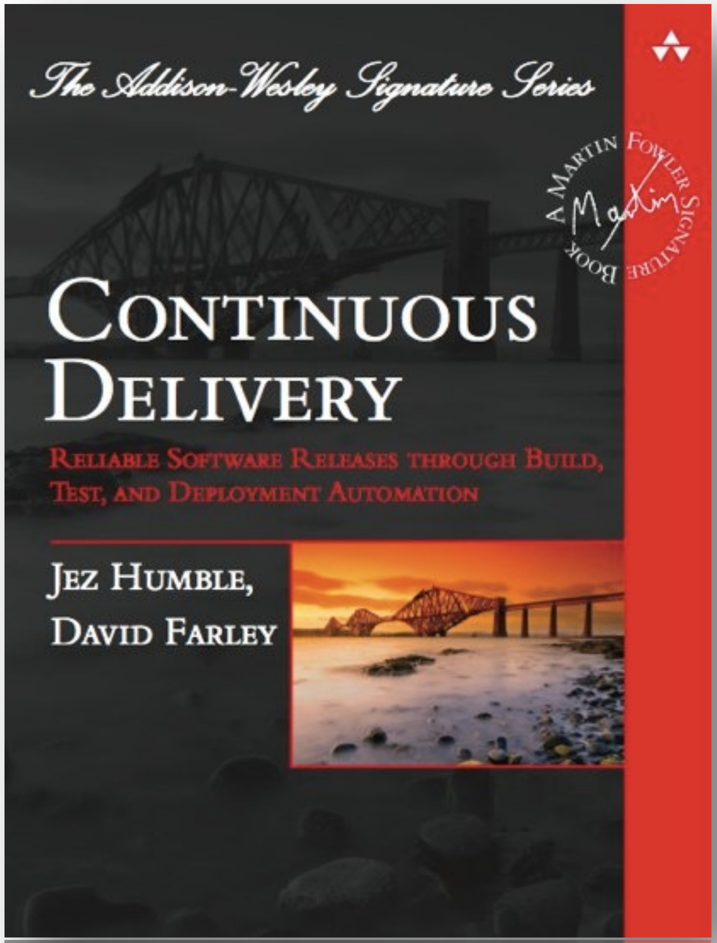
ThoughtWorks®

Continuous Delivery Workshop


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble, David Farley

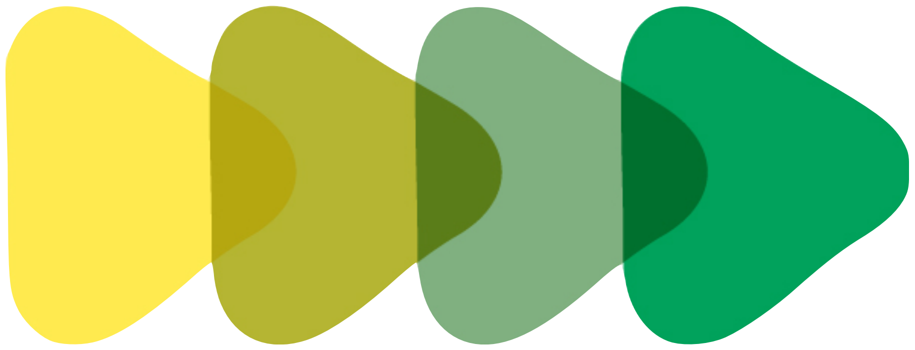
deployment pipelines

ThoughtWorks®

NEAL FORD

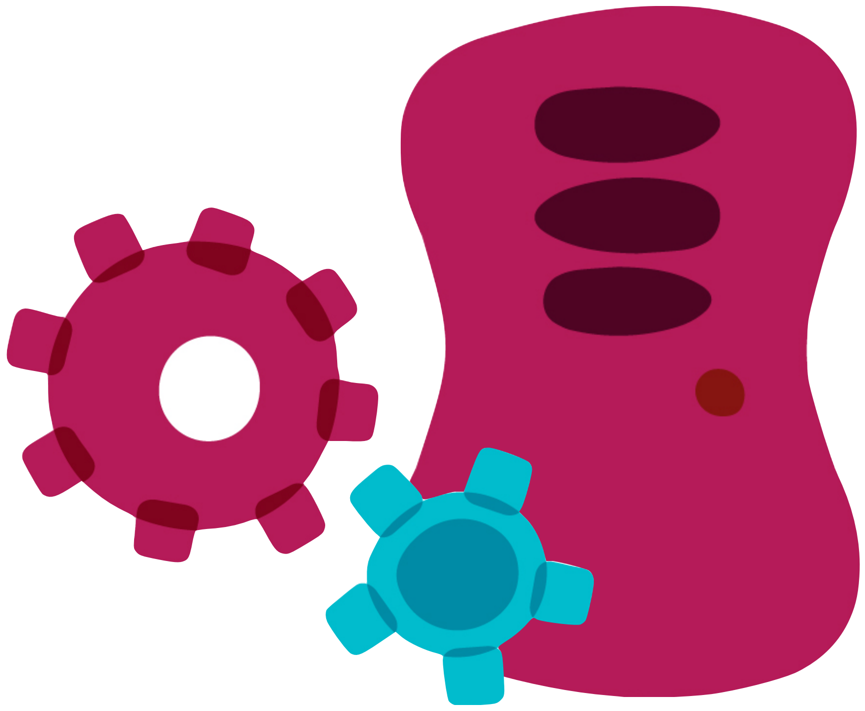
Director / Software Architect / Meme Wrangler

NF *Workshop materials created by Jez Humble, Martin Fowler, Tom Sulston, & Neal Ford*



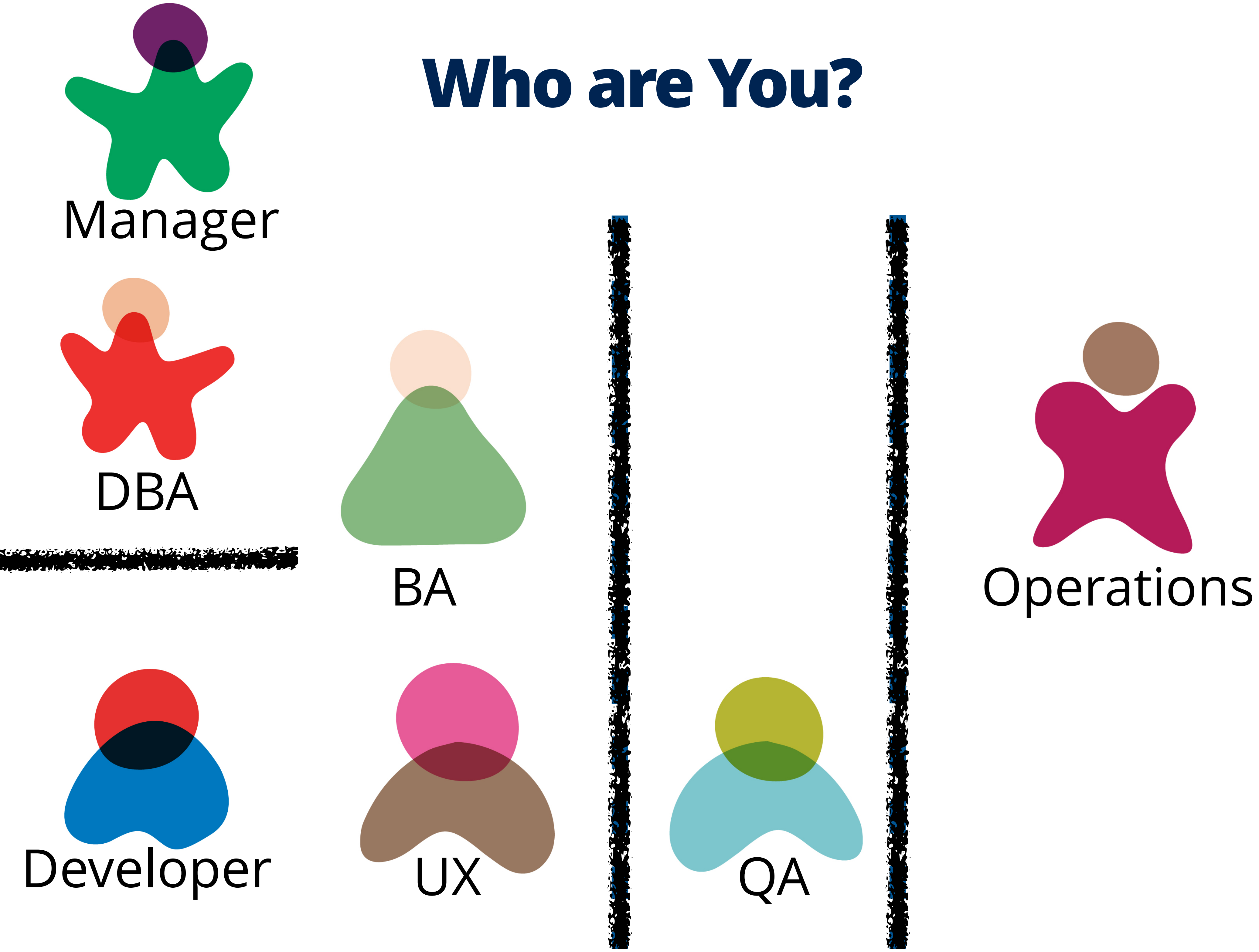
deployment pipelines

tests, synergistic practices, incremental deployment

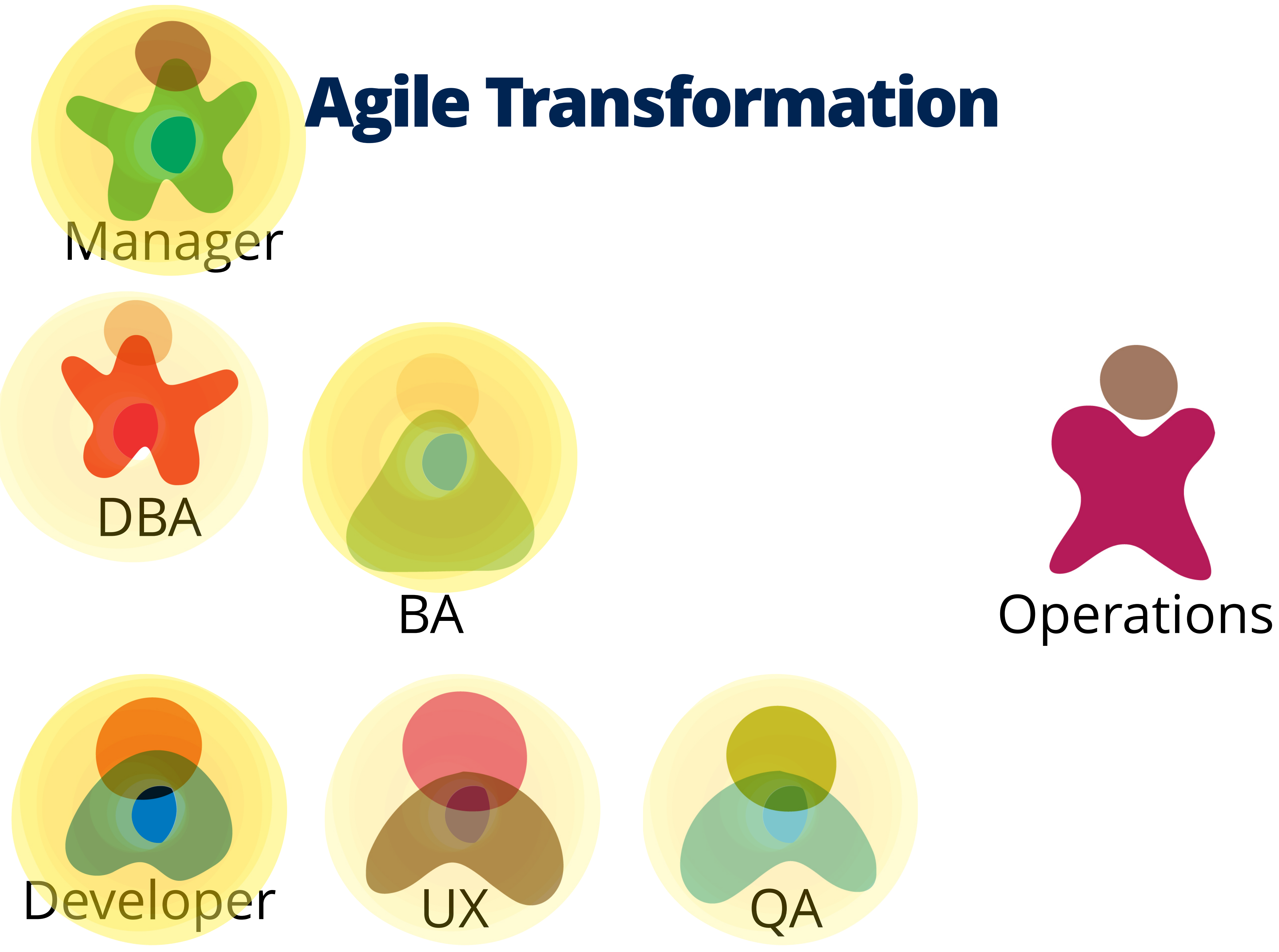


data & infrastructure

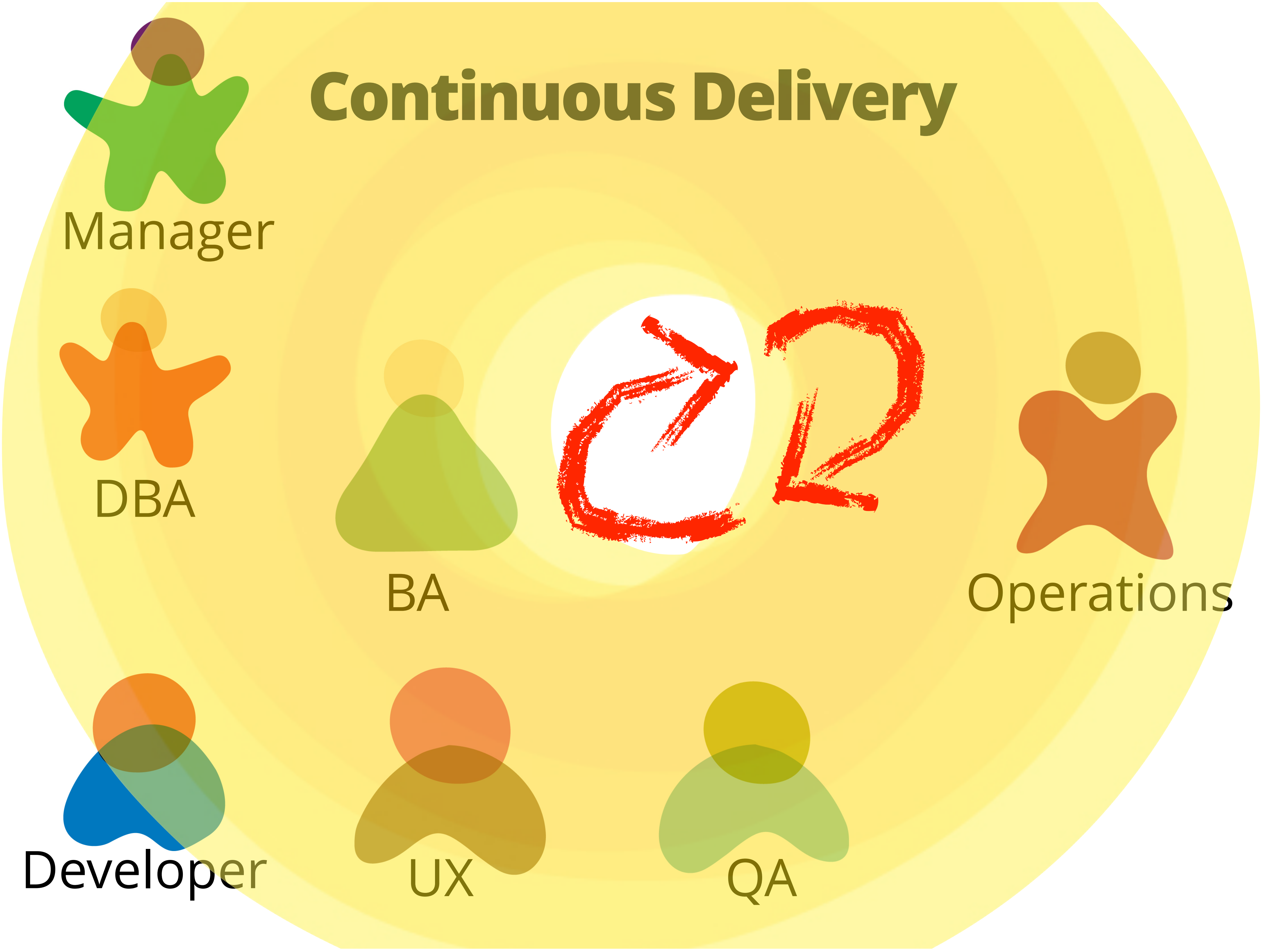
Who are You?

