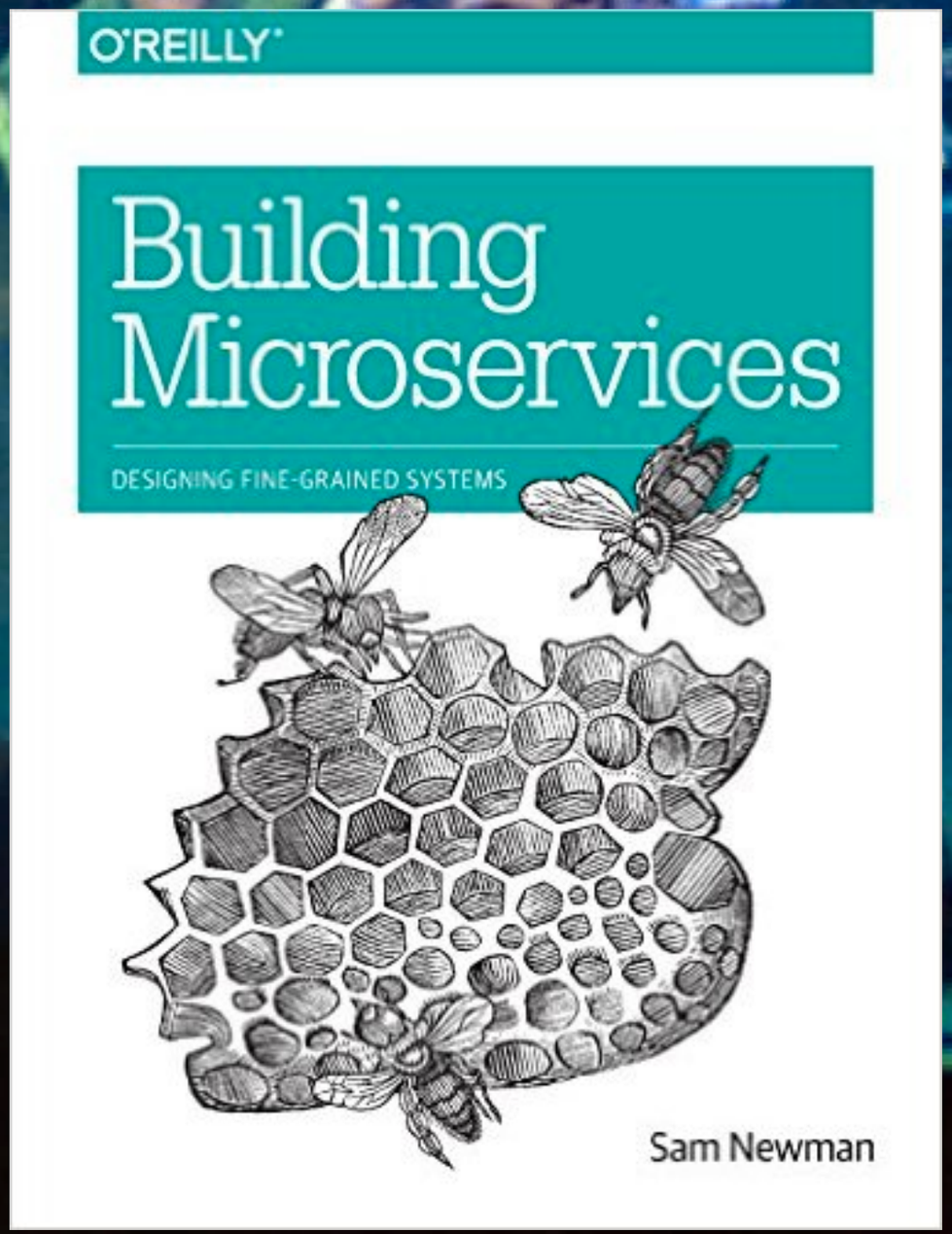
actually means something useful

deployability

deployment is a low-risk,

pushbutton affair

microservices



internet architecture

"Operations at web scale is the ability to consistently create and deploy *reliable software* to an *unreliable platform* that scales horizontally."

