SUCCESFUL TESTING
THE CONTINUOUS DELIVERY PROCESS

@huibschoots & @mieldonkers

INTRODUCTION

Huib Schoots

Tester

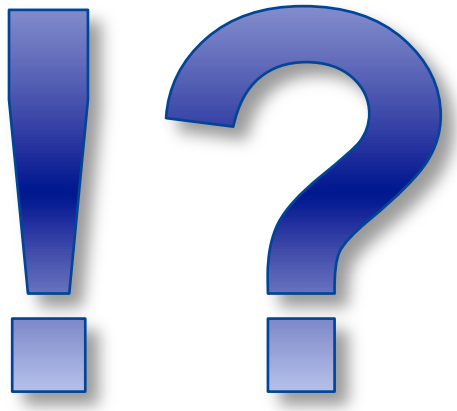
@huibschoots

Miel Donkers

Developer

@mieldonkers

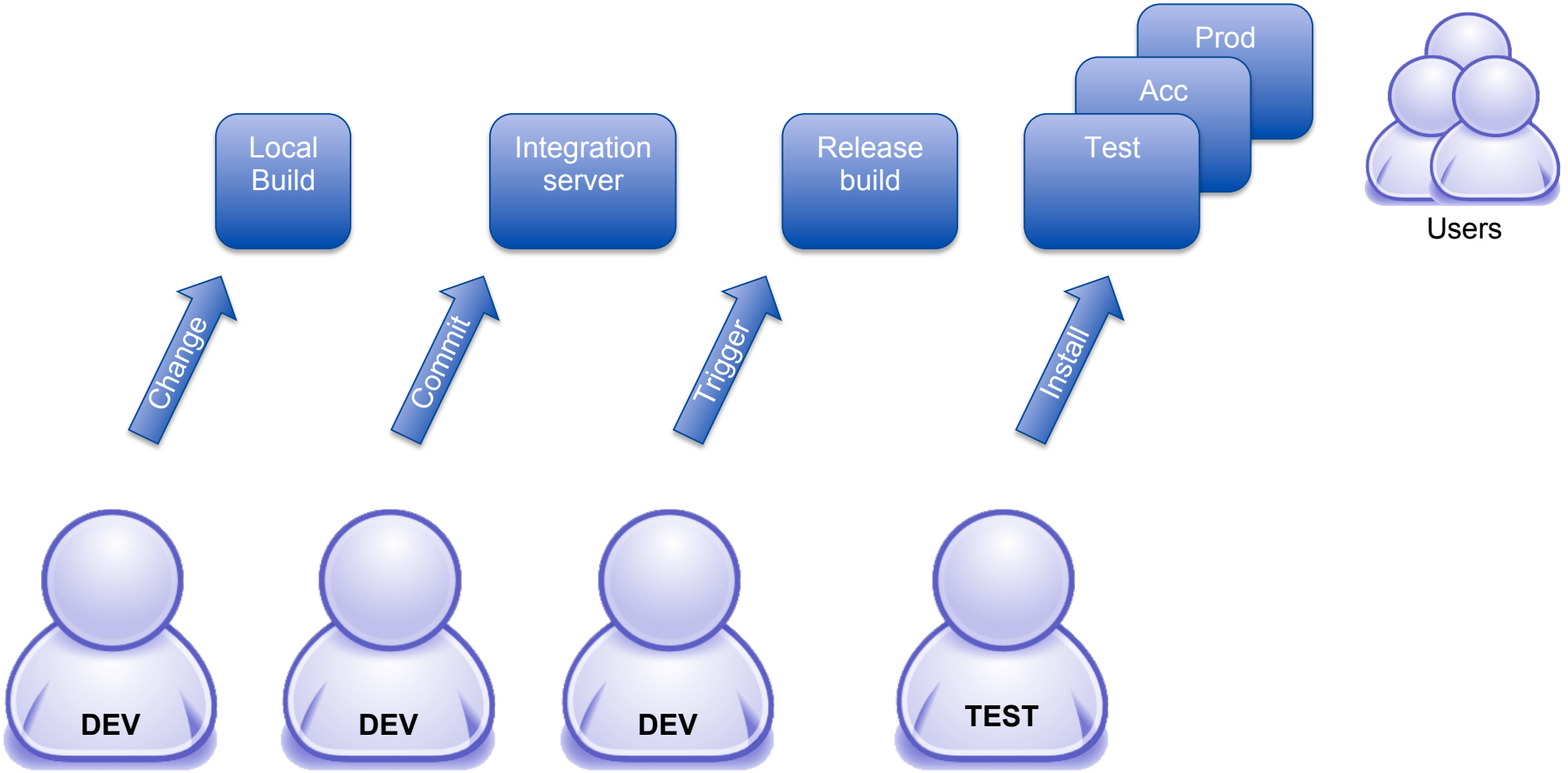
Typical



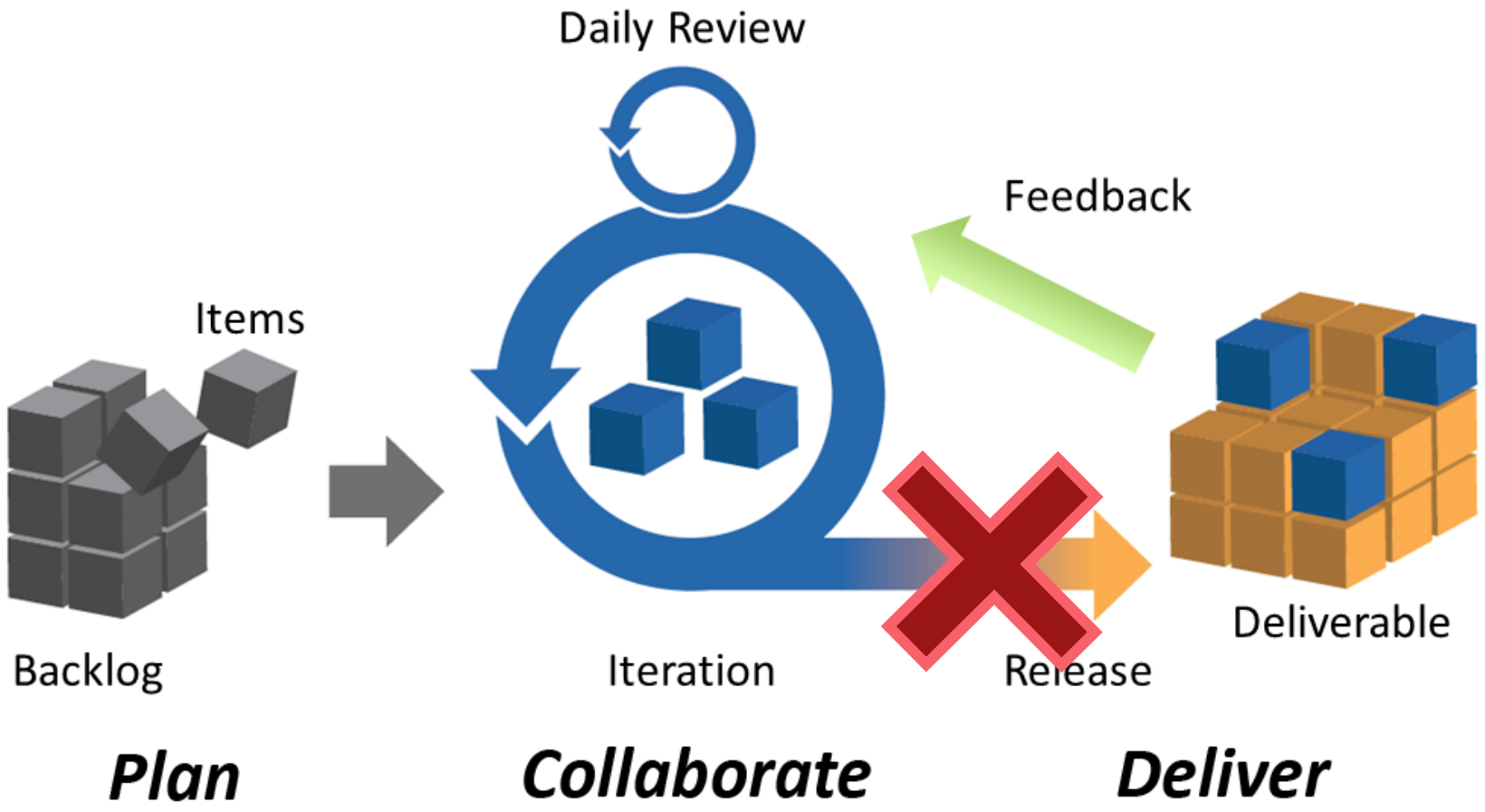
- Experience with Continuous Delivery?
- As a tester, do you need to wait for an acceptable release from developers?
- Loose time doing all repetitive manual stuff?
- Maintain different environments and software versions manually, over-and-over-and-over…
- Is this really release 1.6.5?