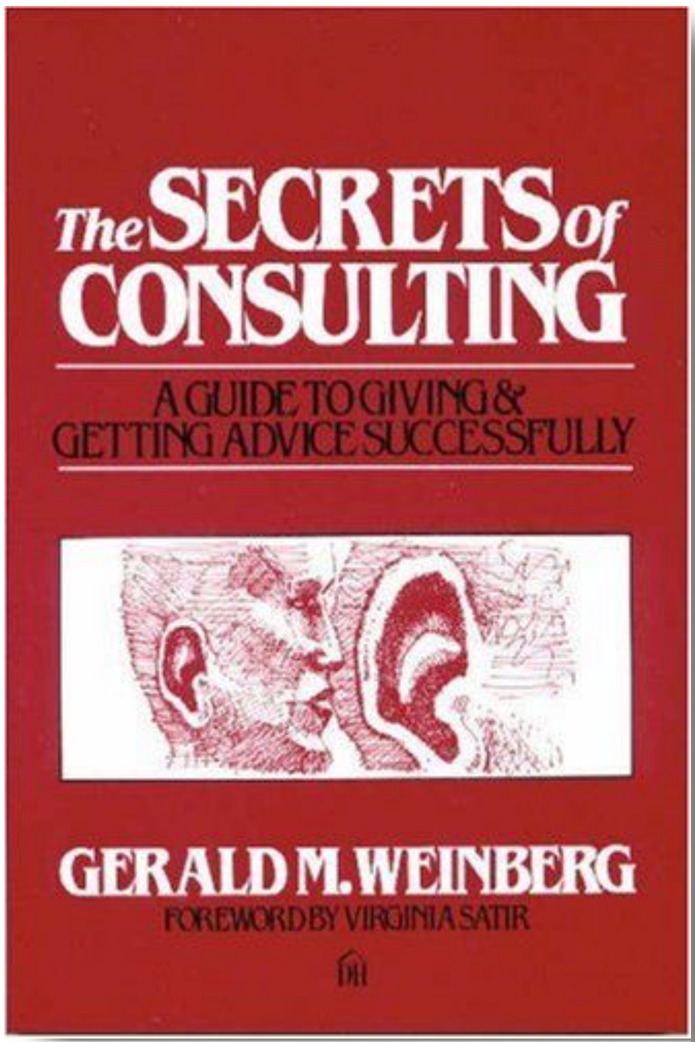


Agile Transformation



Continuous Delivery





The SECRETS of CONSULTING

A GUIDE TO GIVING & GETTING ADVICE SUCCESSFULLY

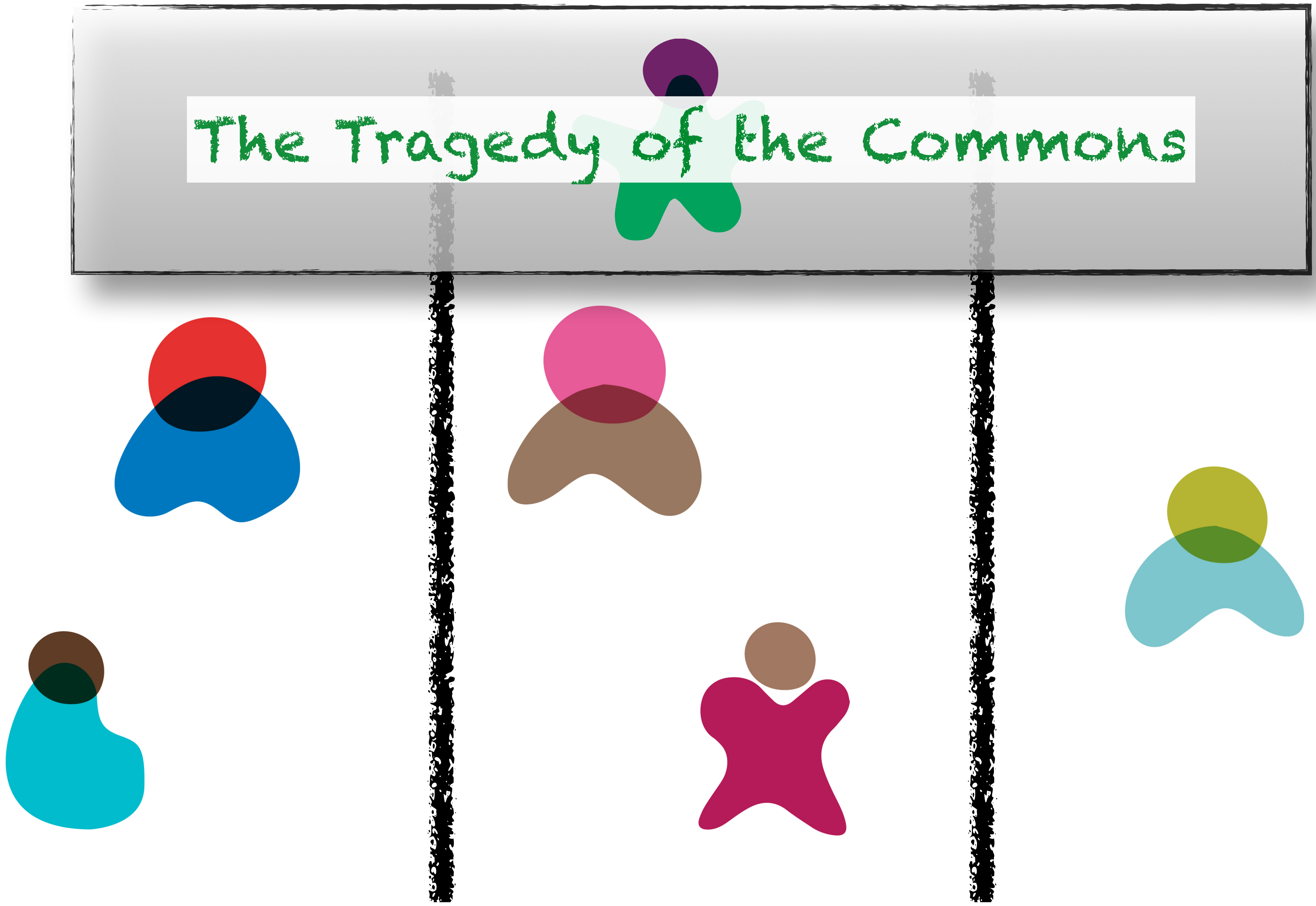
GERALD M. WEINBERG

FOREWORD BY VIRGINIA SATIR

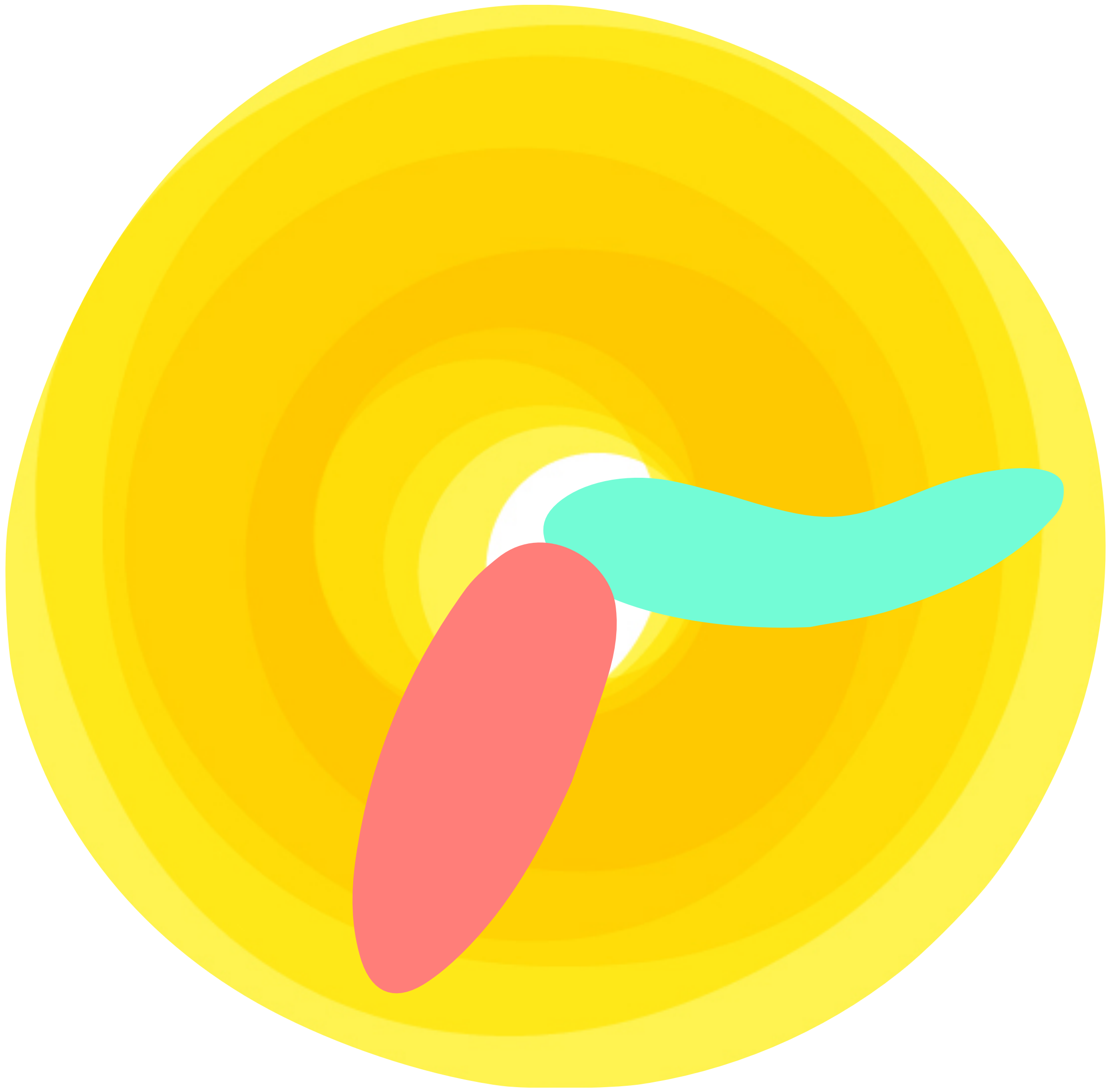
DH

"No matter how it looks at first, it's always a people problem."

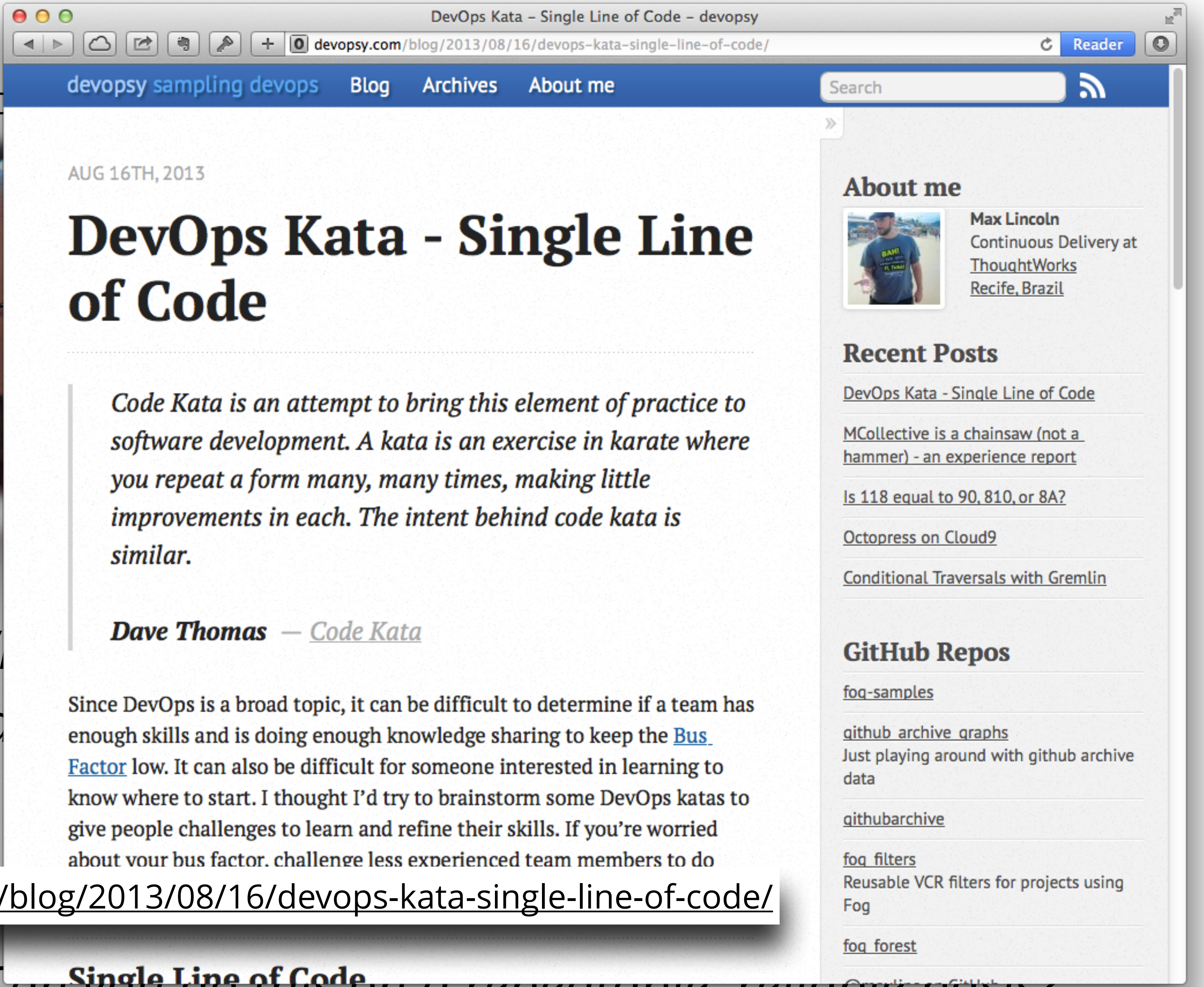
The Dangers of Silos



Release Cadence



Mary P



DevOps Kata – Single Line of Code – devopsy

devopsy.com/blog/2013/08/16/devops-kata-single-line-of-code/

devopsy sampling devops | Blog | Archives | About me | Search

AUG 16TH, 2013

DevOps Kata - Single Line of Code

> *Code Kata is an attempt to bring this element of practice to software development. A kata is an exercise in karate where you repeat a form many, many times, making little improvements in each. The intent behind code kata is similar.*
>
> ***Dave Thomas*** — *Code Kata*

Since DevOps is a broad topic, it can be difficult to determine if a team has enough skills and is doing enough knowledge sharing to keep the Bus Factor low. It can also be difficult for someone interested in learning to know where to start. I thought I'd try to brainstorm some DevOps katas to give people challenges to learn and refine their skills. If you're worried about your bus factor, challenge less experienced team members to do

Single Line of Code

About me

Max Lincoln
Continuous Delivery at ThoughtWorks
Recife, Brazil

Recent Posts

- DevOps Kata - Single Line of Code
- MCollective is a chainsaw (not a hammer) - an experience report
- Is 118 equal to 90, 810, or 8A?
- Octopress on Cloud9
- Conditional Traversals with Gremlin

GitHub Repos

- fog-samples
- github_archive_graphs
  Just playing around with github archive data
- githubarchive
- fog_filters
  Reusable VCR filters for projects using Fog
- fog_forest

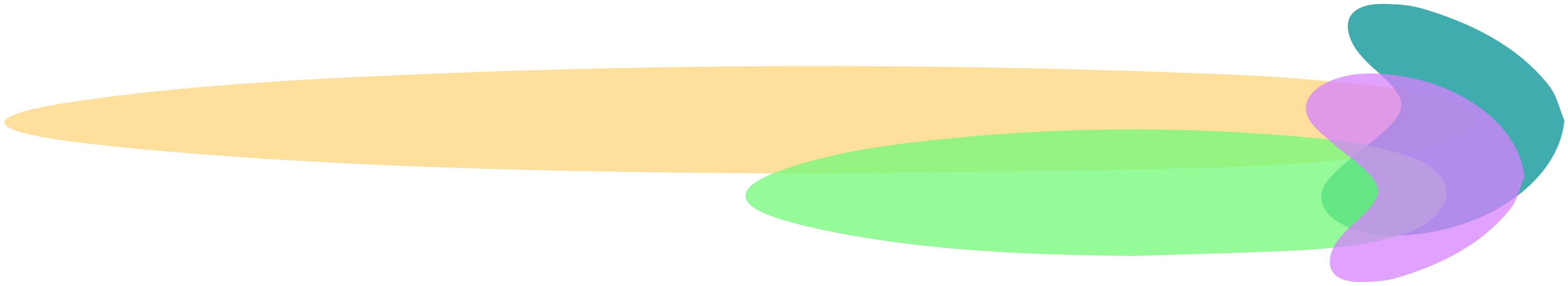
devopsy.com/blog/2013/08/16/devops-kata-single-line-of-code/

"Can you do this on a repeatable, reliable basis?"

Continuous Delivery Metrics

lead time

the time between the initiation and completion of a production process.



cycle time

the total elapsed time to move a unit of work from the beginning to the end of a physical process

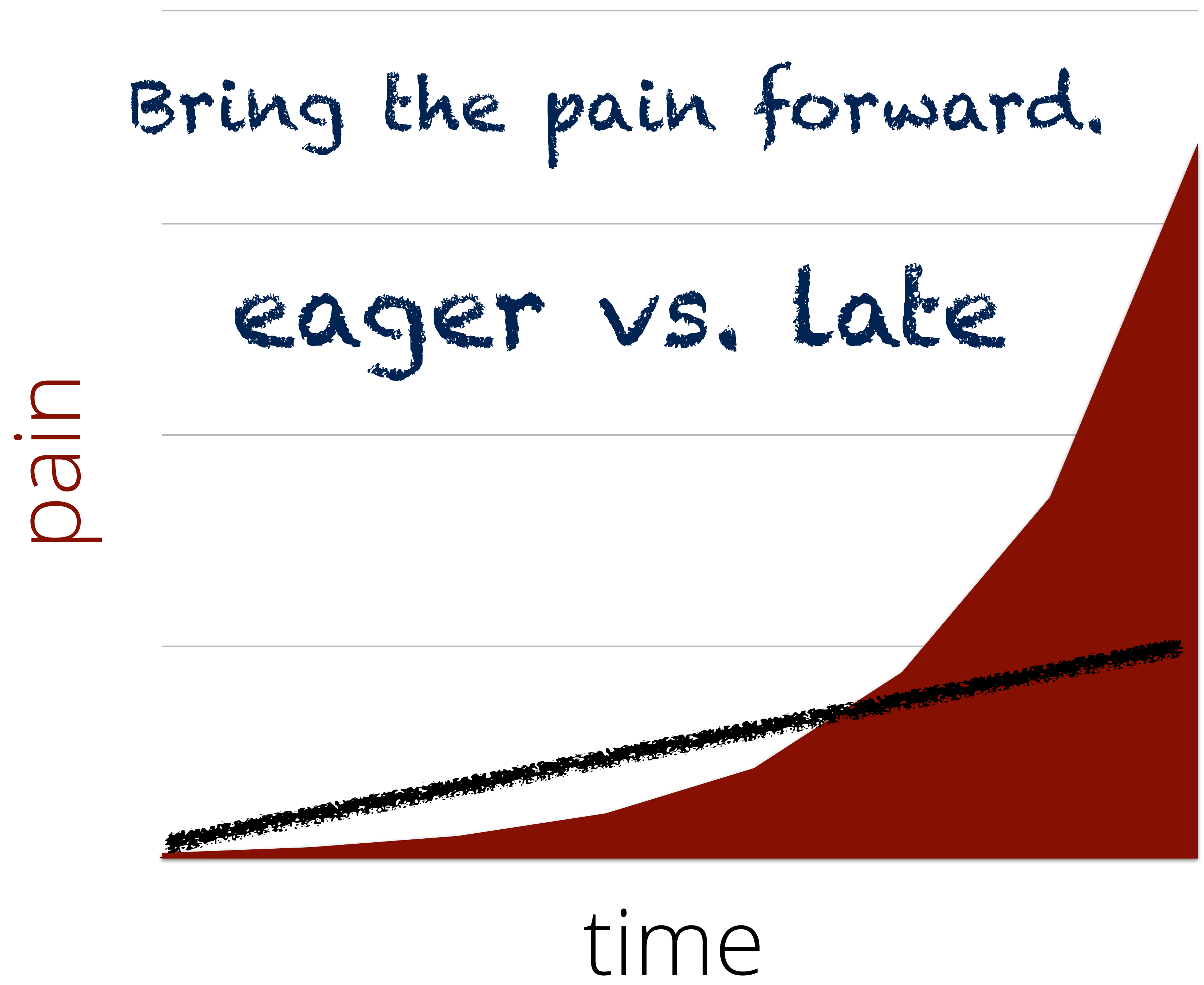
Continuous Integration

Integration early and often.