Jesse Robbins, "Master of Disaster" @ Amazon| @jesserobbins | http://oreil.ly/1HRKUVE

component / service

part of your system that could be swapped out for another implementation

make system more maintainable (better encapsulation, lower coupling)

make system easier to build and test

enable collaboration

conway's law

"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

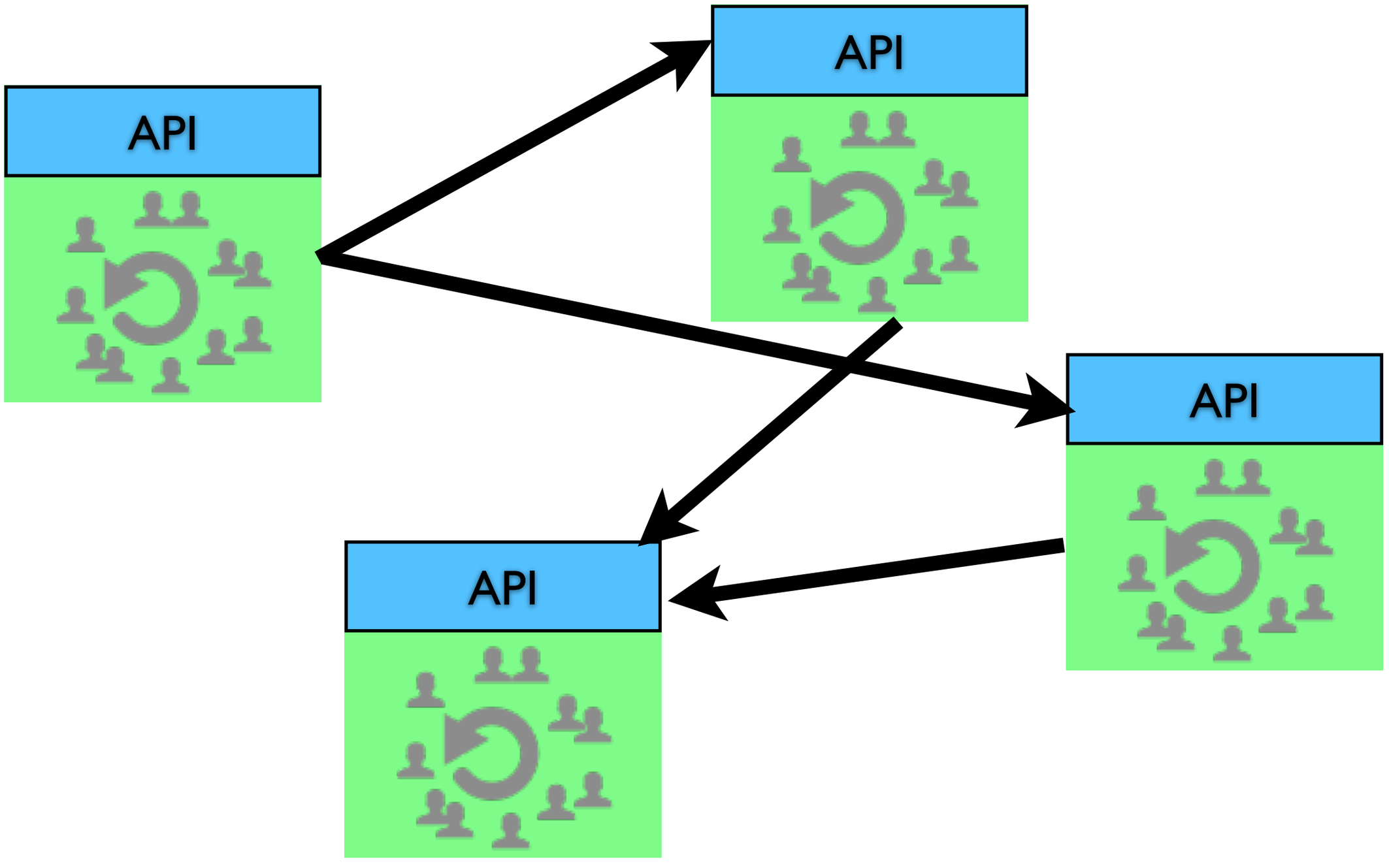
—Melvin Conway

conway's law

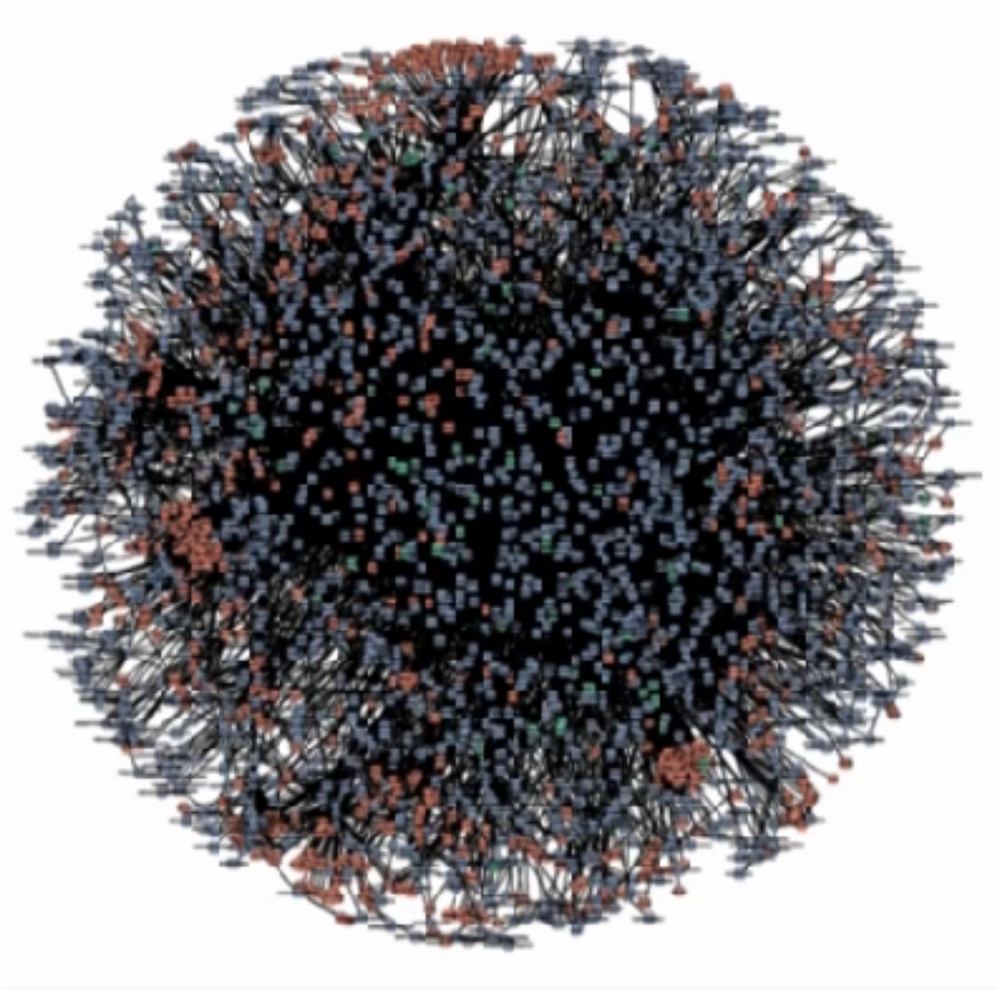
"If you don't want your product to look like your organization, change your organization or change your product."

—Rebecca Parsons

one team per service



Bind Components at Run Time (µS)



Independent deployment

Don’t break downstream!