- All sounds familiar?

Building a release



Why Continuous Delivery



Agile Project Management: Iteration

What is Continuous Delivery

- Continuous Delivery pipeline
- Automation
- Manual control
- Feedback

Goals

- Predictability
- Quality
- Speed

What are the key benefits of Continuous Delivery?

- Low-risk releases
- Faster return on investment in software projects
- Improvement of competitiveness and responsiveness
- Quality improvement of new software versions

Continuous Improvement Cycle

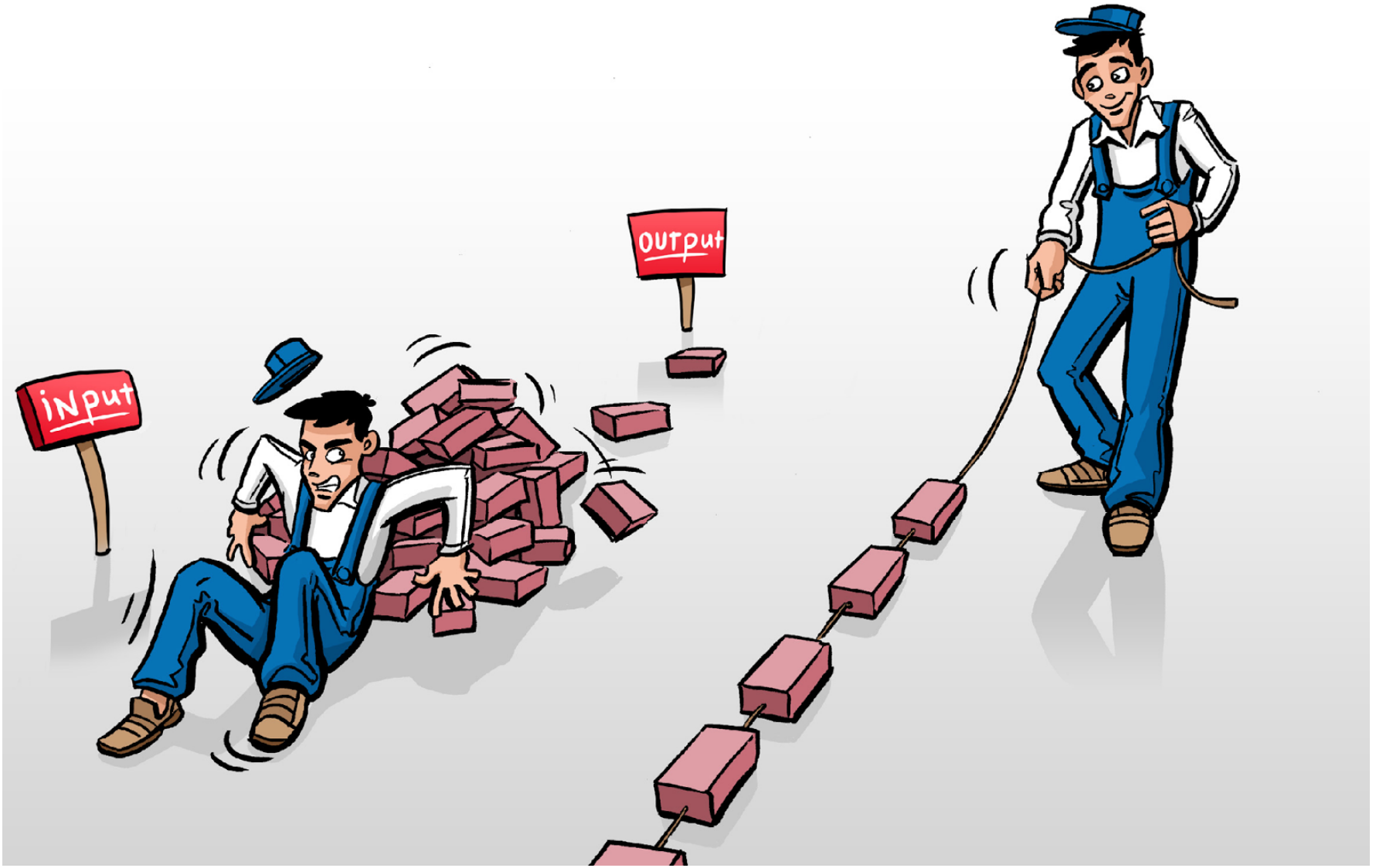
Requires:

Discipline &

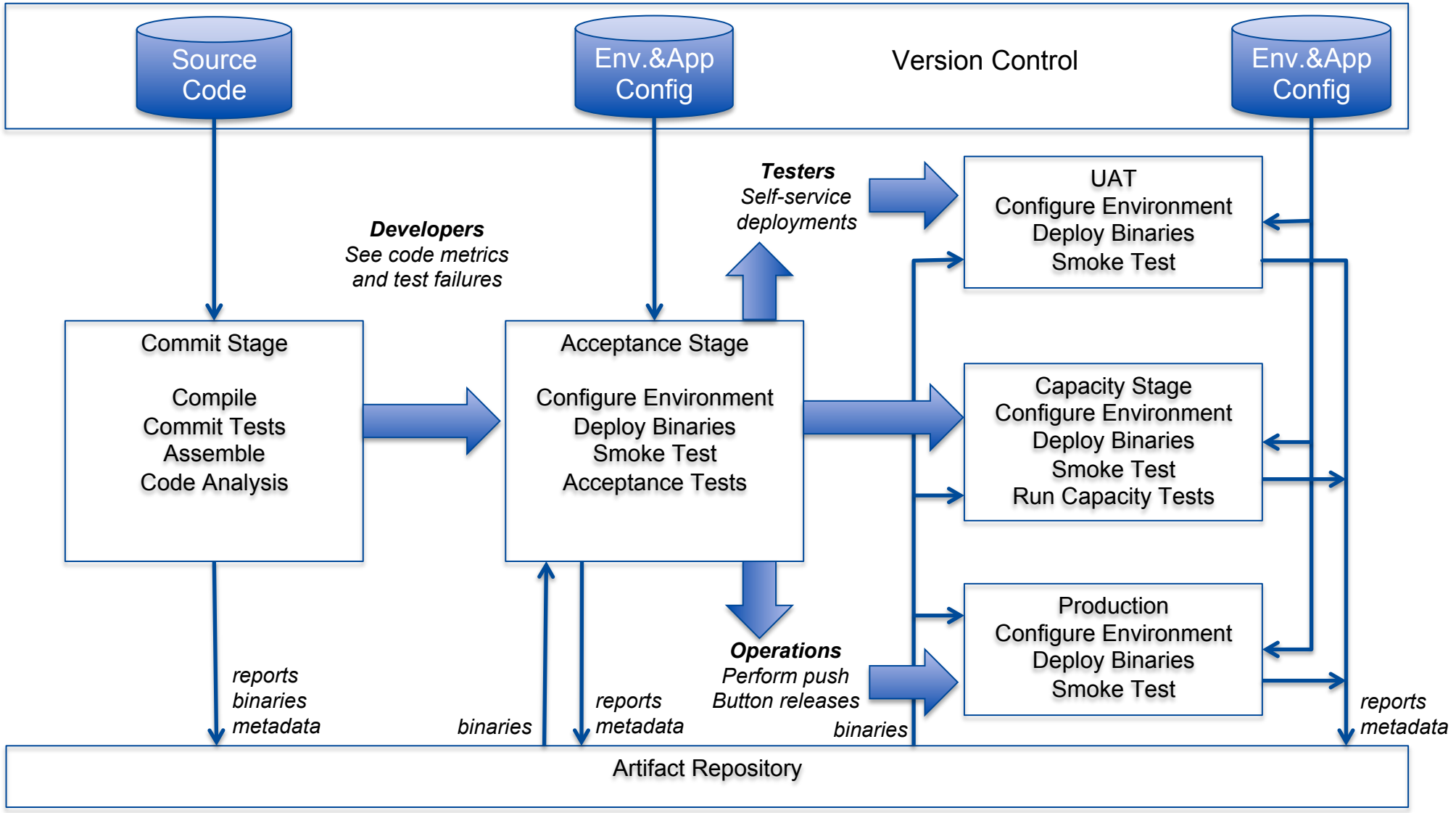
Responsibility

Stop the line!

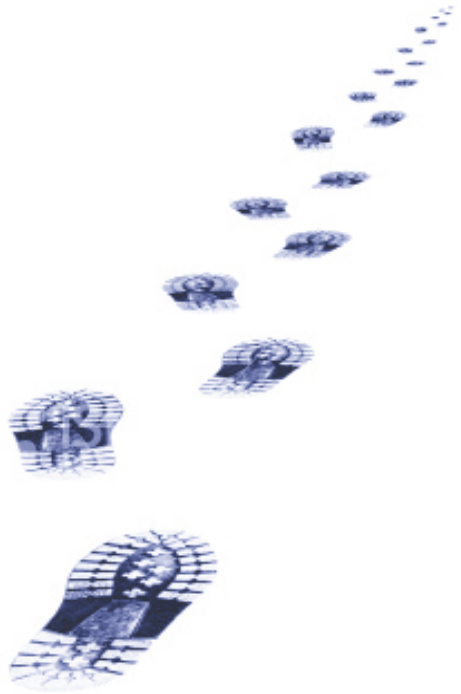
Push vs. Pull



ANATOMY IN DETAIL



What are the main steps of Continuous Delivery?



- Build – compile, unit test, version, package
- Quality – metrics, documentation
- Test – acceptance-, regression- and performance tests
- Provision environments – deployment to test- and staging environment
- Production – green/blue deployment to production

Main steps to create a Continuous Delivery pipeline?

- Getting started with virtual environments, e.g. Amazon EC2
- Configure your Continuous Integration Server
- Provisioning your Test, Staging and Production environments
- Configure your Continuous Delivery Pipeline
- Create a Dashboard of your Systems

EXAMPLE

codecentric

Continuous Delivery :: Production Step :: select version and environment for deployment