Everyone checks into `trunk` at least once a day.

Bring the pain forward.

eager vs. late



Continuous Integration

Integration early and often.

Everyone checks into **`trunk`** at least once a day.

Integration

Integration early and often.

Everyone checks into **`trunk`** at least once a day.

Continuous Deployment

Deploy as the final stage of continuous integration.

Integration

Integration early and often.

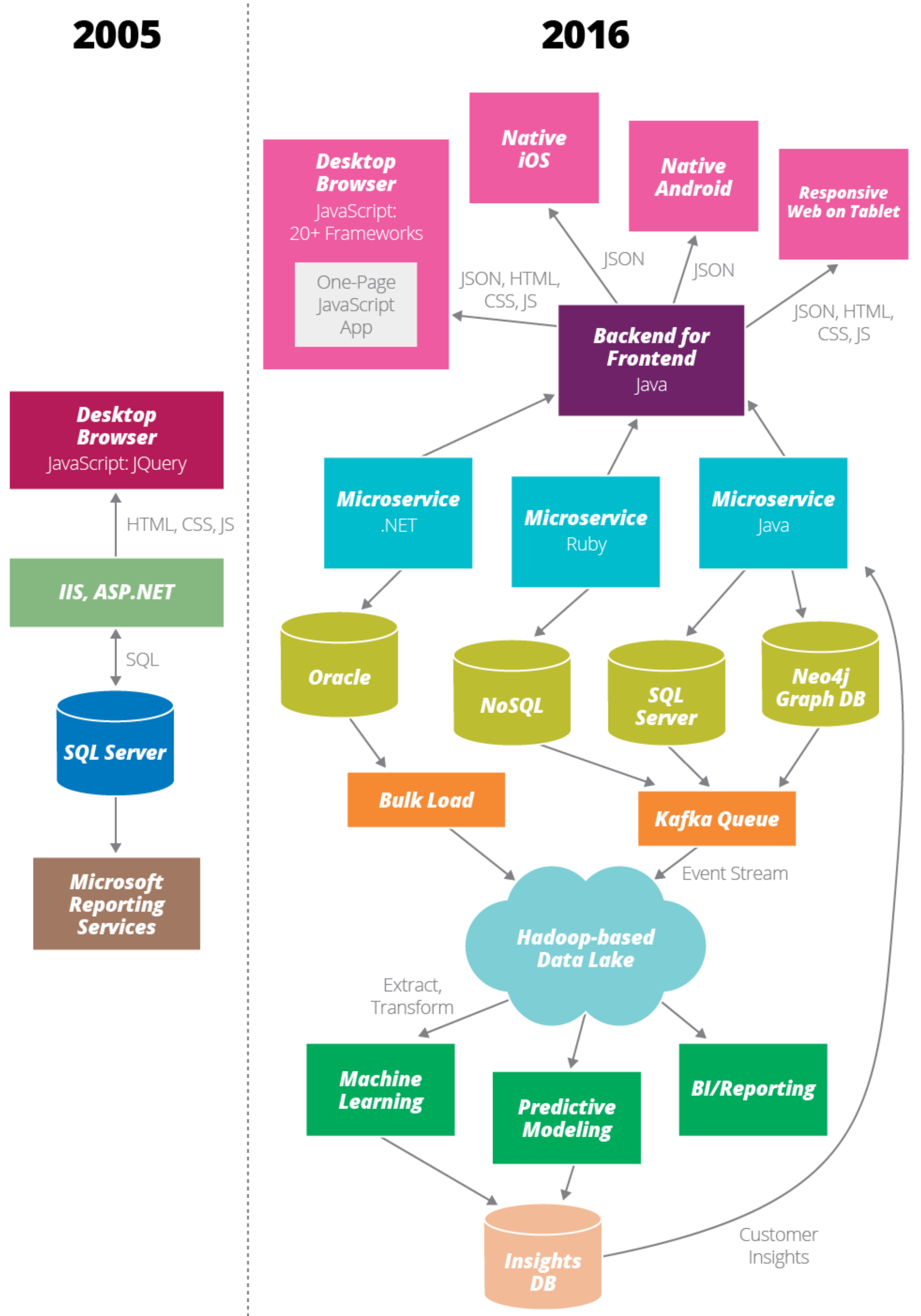
Everyone checks into **`trunk`** at least once a day.

Deployment

Deploy as the final stage of continuous integration.

Continuous Delivery

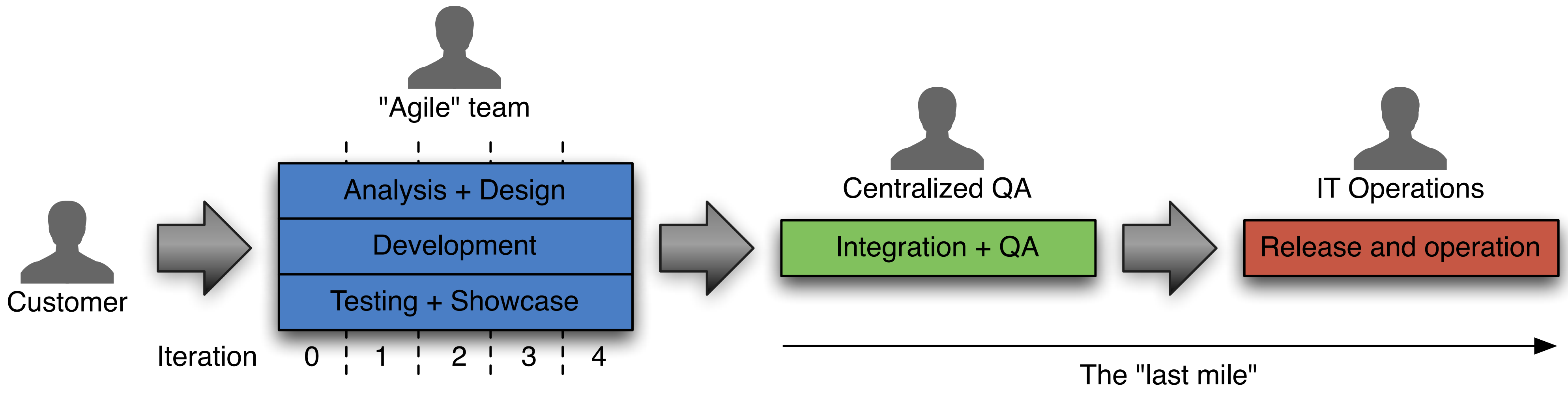
Software is always in a deployable state.



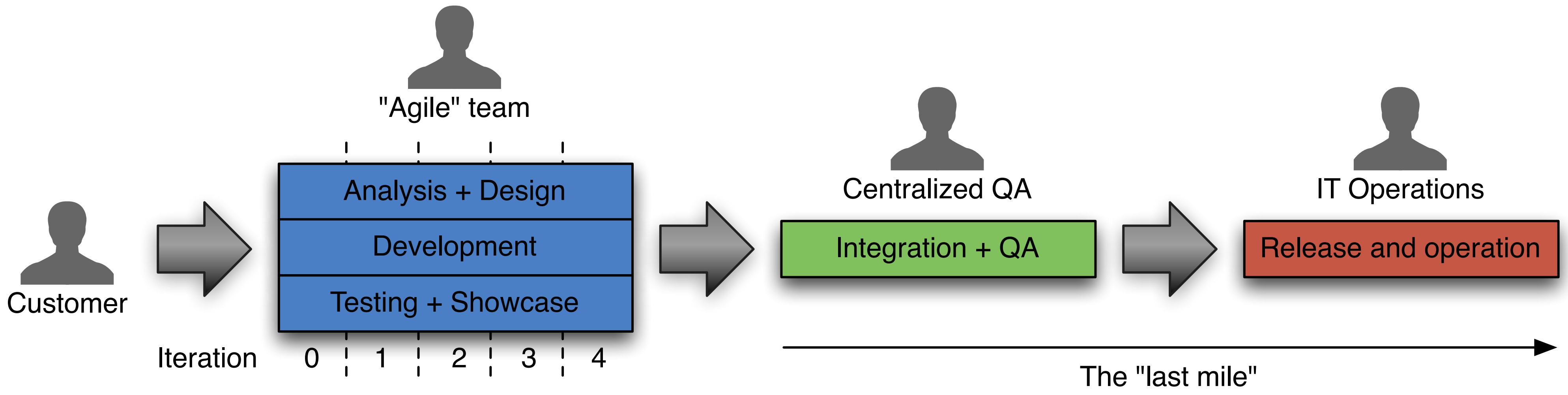
Modern software is *complex!*

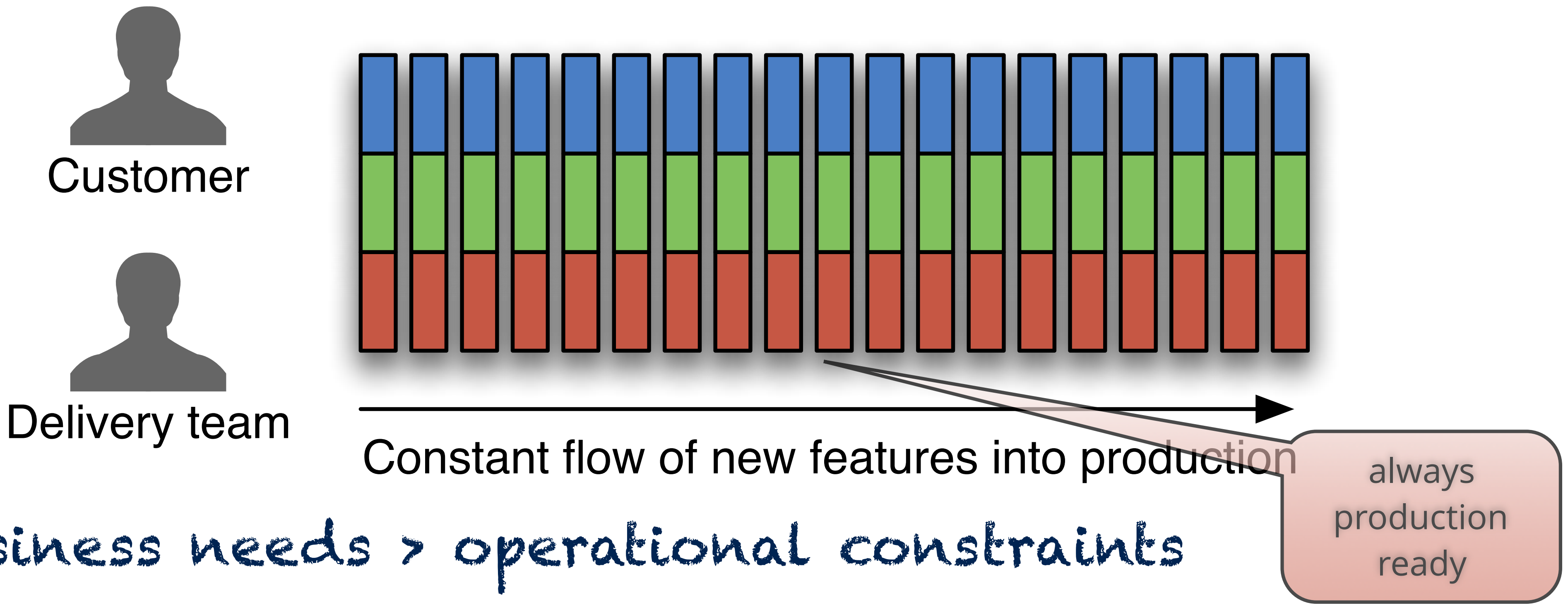
https://www.thoughtworks.com/insights/blog/implications-tech-stack-complexity-executives

Agile 101


"Agile" team
Customer
Analysis + Design
Development
Testing + Showcase
Iteration
0
1
2
3
4
Centralized QA
Integration + QA
IT Operations
Release and operation
The "last mile"

Continuous Delivery



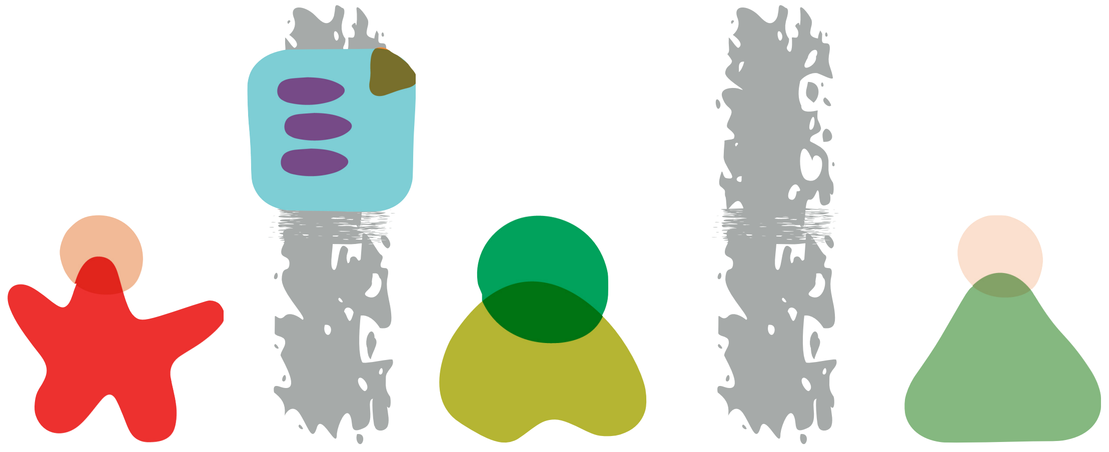


business needs > operational constraints

Potential Hindrances

Lead time is too long

Last mile is too painful



Poor collaboration

Identify & remove friction

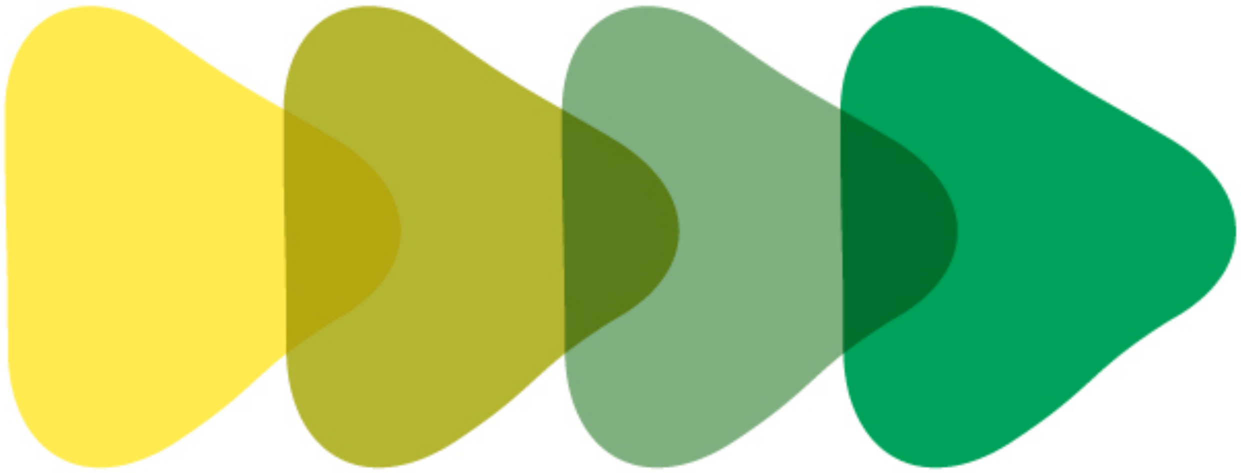
Continuous Integration

Fast, automated feedback on the correctness of your application every time there is a change to code

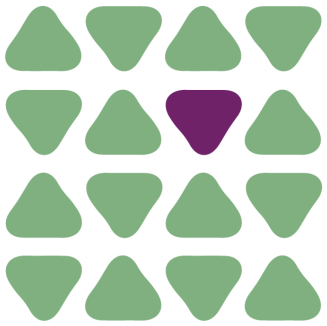
Deployment Pipeline

Fast, automated feedback on the production readiness of your application every time there is a change — to code, infrastructure or configuration