API versioning

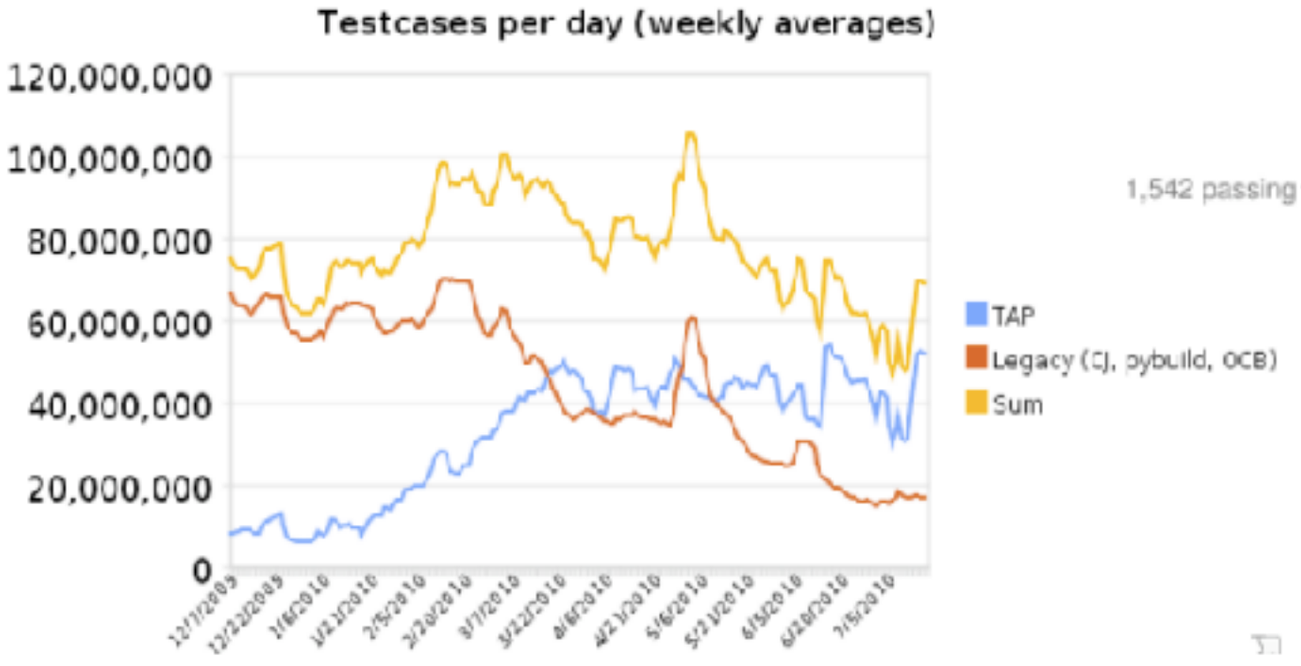
Monitoring

“Transforming Software Development” | Ken Exner | http://bit.ly/1HxCEZy

Bind Components at Build Time

Continuous Integration at Scale

- 200K test suites in the code base
- Run 10M test suites per day
- > 60M individual test cases / day and growing
- > 4000 continuous integration builds



“Large Scale Continuous Testing in the Cloud” | John Penix | http://bit.ly/1BYMf70

Unreliable Platform

Resilience, security, scalability, availability deployability, testability *are architectural concerns.*

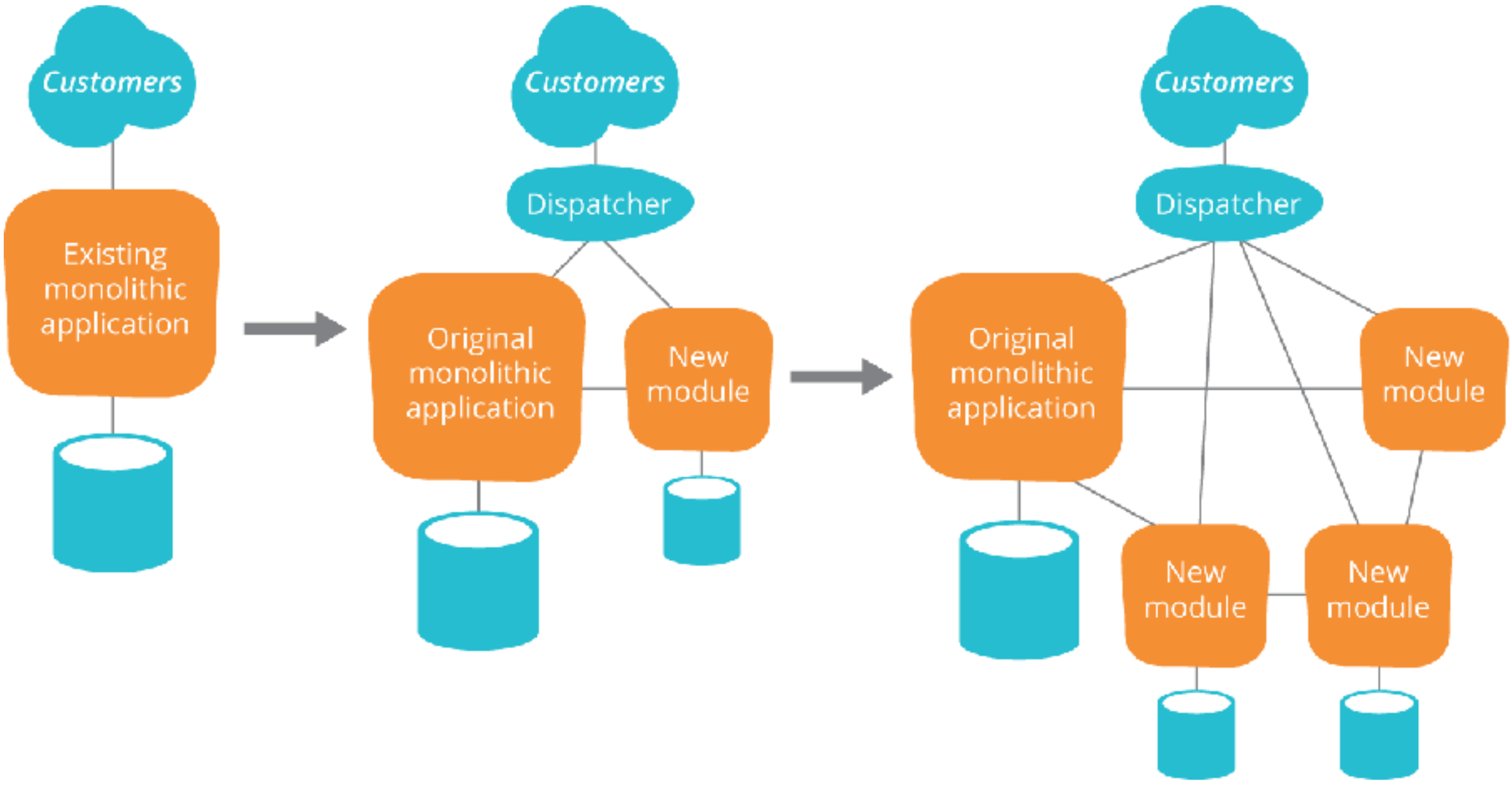
1. All teams will henceforth expose their data and functionality through service interfaces.
2. Teams must communicate with each other through these interfaces.
3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols—doesn't matter. Bezos doesn't care.
5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
6. Anyone who doesn't do this will be fired.