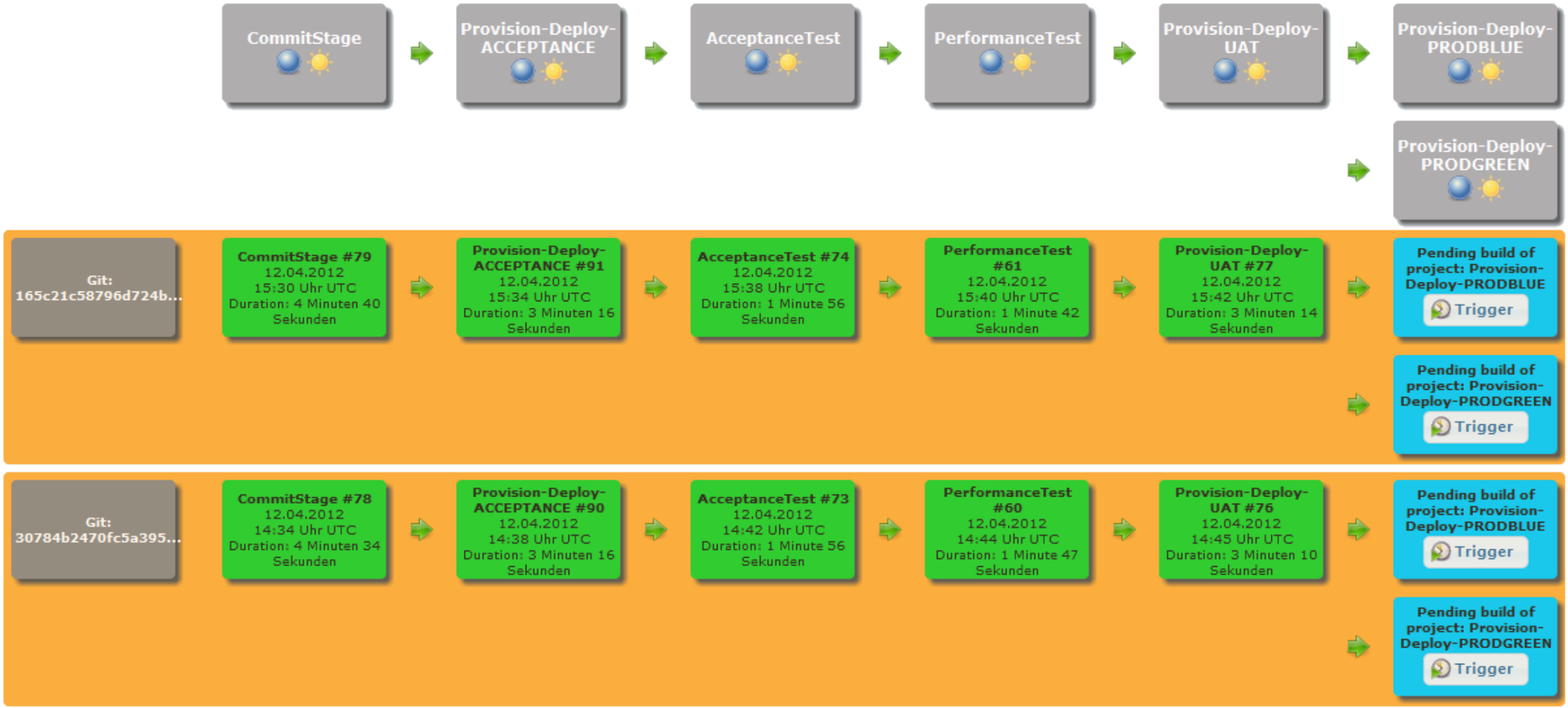
Available Releases

Deploy VERSION Version 1.4.237 to ENVIRONMENT PRODUCTION-BLUE Submit

Continuous Delivery :: System Status :: displays server status and versions deployed

Acceptance/Performance Test	UAT Environment	Production Green	Production Blue
Version: 1.4.237	Version: 1.4.236	Version: 1.4.174	Version: 1.4.225
Go to ACCEPTANCE	UAT-ENVIRONMENT	not running	PRODUCTION-BLUE
Status: running AWS Instance ID: i-44b9bc20 Instance-Type: m1.small Time started: 2012-04-12T10:25:14+0000 AWS-TAG: ACCEPTANCE	Status: running AWS Instance ID: i-abd722cc Instance-Type: t1.micro Time started: 2012-06-22T07:39:53+0000 AWS-TAG: UAT-ENVIRONMENT	Status: not running AWS Instance ID: i-375eab50 Instance-Type: t1.micro Time started: AWS-TAG: PRODUCTION-GREEN	Status: running AWS Instance ID: i-40181224 Instance-Type: t1.micro Time started: 2012-06-22T08:09:47+0000 AWS-TAG: PRODUCTION-BLUE
			 LIVE

What are the main steps of Continuous Delivery?