Deployment Pipelines



Prerequisites

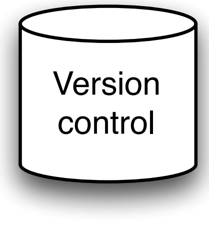


comprehensive configuration management

continuous integration

excellent automated testing at all levels

commit Stage



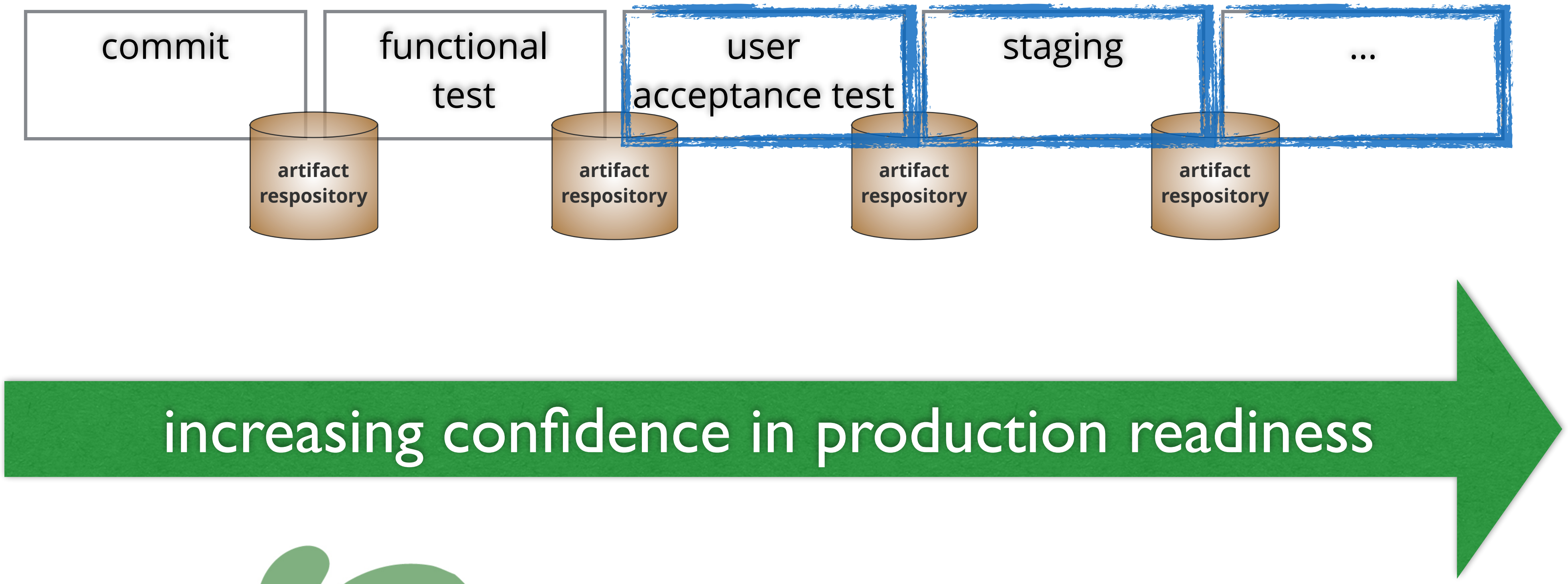
source code
commit tests
build scripts

Run against each check-in

Starts building a release candidate

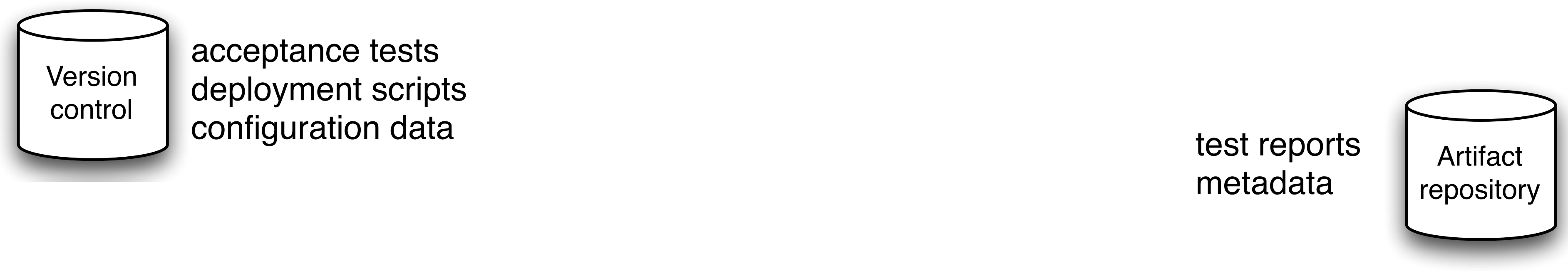
If it fails, fix it immediately

Pipeline Construction



Pipeline stages = feedback opportunities

UAT Stage

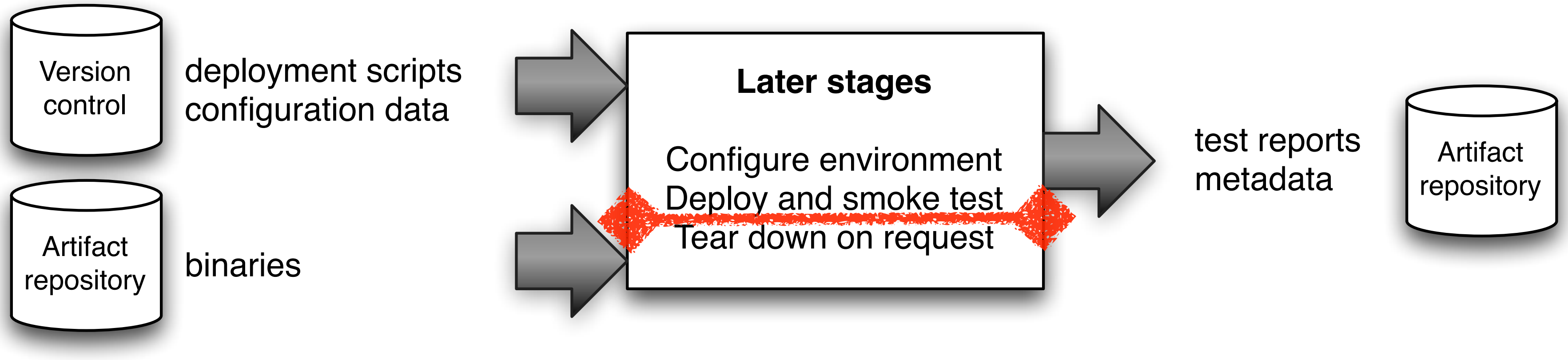


End-to-end tests in production-like environment

Triggered when upstream stage passes

First DevOps-centric build

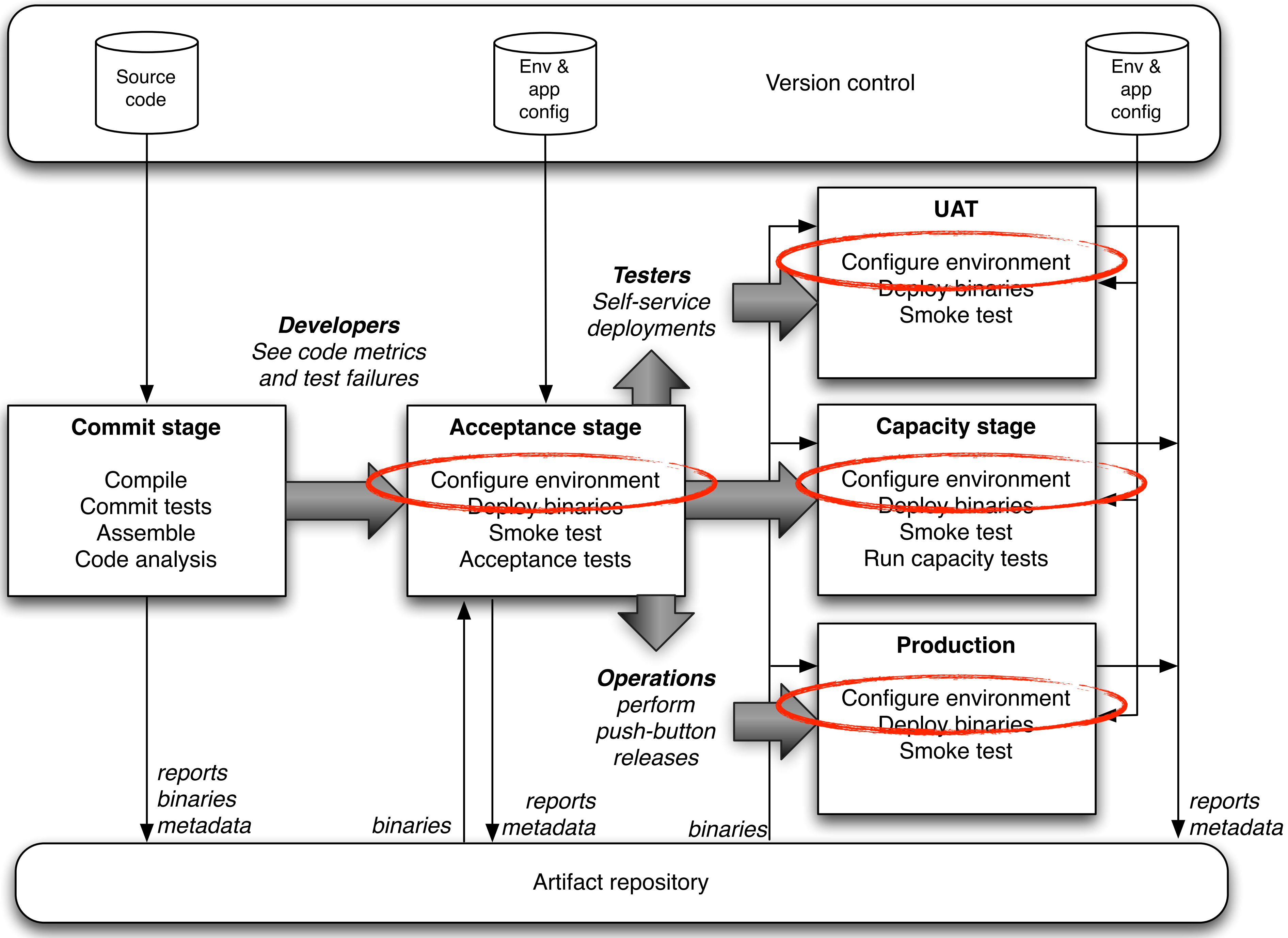
Manual Stage



UAT, staging, integration, production, ...

Push versus Pull model

Deployments self-serviced through push-button process


Source code
Env & app config
Version control
Env & app config
UAT
Configure environment
Deploy binaries
Smoke test
Testers
Self-service deployments
Developers
See code metrics and test failures
Commit stage
Compile
Commit tests
Assemble
Code analysis
Acceptance stage
Configure environment
Deploy binaries
Smoke test
Acceptance tests
Capacity stage
Configure environment
Deploy binaries
Smoke test
Run capacity tests
Operations
perform push-button releases
Production
Configure environment
Deploy binaries
Smoke test
reports binaries metadata
binaries
reports metadata
binaries
reports metadata
Artifact repository

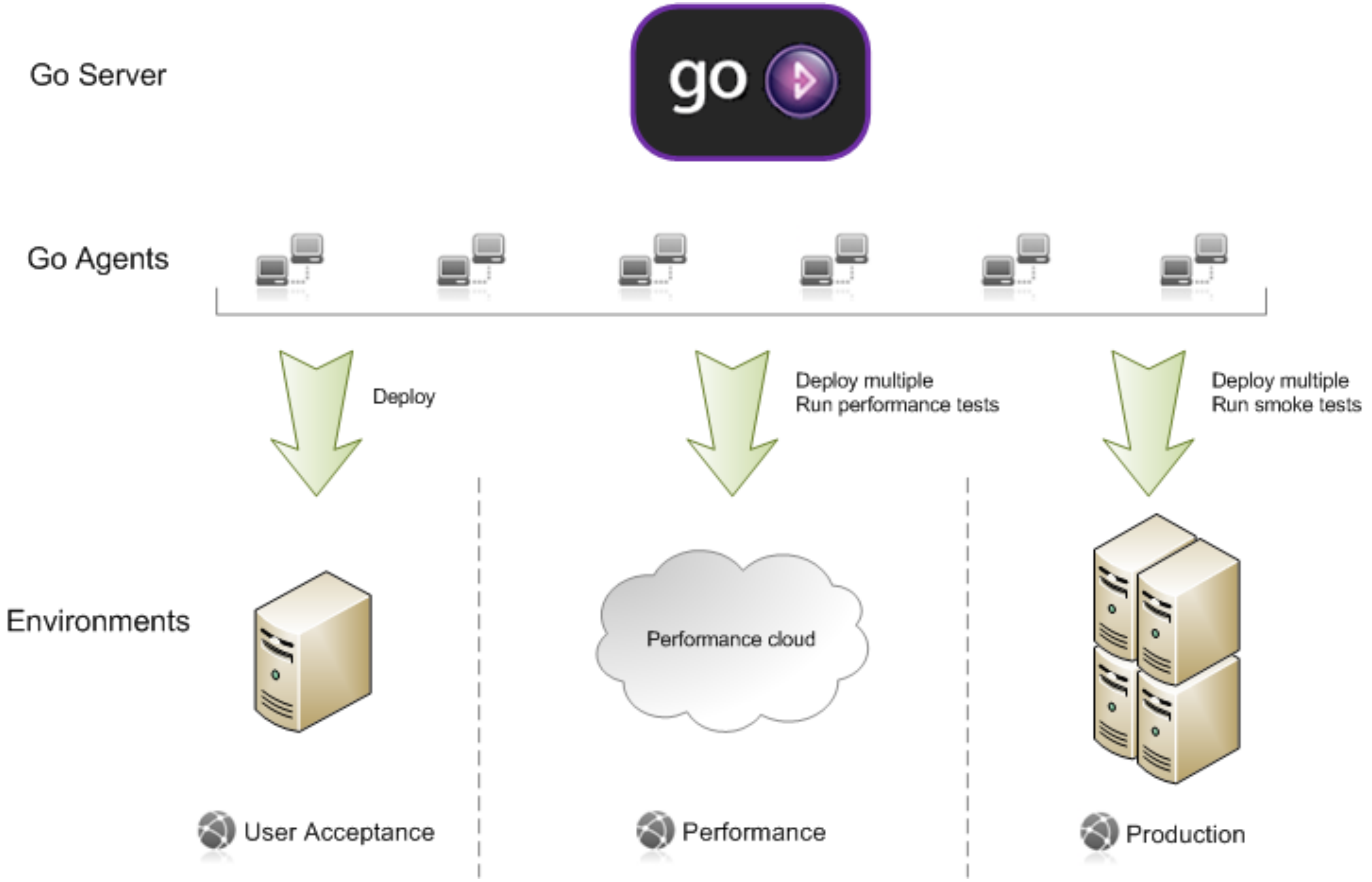
Machinery

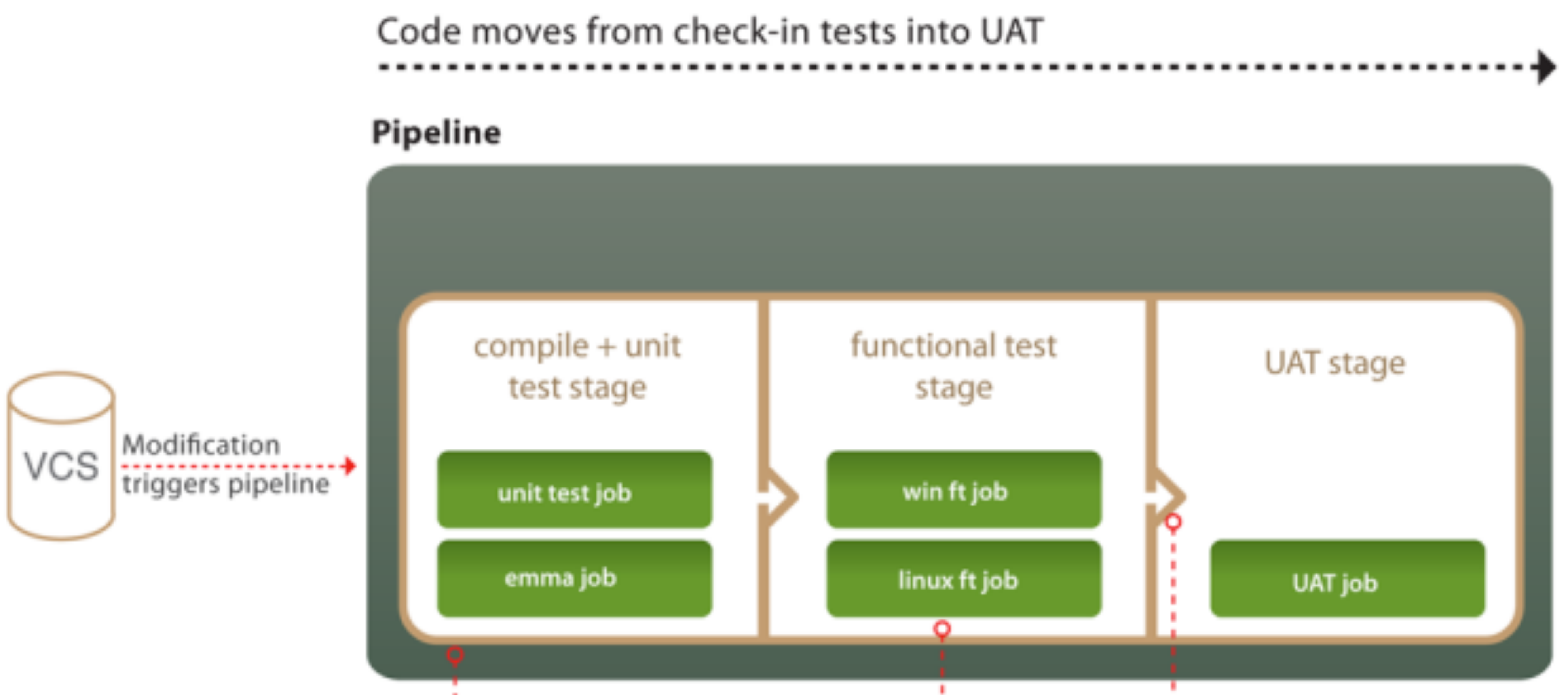
continuous integration ++

Jenkins

go™
Continuous Delivery

www.thoughtworks.com/products/go-continuous-delivery


Go Server
go
Go Agents
Deploy
Deploy multiple
Run performance tests
Deploy multiple
Run smoke tests
Environments
Performance cloud
User Acceptance
Performance
Production


Code moves from check-in tests into UAT
Pipeline
VCS
Modification triggers pipeline
compile + unit test stage
unit test job
emma job
functional test stage
win ft job
linux ft job
UAT stage
UAT job