Steve Yegge's Platform Rant | http://bit.ly/1zxknpR

Werner Vogels, CTO, Amazon | http://bit.ly/11UJjmN

http://www.flickr.com/photos/trustedsource/6132507962/

Strangler Application



devops benchmarking

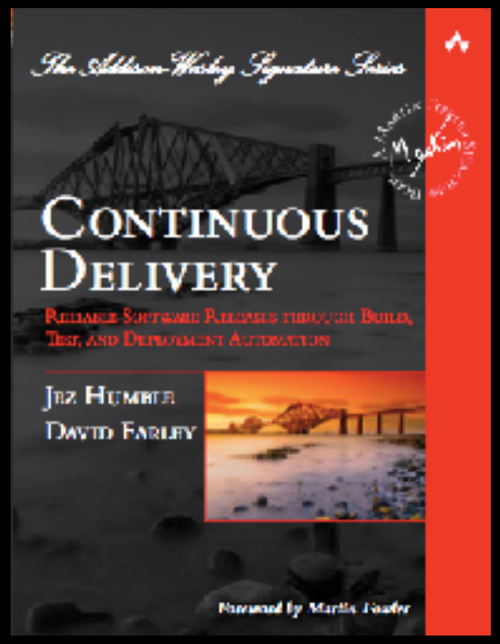
From the team that helped create the Puppet Labs *State of Devops Report*: Gene Kim, Nicole Forsgren, Jez Humble.

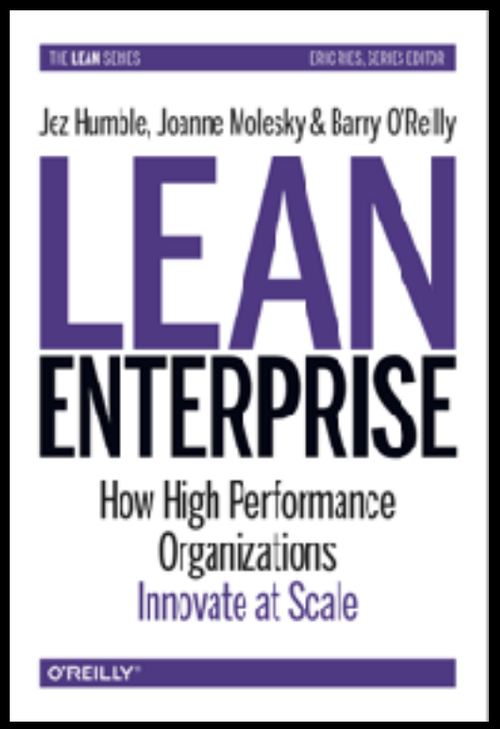
- Benchmark your performance across four axes: culture, automation, process, measurement
- Compare your IT performance against the industry standard
- Get a personalized report with your results that you can share

want to learn more?

To receive the following:

- **An exclusive invite to our DevOps benchmarking tool**
- A chance to get a personalized analysis of your results
- A copy of this presentation
- A 100 page excerpt from *Lean Enterprise*
- *A* 20m preview of my Continuous Delivery video workshop
- Discount code for CD video + interviews with Eric Ries & more
- Early drafts of the *DevOps Handbook*





Just pick up your phone and send an email

To: jezhumble@sendyourslides.com

Subject: devops

CC BY NC SA © 2015 Humble, O'Reilly & Associates LLC