Approval

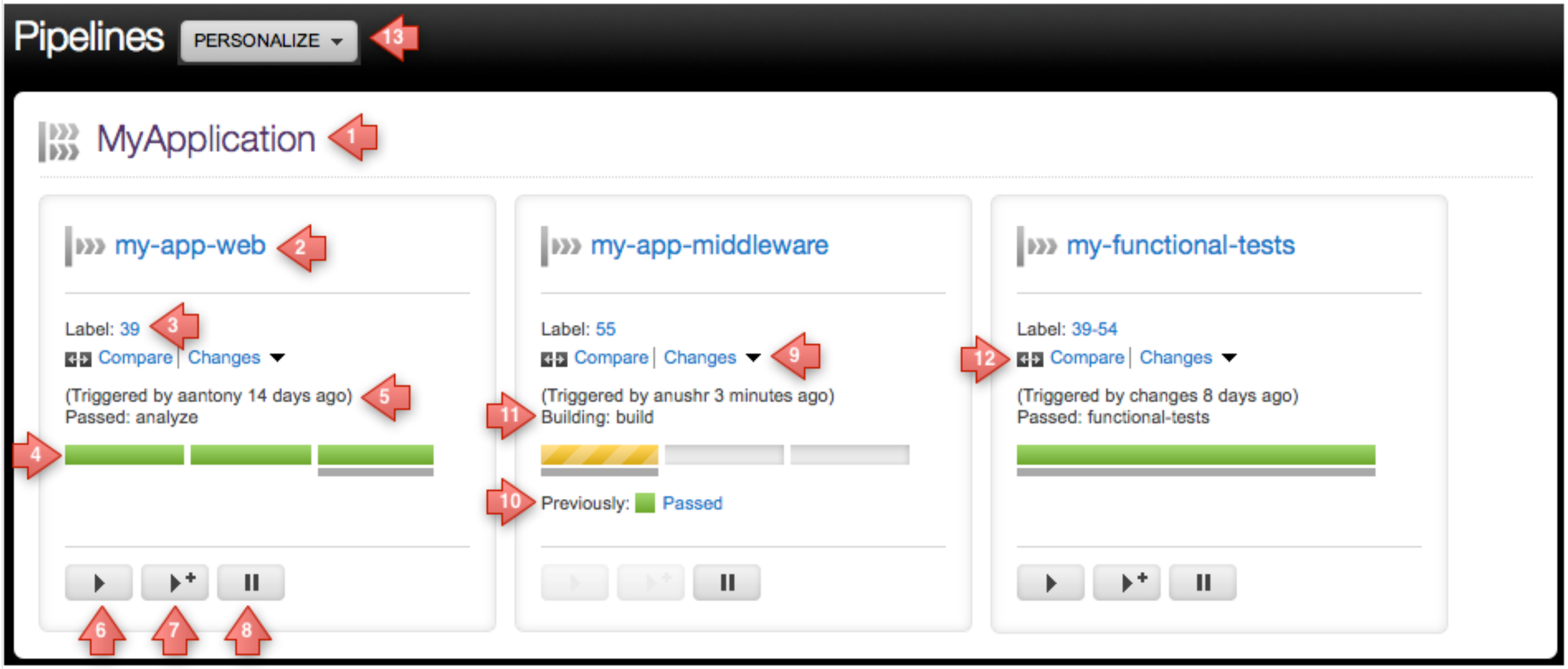
- Can be automatic or manual

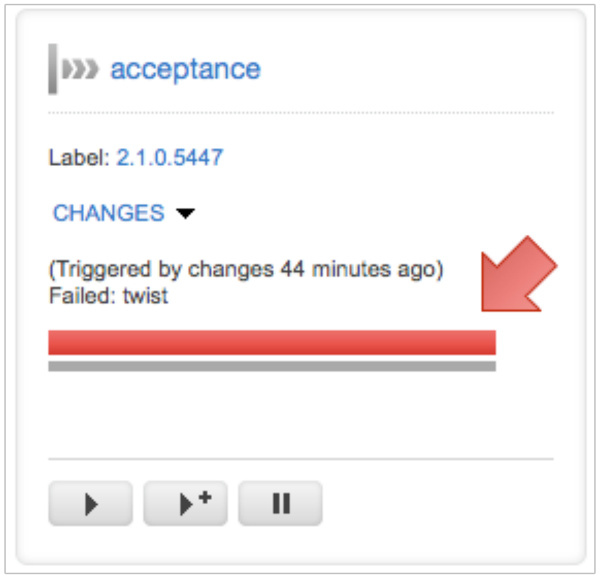
Stage

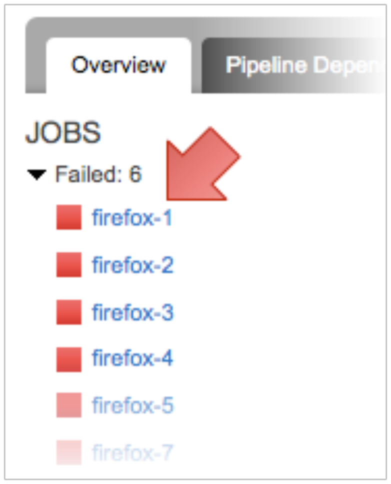
- Stages run consecutively
- Each stage is triggered automatically by the successful completion of the previous stage
- Can also be triggered manually

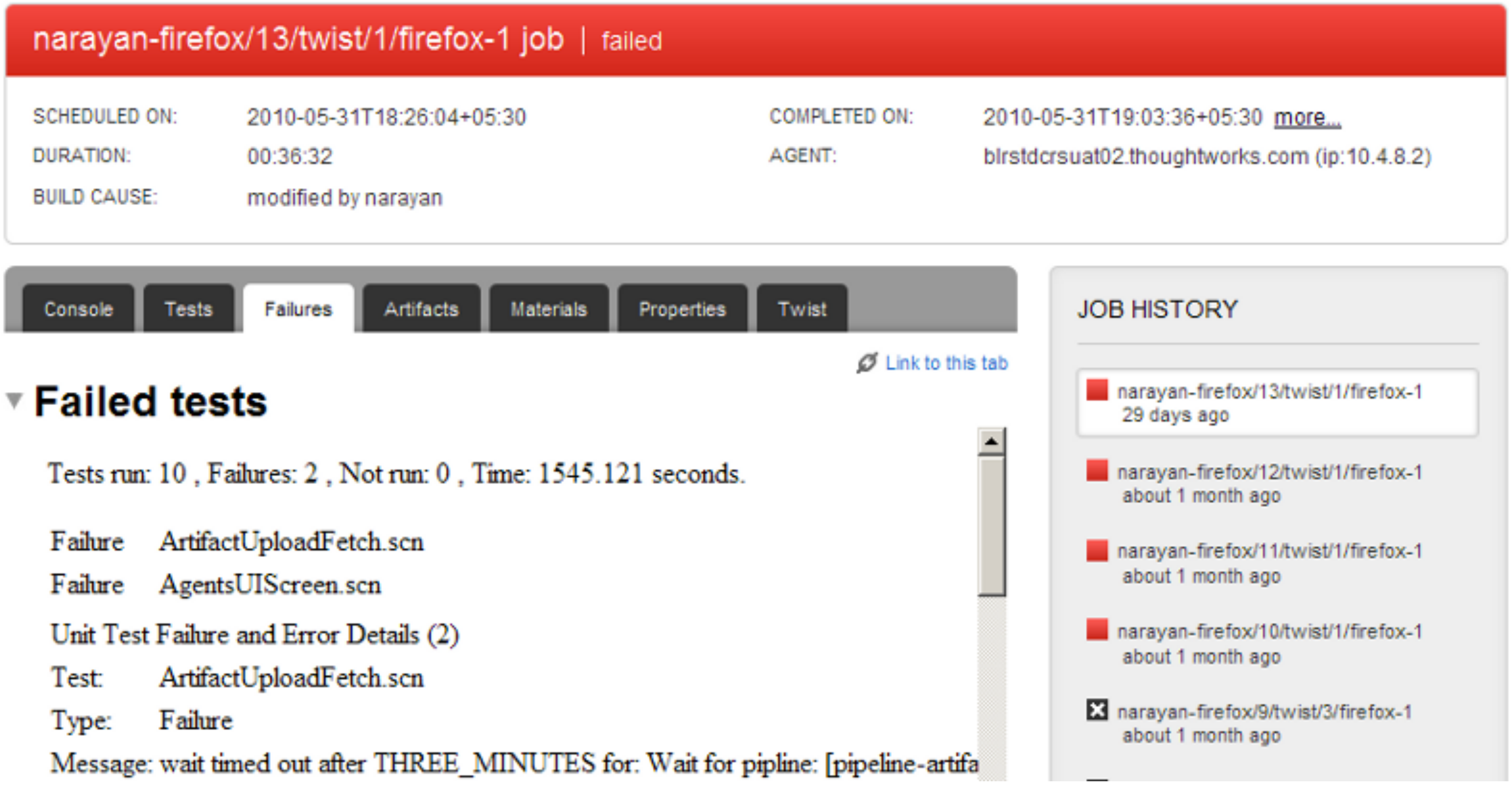
Job

- Jobs run concurrently within a stage
- If a job fails, the stage it is in fails
- Each job plan runs one or more targets (ant, nant, or ecec)
- Jobs can run in Parallel within a stage if you have multiple agents


Pipelines
PERSONALIZE
13
MyApplication
1
my-app-web
2
Label: 39
3
Compare | Changes
(Triggered by aantony 14 days ago)
Passed: analyze
5
4
6
7
8
my-app-middleware
Label: 55
Compare | Changes
9
(Triggered by anushr 3 minutes ago)
Building: build
11
Previously: Passed
10
my-functional-tests
Label: 39-54
12
Compare | Changes
(Triggered by changes 8 days ago)
Passed: functional-tests







narayan-firefox/13/twist/1/firefox-1 job | failed

SCHEDULED ON:	2010-05-31T18:26:04+05:30	COMPLETED ON:	2010-05-31T19:03:36+05:30 more...
DURATION:	00:36:32	AGENT:	blrstdcrsuat02.thoughtworks.com (ip:10.4.8.2)
BUILD CAUSE:	modified by narayan		

Failed tests

Tests run: 10 , Failures: 2 , Not run: 0 , Time: 1545.121 seconds.

Failure	ArtifactUploadFetch.scn
Failure	AgentsUIScreen.scn

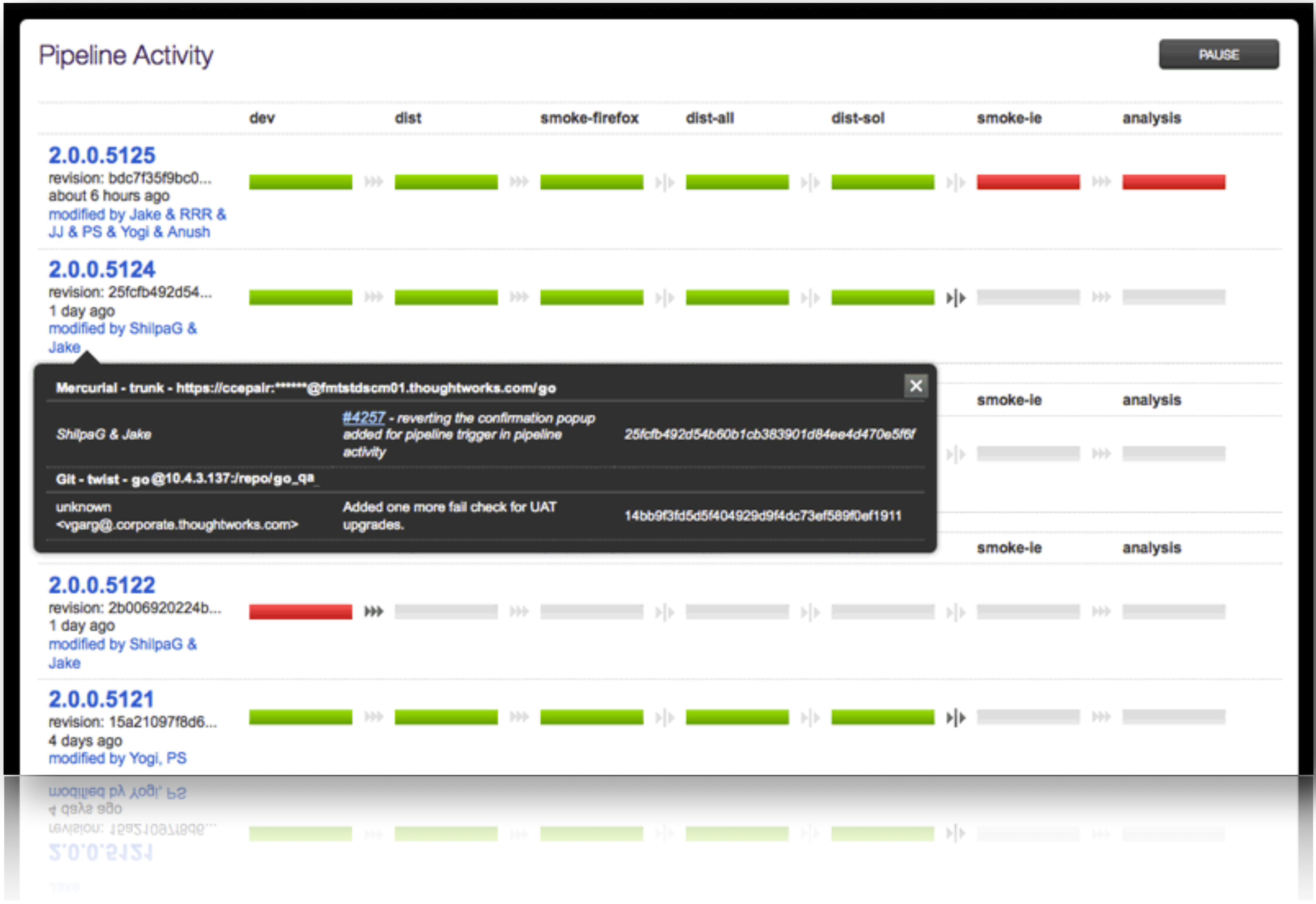
Unit Test Failure and Error Details (2)

Test:	ArtifactUploadFetch.scn
Type:	Failure

Message: wait timed out after THREE_MINUTES for: Wait for pipline: [pipeline-artifa

JOB HISTORY

- narayan-firefox/13/twist/1/firefox-1 29 days ago
- narayan-firefox/12/twist/1/firefox-1 about 1 month ago
- narayan-firefox/11/twist/1/firefox-1 about 1 month ago
- narayan-firefox/10/twist/1/firefox-1 about 1 month ago
- narayan-firefox/9/twist/3/firefox-1 about 1 month ago



Pipeline Activity

PAUSE

	dev	dist	smoke-firefox	dist-all	dist-sol	smoke-ie	analysis
2.0.0.5125 revision: bdc7f35f9bc0... about 6 hours ago modified by Jake & RRR & JJ & PS & Yogi & Anush	🟩	🟩	🟩	🟩	🟩	🟥	🟥
2.0.0.5124 revision: 25fcfb492d54... 1 day ago modified by ShilpaG & Jake	🟩	🟩	🟩	🟩	🟩	⬜	⬜
2.0.0.5122 revision: 2b006920224b... 1 day ago modified by ShilpaG & Jake	🟥	⬜	⬜	⬜	⬜	⬜	⬜
2.0.0.5121 revision: 15a21097f8d6... 4 days ago modified by Yogi, PS	🟩	🟩	🟩	🟩	🟩	⬜	⬜

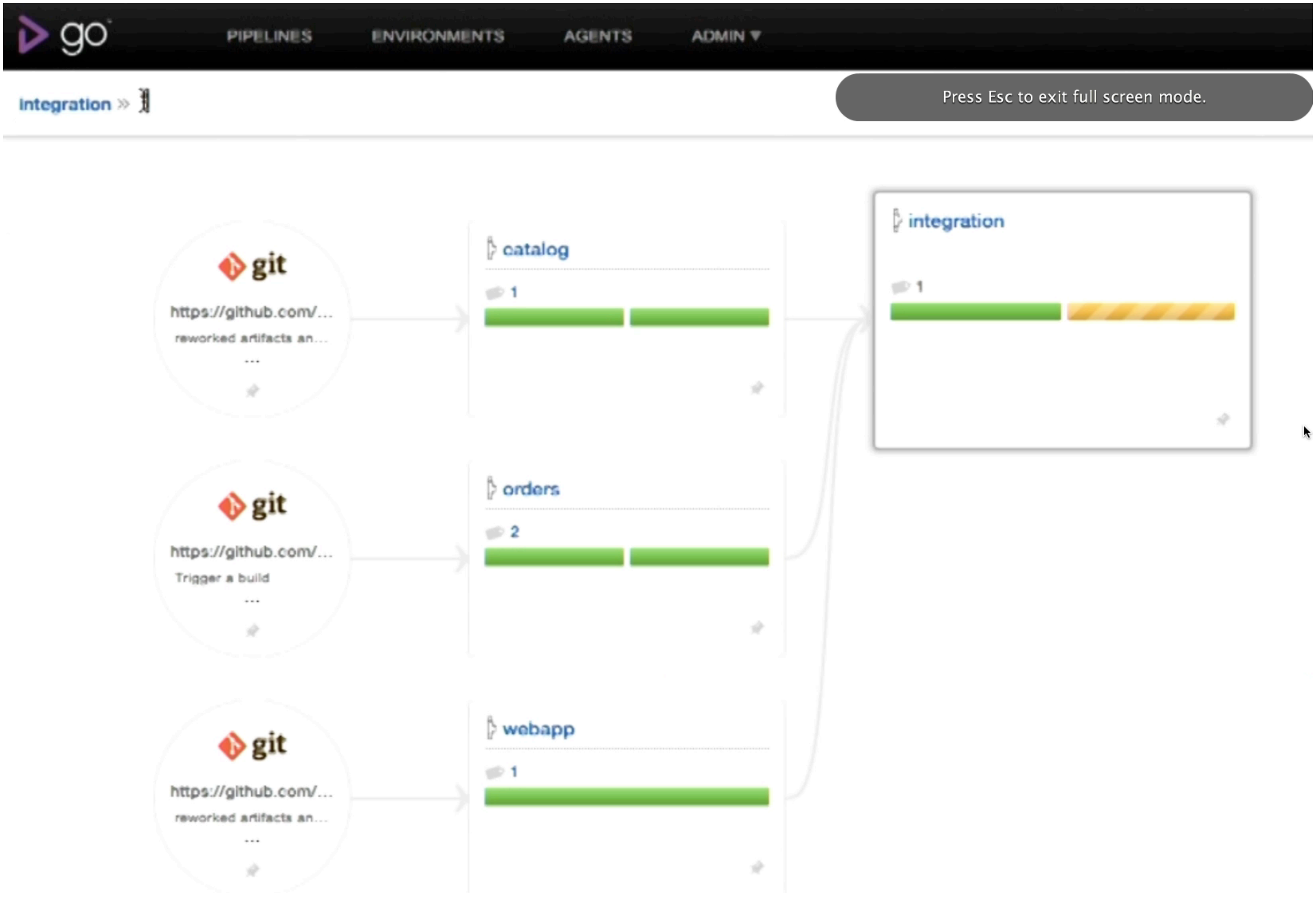
Mercurial - trunk - https://ccepair:****@fmtstdscm01.thoughtworks.com/go**

ShilpaG & Jake	*#4257 - reverting the confirmation popup added for pipeline trigger in pipeline activity*	*25fcfb492d54b60b1cb383901d84ee4d470e5f6f*

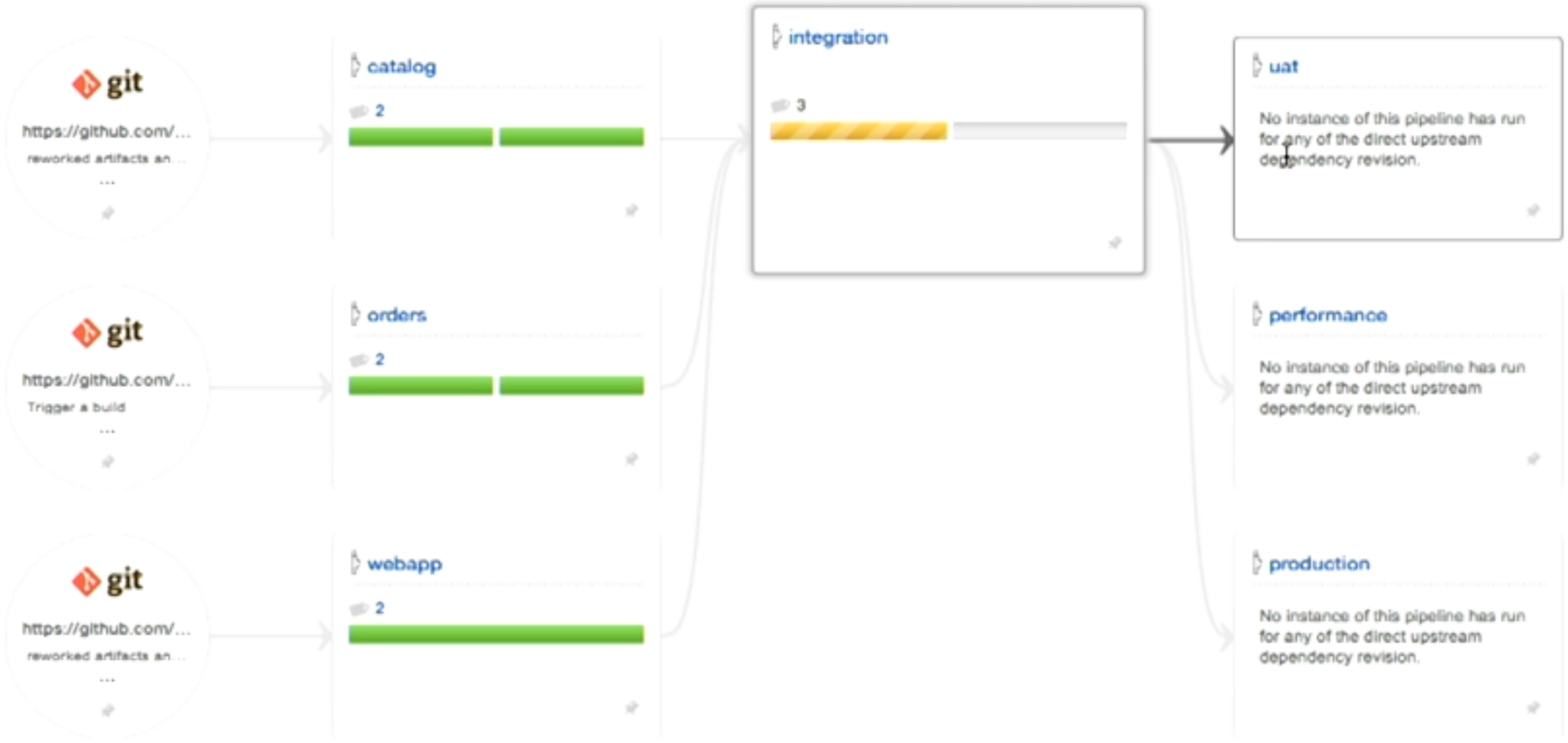
Git - twist - go@10.4.3.137:/repo/go_qa

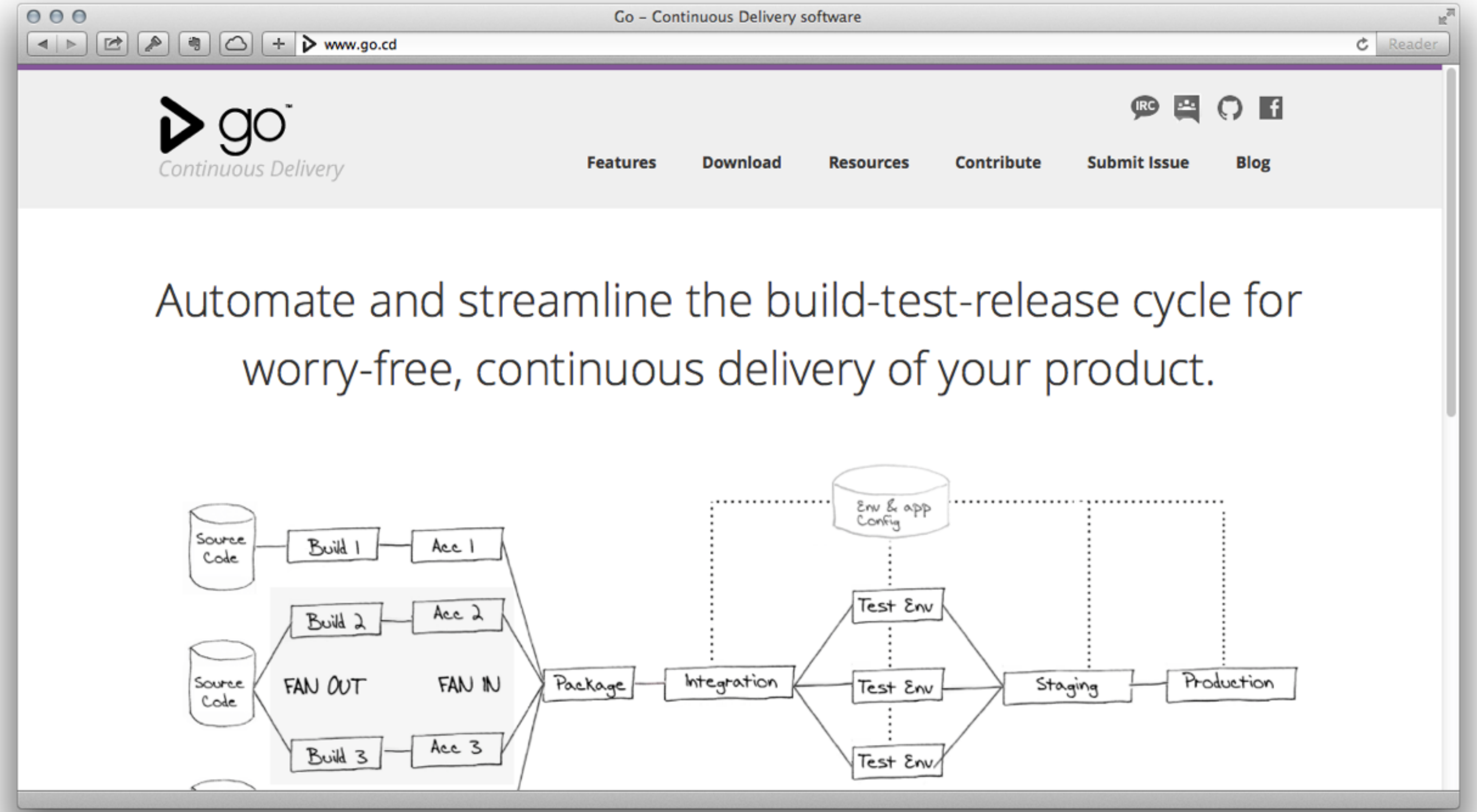
unknown <vgarg@.corporate.thoughtworks.com>	Added one more fail check for UAT upgrades.	14bb9f3fd5d5f404929d9f4dc73ef589f0ef1911

Integration Pipeline in Go CD



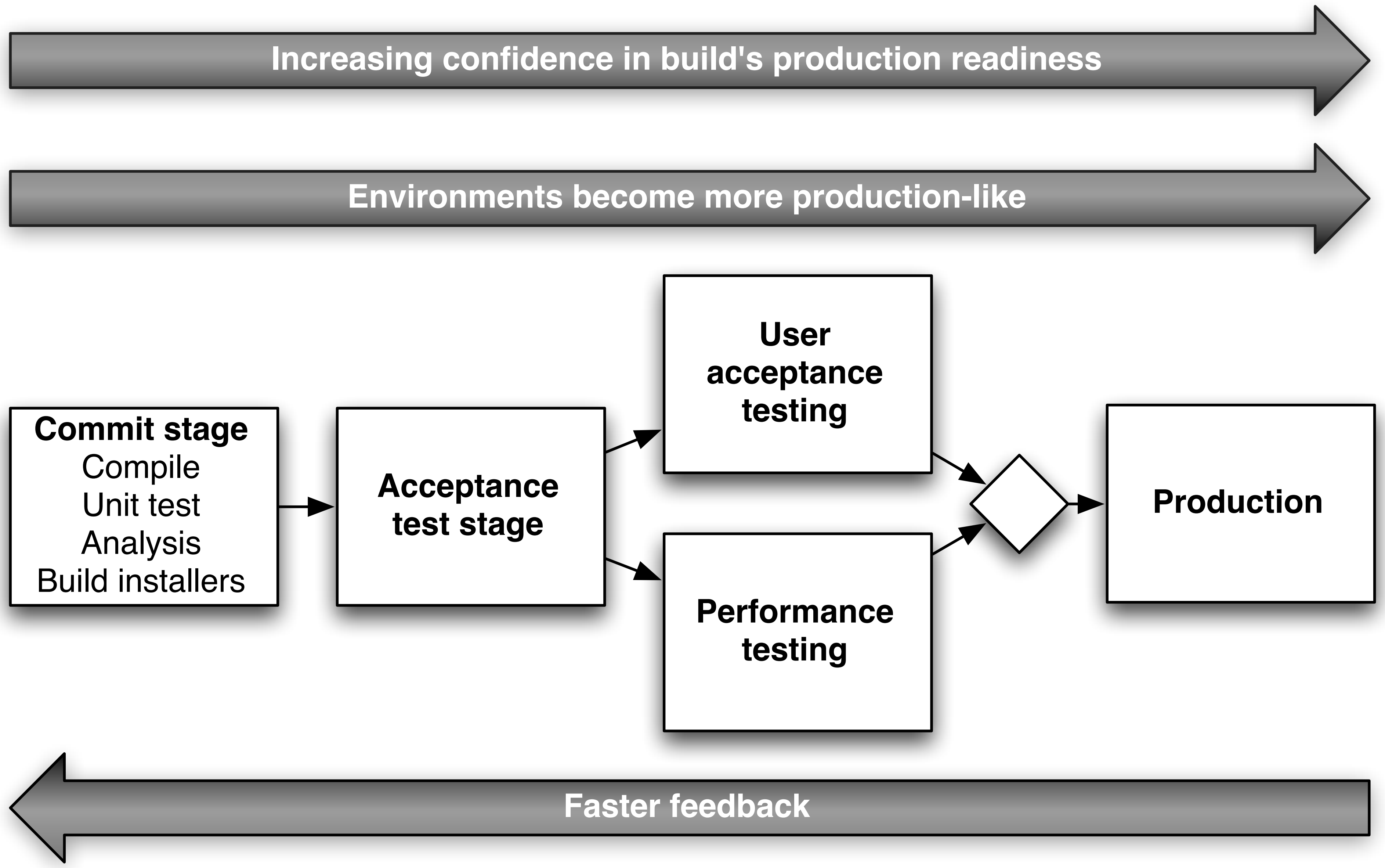
Integration Pipeline in Go CD


git
https://github.com/...
reworked artifacts an...
catalog
2
integration
3
uat
No instance of this pipeline has run for any of the direct upstream dependency revision.
git
https://github.com/...
Trigger a build
orders
2
performance
No instance of this pipeline has run for any of the direct upstream dependency revision.
git
https://github.com/...
reworked artifacts an...
webapp
2
production
No instance of this pipeline has run for any of the direct upstream dependency revision.



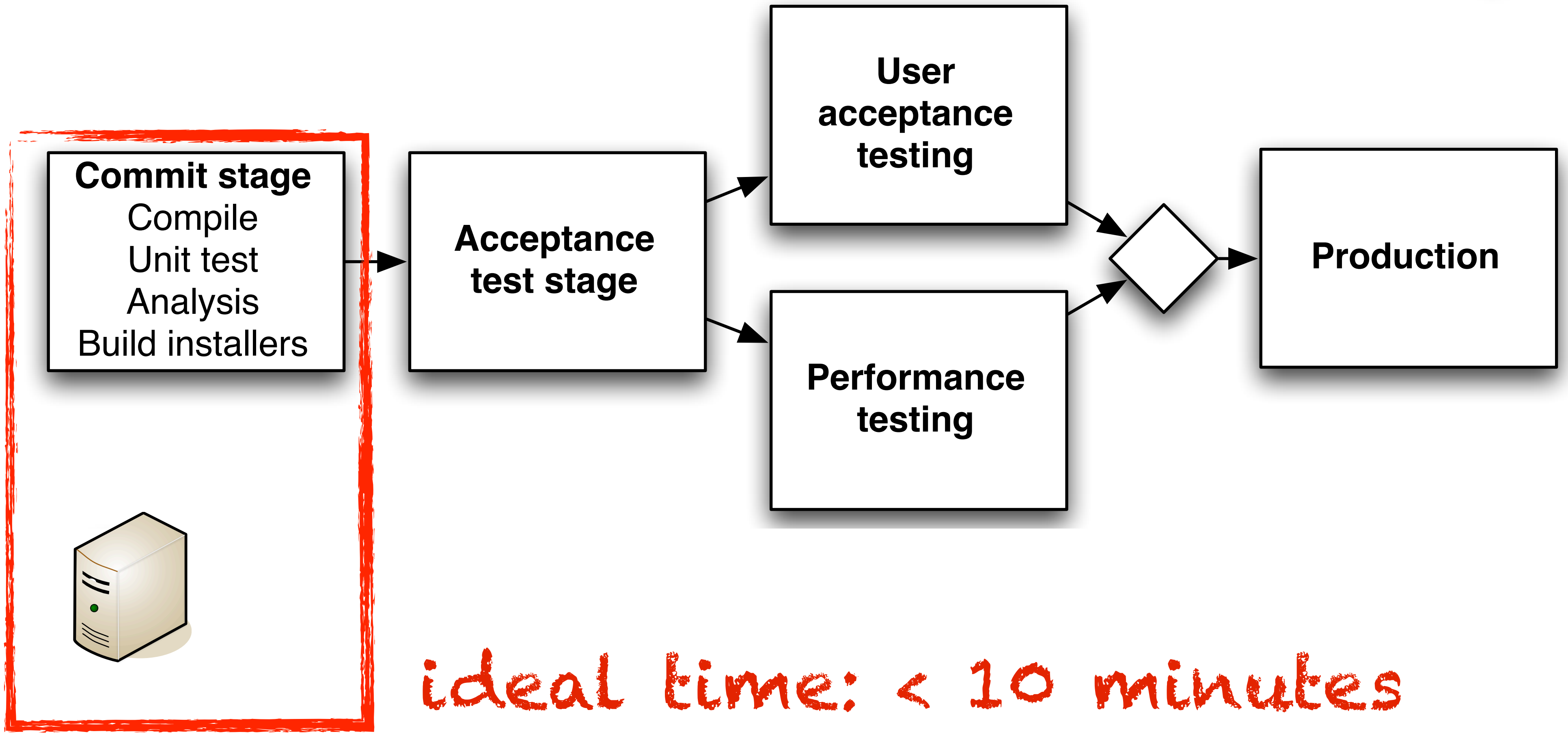
Automate and streamline the build-test-release cycle for worry-free, continuous delivery of your product.

www.go.cd


Increasing confidence in build's production readiness
Environments become more production-like
Commit stage
Compile
Unit test
Analysis
Build installers
Acceptance test stage
User acceptance testing
Performance testing
Production
Faster feedback

Pipeline Anti-patterns

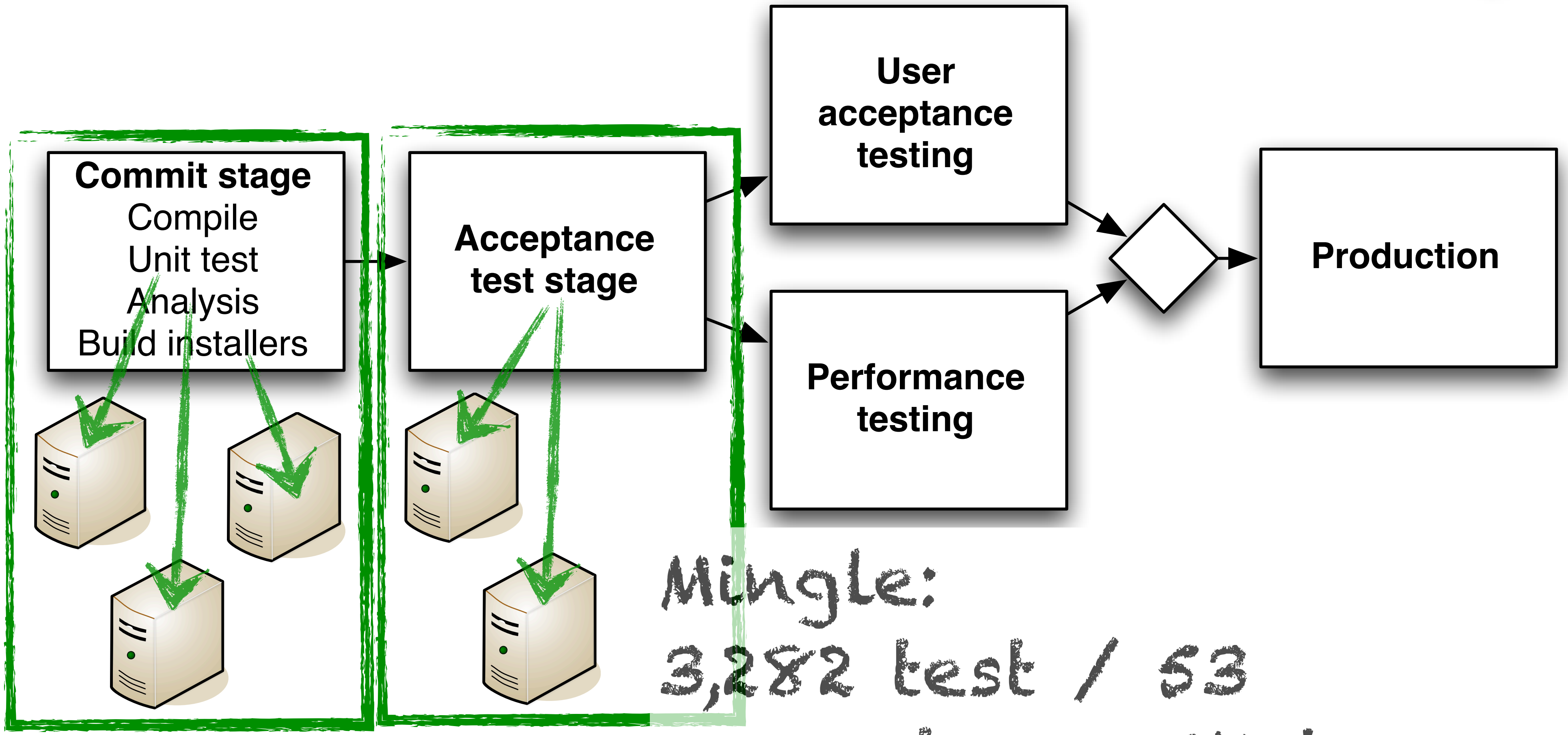
insufficient parallelization



ideal time: < 10 minutes

Pipeline Anti-patterns

insufficient parallelization



Mingle:
3,282 test / 53 computers = ~1 hour

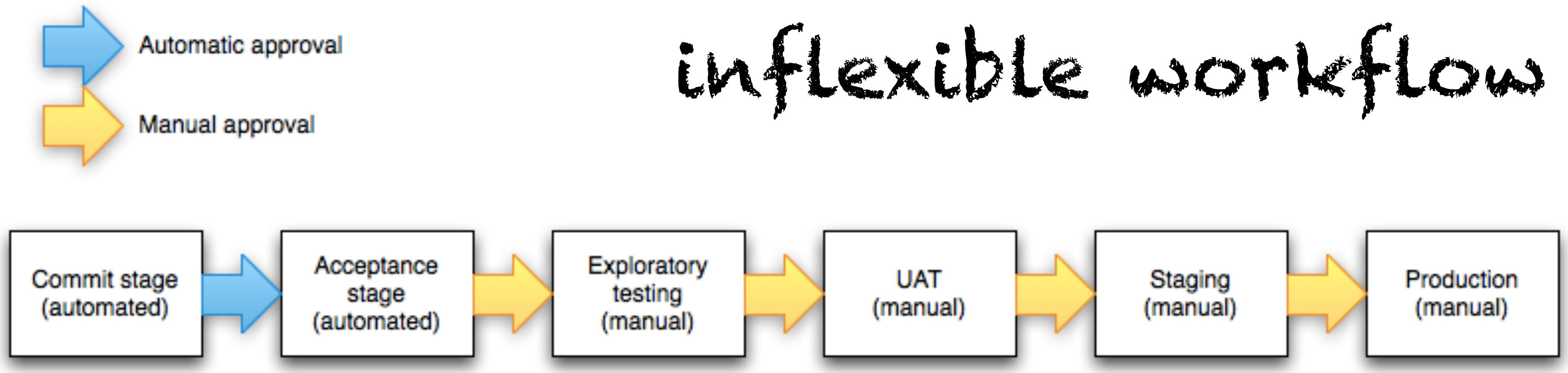
Insufficient Parallelization Heuristic:

make your pipeline *wide*, not *long*

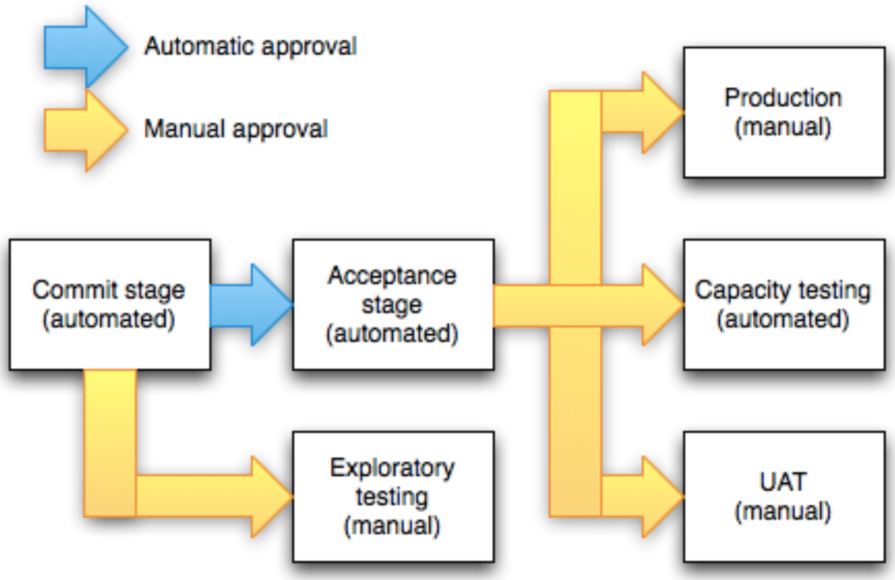
- reduce the number of stages as much as possible
- parallelize each stage as much as you can

create more stages if necessary to optimize feedback

Pipeline Anti-patterns

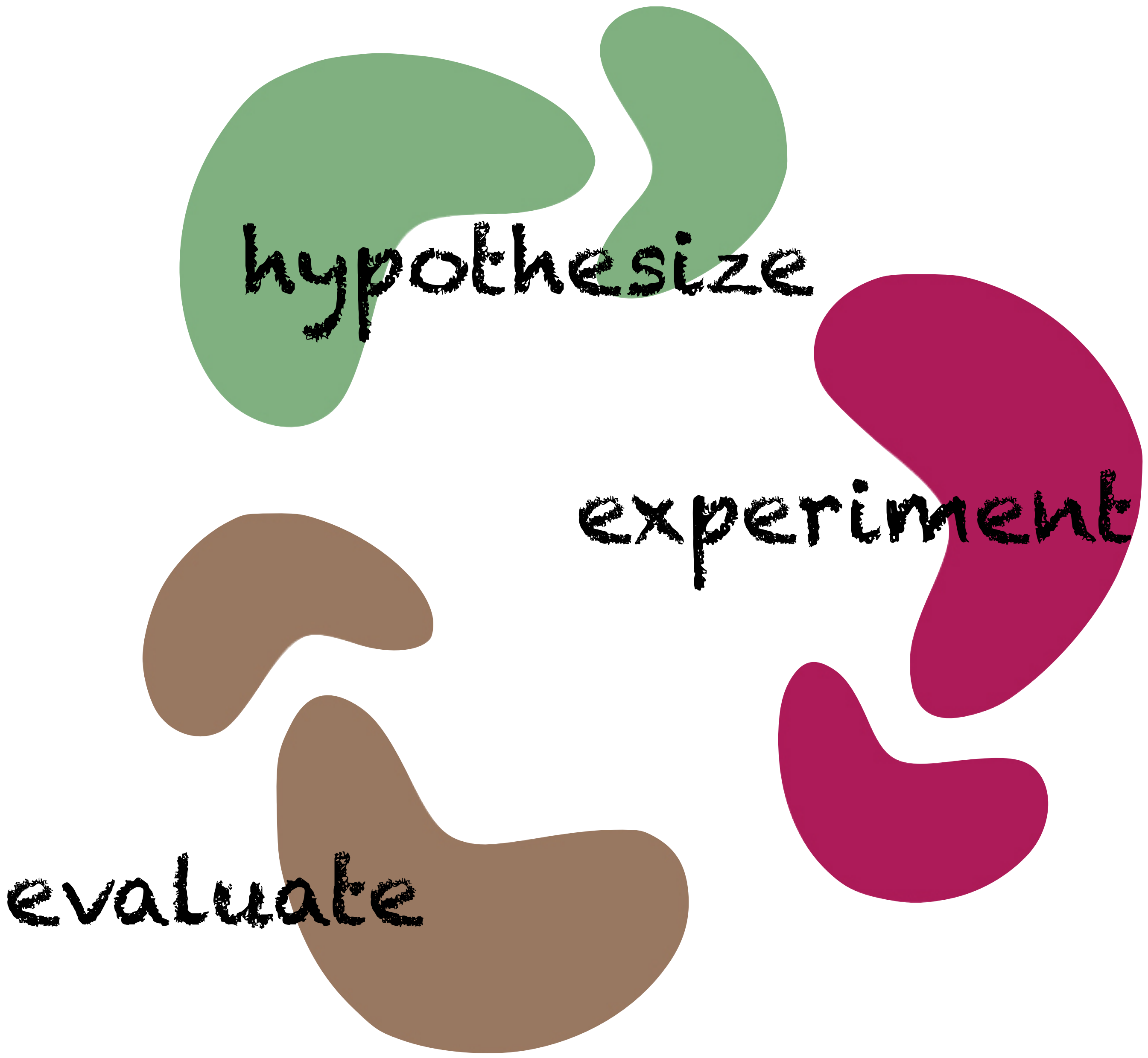


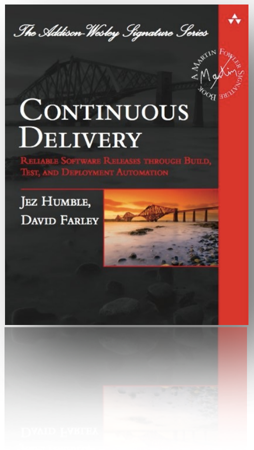
inflexible workflow



pipeline fans out as soon as it makes sense to do so

The Scientific Method



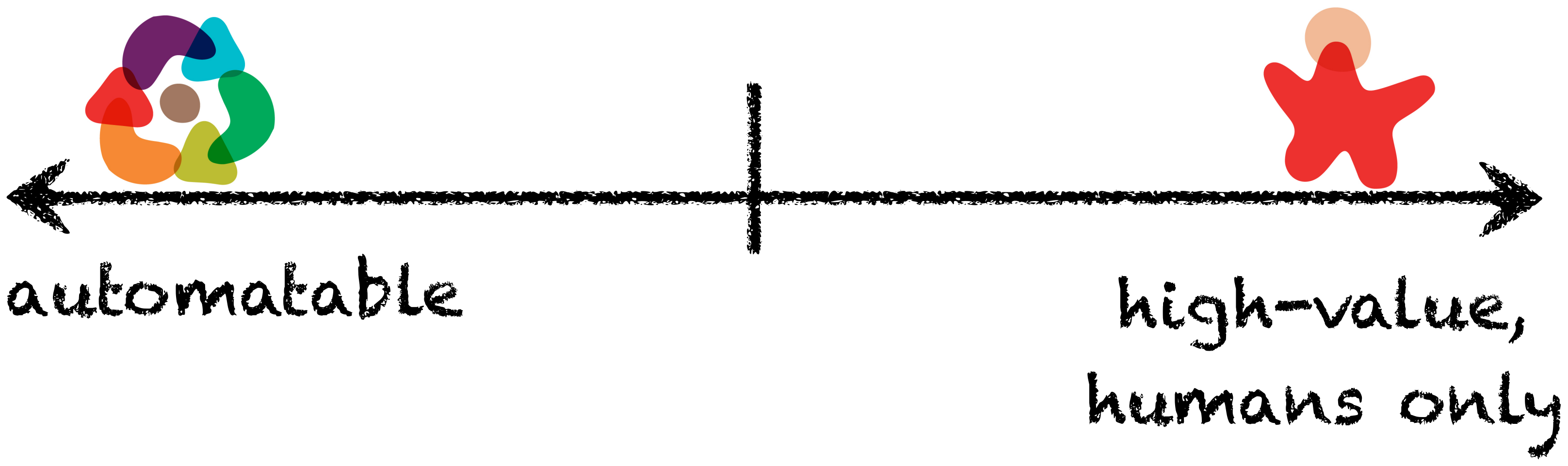

The Addison-Wesley Signature Series
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble, David Farley

Principles

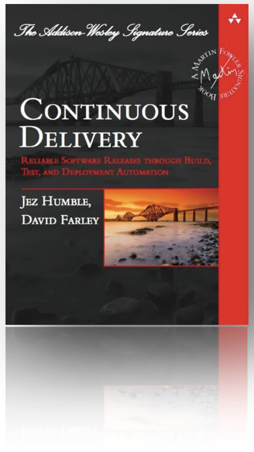
automate almost everything

build, deploy, test, release

manual testing, approvals



Principles

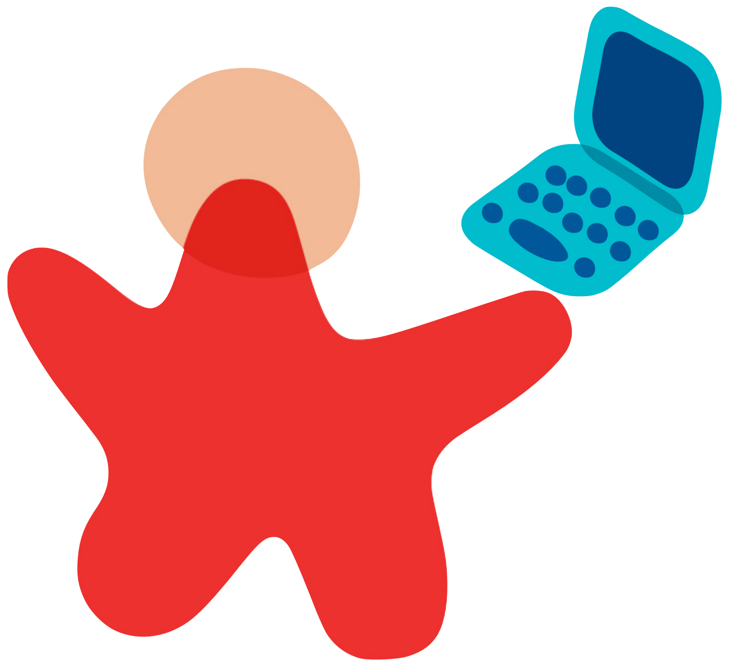

The Addison-Wesley Signature Series
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble, David Farley

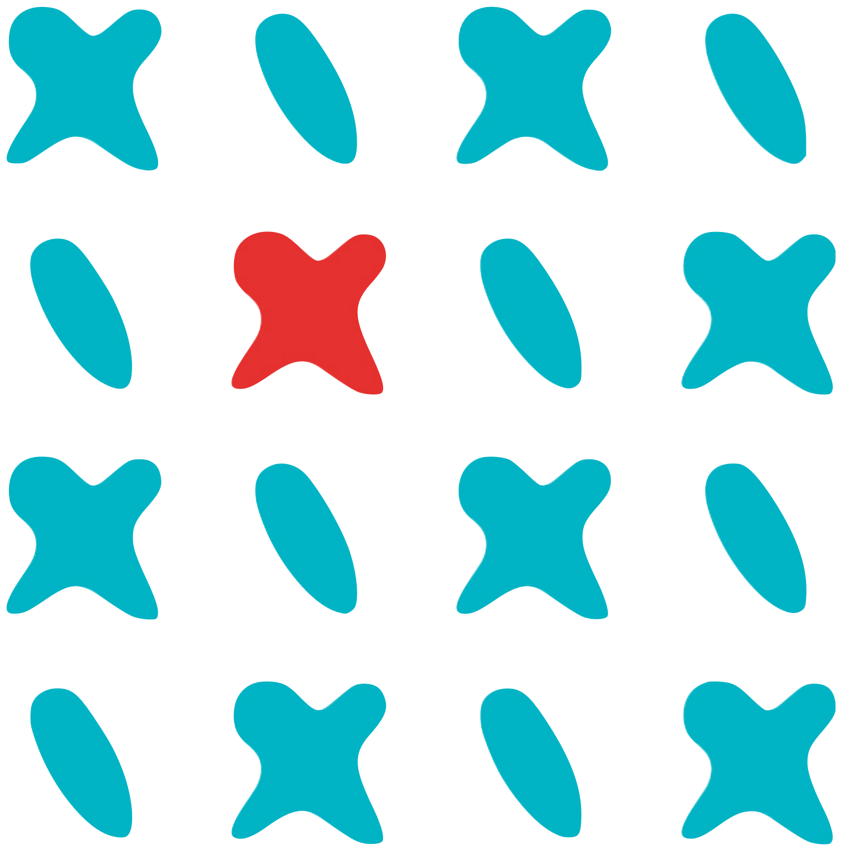
keep everything you need to build, deploy, test, & release in version control

- requirements documents
- test scripts
- automated test cases
- network configuration scripts
- technical documentation

- database creation, upgrade, downgrade, and initialization scripts
- application stack configuration scripts
- libraries
- deployment scripts
- tool chains

When You Hire a New Developer...



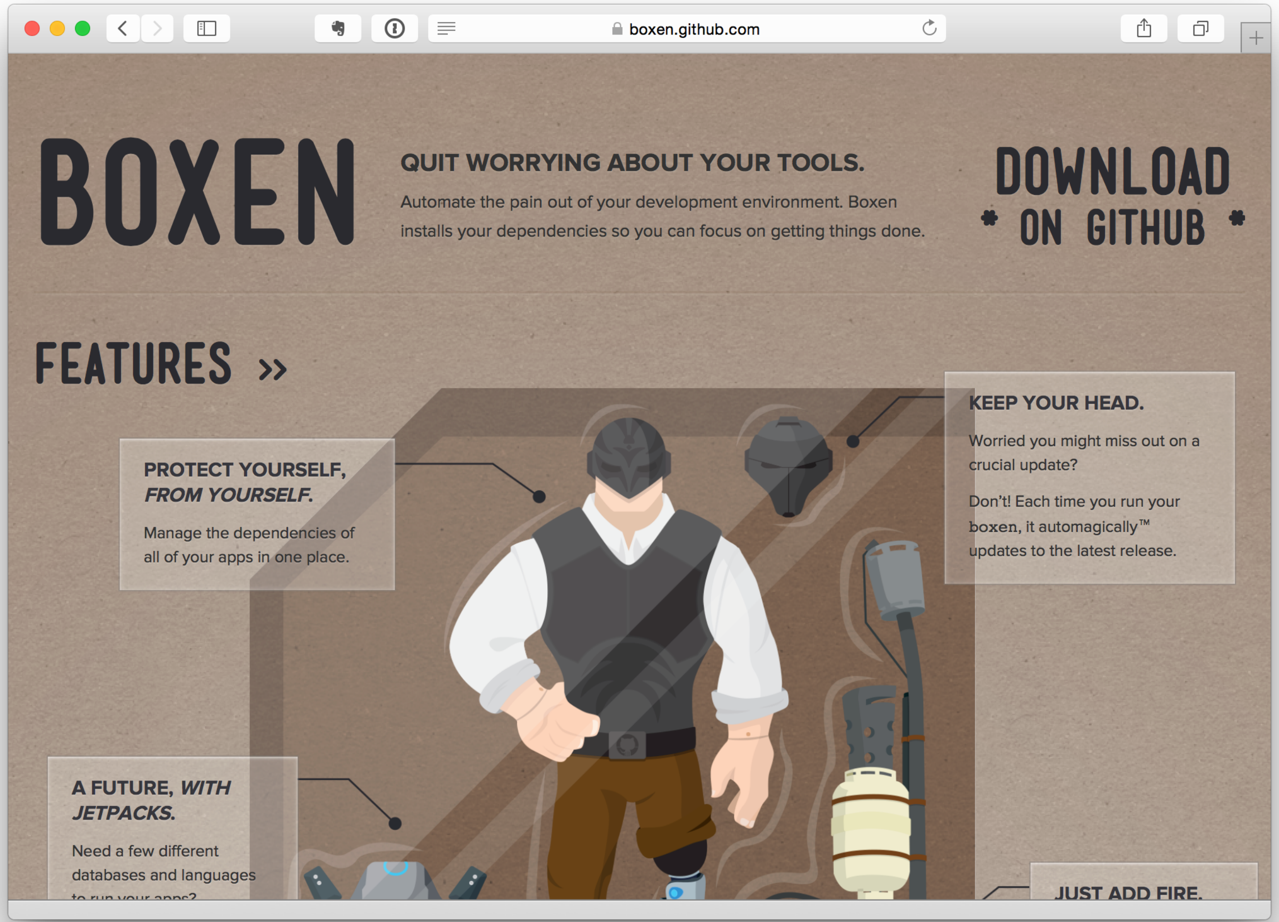


Infrastructure Consistency

Chef

puppet labs

ANSIBLE


boxen.github.com

BOXEN

QUIT WORRYING ABOUT YOUR TOOLS.

Automate the pain out of your development environment. Boxen installs your dependencies so you can focus on getting things done.

DOWNLOAD ON GITHUB

FEATURES »

PROTECT YOURSELF, *FROM YOURSELF.*

Manage the dependencies of all of your apps in one place.

KEEP YOUR HEAD.

Worried you might miss out on a crucial update?

Don't! Each time you run your boxen, it automagically™ updates to the latest release.

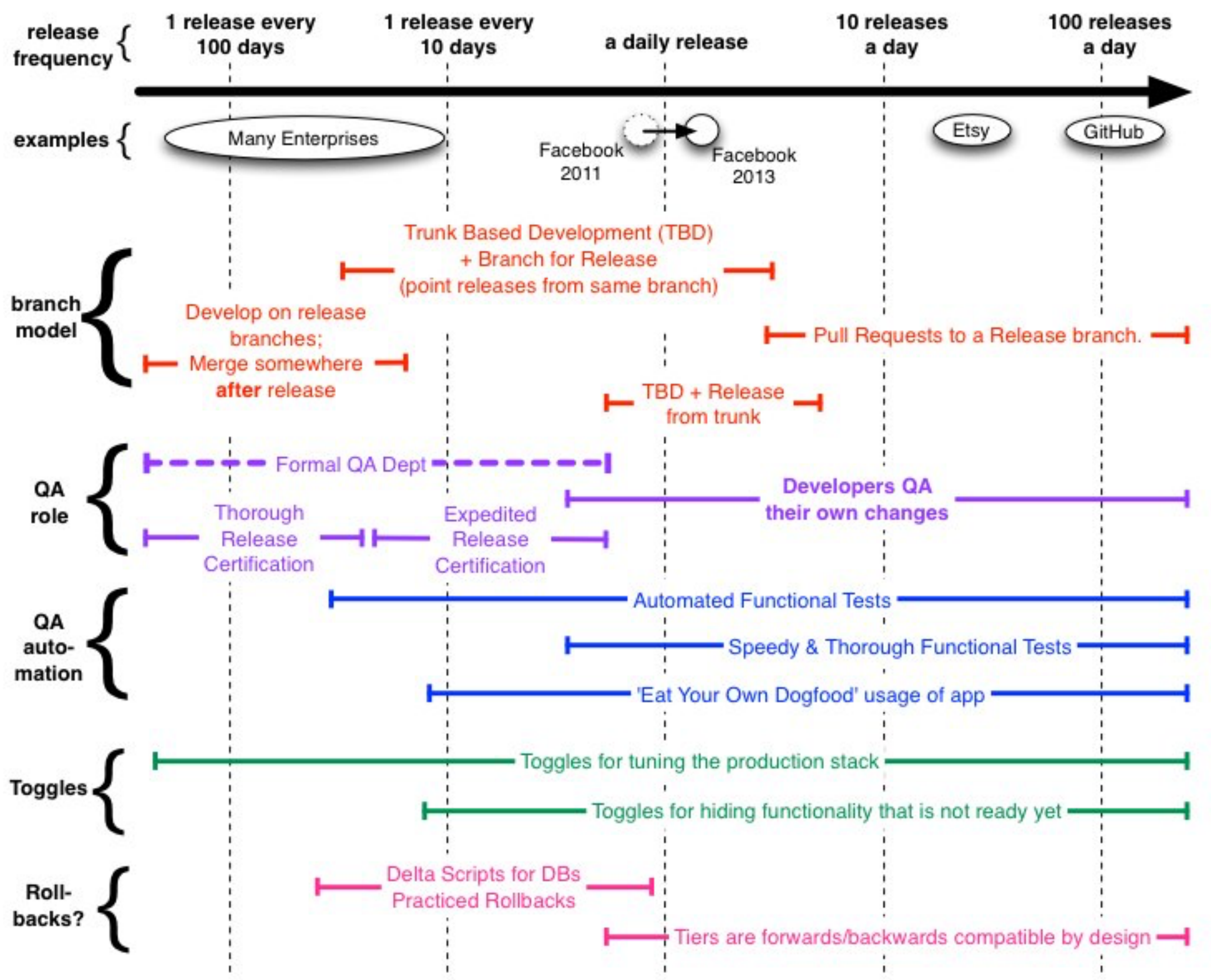
A FUTURE, *WITH JETPACKS.*

Need a few different databases and languages to run your apps?

JUST ADD FIRE.

boxen.github.com

Continuous Delivery Maturity Model


release frequency
1 release every 100 days
1 release every 10 days
a daily release
10 releases a day
100 releases a day
examples
Many Enterprises
Facebook 2011
Facebook 2013
Etsy
GitHub
branch model
Trunk Based Development (TBD) + Branch for Release (point releases from same branch)
Develop on release branches; Merge somewhere after release
Pull Requests to a Release branch.
TBD + Release from trunk
QA role
Formal QA Dept
Developers QA their own changes
Thorough Release Certification
Expedited Release Certification
QA auto-mation
Automated Functional Tests
Speedy & Thorough Functional Tests
'Eat Your Own Dogfood' usage of app
Toggles
Toggles for tuning the production stack
Toggles for hiding functionality that is not ready yet
Roll-backs?
Delta Scripts for DBs Practiced Rollbacks
Tiers are forwards/backwards compatible by design

http://paulhammant.com/2013/03/13/facebook-tbd-take-2/

Continuous Delivery Maturity Model

1: initial

delivery focus	characteristics	result
A few smart people performing heroics	There is an ad hoc release delivery process • Teams rely mainly on manual testing after development is complete to find defects. • System integration is painful and happens after development on a module is completed. • Provisioning production-like integrated testing environments is expensive and manual. • Deployment process is manual. • Developers, testers, operations, and management have goals that bring them into conflict. • Change management is ad hoc or heavyweight and often circumvented or ignored.	Ad hoc deployments

Continuous Delivery Maturity Model

2: managed

delivery focus	characteristics	result
Time-boxed releases (the team sets a release date and manages to it)	There is an adaptive delivery process. • Clear product ownership and chain of responsibility are in place. • Change management controls are implemented, including a process to detect unauthorized changes with consequences defined. • Business participates fully and regularly in development activities and decisions related to delivery. • There is some automated acceptance testing. • Production-like testing environments are available for projects early on. • There is some scripting to reliably and repeatedly configure environments and build packages from version control. • Teams work in iterations of one month or less and showcase integrated	Planned release: Release time box is well defined, but duration from idea inception to production release is greater than business need.

Continuous Delivery Maturity Model

3: defined

delivery focus	characteristics	result
Regular releases over a defined period with interim milestone builds	Teams build quality into release process. • Teams practice trunk-based development with continuous integration of all changes. • There are enough automated tests that critical defects are detected and prevented fast and automatically. • Provisioning of integrated testing environments is fast and mostly automated. • No work is considered done until it has passing automated unit and acceptance tests associated with it. • Testers are not primarily focused on regression testing. Database changes are versioned and scripted.	Regular release cadence: Release time box is well defined, but duration from idea inception to production release is greater than business need.

Continuous Delivery Maturity Model

4: quantitatively managed

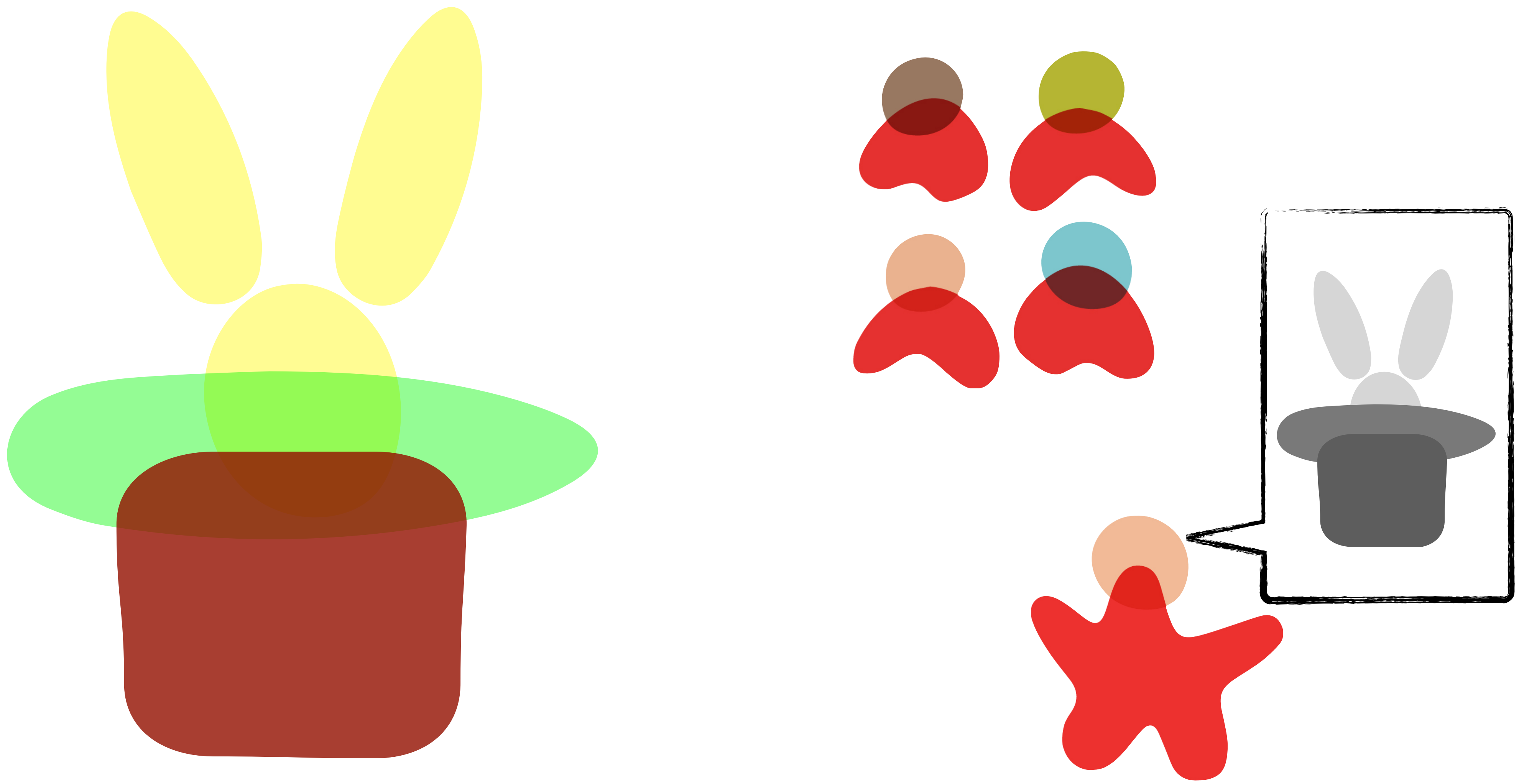
delivery focus	characteristics	result
Release on demand	Delivery teams prioritize keeping code trunk deployable over doing new work. • Deployment pipeline automatically rejects bad changes from version control. • Cross-functional end-to-end product- centric teams manage products throughout life cycle. • Comprehensive automated test suites are created through TDD/ ATDD and maintained by developers and testers working together. • Teams monitor and manage work in process and deliver work in small batches.	Release on demand: Software is always in a releasable state. Release time box is well defined and equal to, or less than, business need.

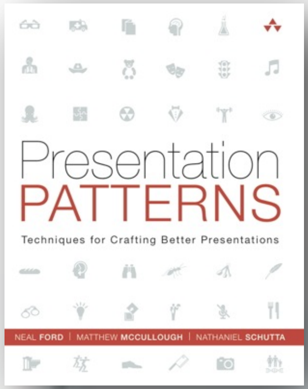
Continuous Delivery Maturity Model

5: optimizing

delivery focus	characteristics	result
Hypothesis-driven delivery	Teams focus on optimizing cycle time to learn from customers. • All new requirements describe how the value of the feature will be measured. • Product teams are responsible for implementing metrics to gather this data through techniques such as A/B testing. • Systems are architected with continuous deployment in mind, supporting patterns such as dark launching to decouple deployment from release. • Database changes are decoupled from application deployments.	Continuous deployment capability enables business innovation/ experimentation

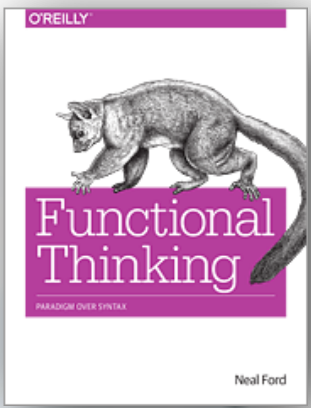
Demonstration trumps discussion.




Presentation PATTERNS

Techniques for Crafting Better Presentations

NEAL FORD | MATTHEW MCCULLOUGH | NATHANIEL SCHUTTA


O'REILLY

Functional Thinking

PARADIGM OVER SYNTAX

Neal Ford

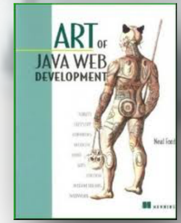
nealford.com

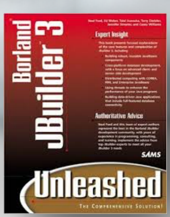
@neal4d

ThoughtWorks

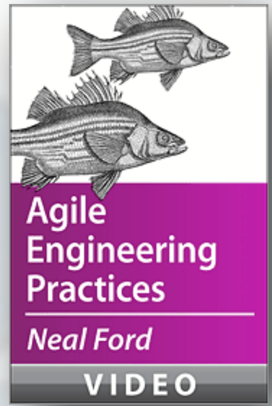
NEAL FORD

Director / Software Architect / Meme Wrangler


ART OF JAVA WEB DEVELOPMENT

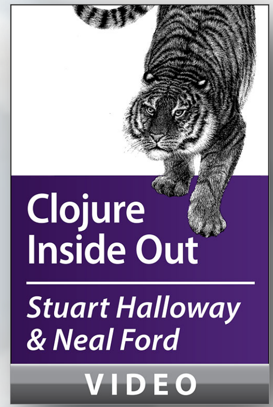

Borland JBuilder 3 Unleashed

Developing with Delphi


Agile Engineering Practices

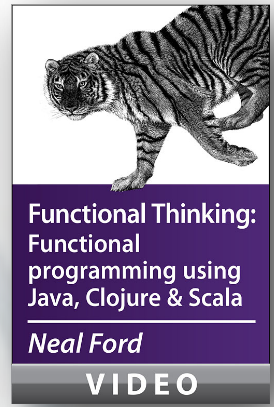
Neal Ford

VIDEO


Clojure Inside Out

Stuart Halloway & Neal Ford

VIDEO

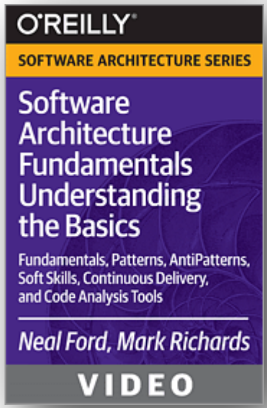

Functional Thinking: Functional programming using Java, Clojure & Scala

Neal Ford

VIDEO

O'REILLY®

SOFTWARE ARCHITECTURE SERIES


O'REILLY

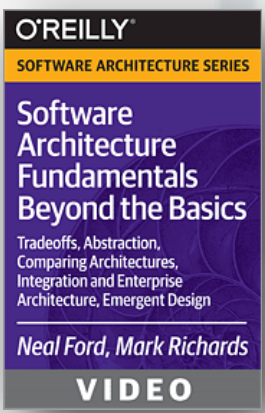
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Understanding the Basics

Fundamentals, Patterns, AntiPatterns, Soft Skills, Continuous Delivery, and Code Analysis Tools

Neal Ford, Mark Richards

VIDEO


O'REILLY

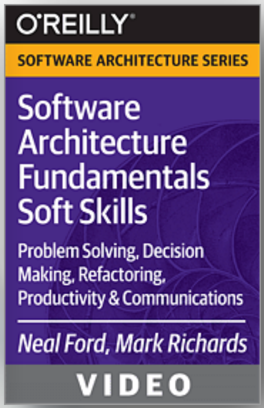
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Beyond the Basics

Tradeoffs, Abstraction, Comparing Architectures, Integration and Enterprise Architecture, Emergent Design

Neal Ford, Mark Richards

VIDEO


O'REILLY

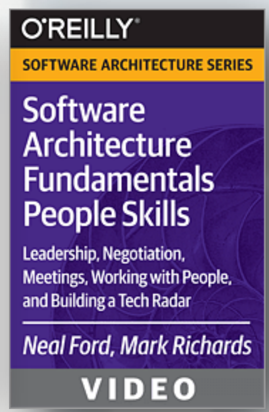
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Soft Skills

Problem Solving, Decision Making, Refactoring, Productivity & Communications

Neal Ford, Mark Richards

VIDEO


O'REILLY

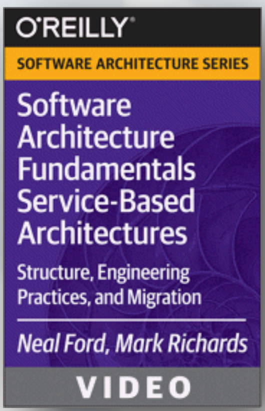
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals People Skills

Leadership, Negotiation, Meetings, Working with People, and Building a Tech Radar

Neal Ford, Mark Richards

VIDEO


O'REILLY

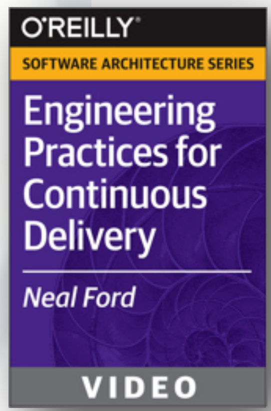
SOFTWARE ARCHITECTURE SERIES

Software Architecture Fundamentals Service-Based Architectures

Structure, Engineering Practices, and Migration

Neal Ford, Mark Richards

VIDEO


O'REILLY

SOFTWARE ARCHITECTURE SERIES

Engineering Practices for Continuous Delivery

Neal Ford

VIDEO