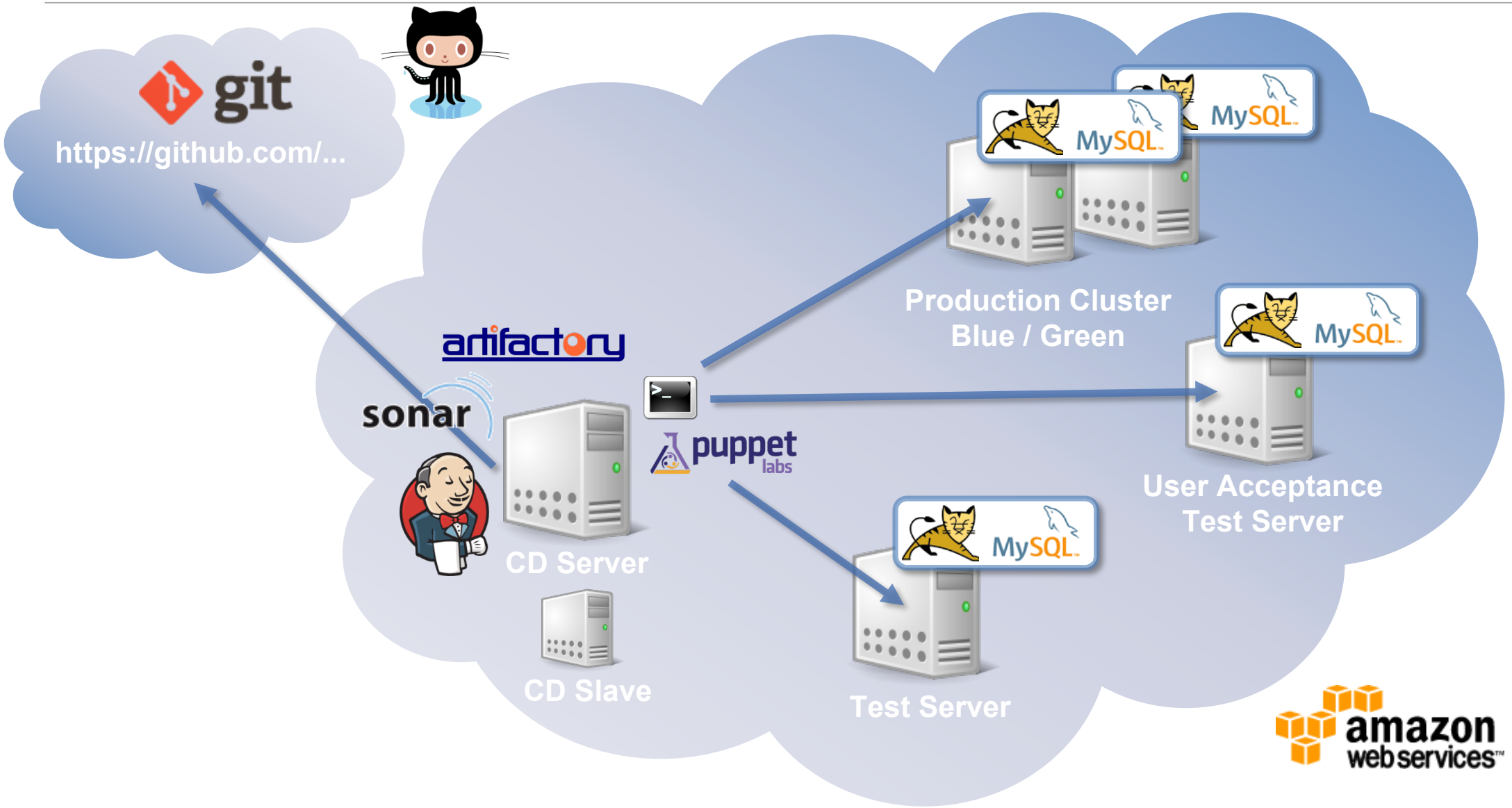


Tooling that can help you

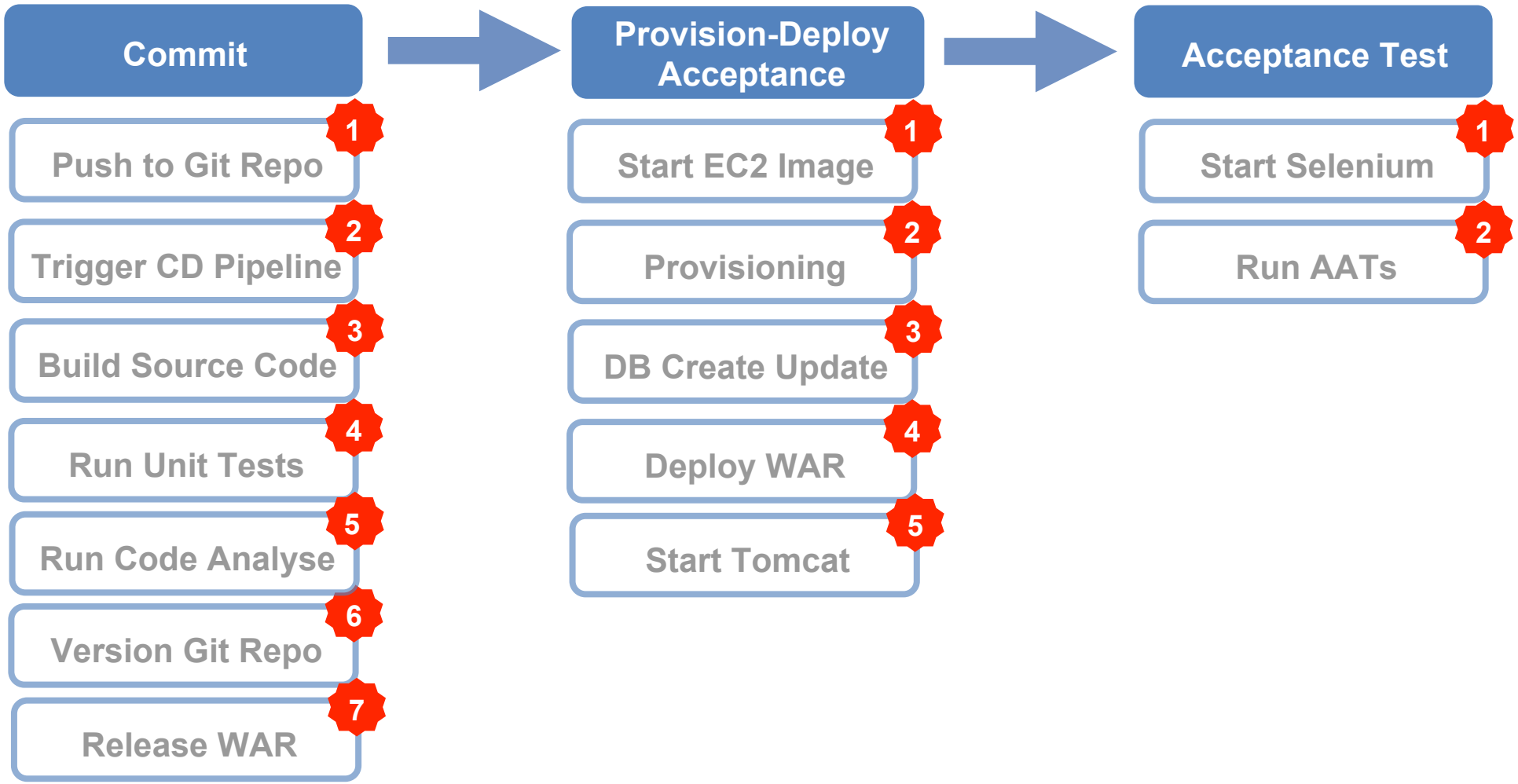
Example – Technology Stack

- Amazon EC2 (Virtualization, Cloud)
- Jenkins (CI Server)
- Git/Github (Version Control)
- Sonar (Code Quality)
- jUnit (Unit Tests)
- jBehave (Acceptancetests)
- Selenium (UI Tests)
- Puppet (Provisioning)
- Tomcat (Application Server)
- Artifactory (Artifact Repository)
- Maven (Build Tool)
- jMeter (Performance Tests)
- AppDynamics (Performance Tests)
- Liquibase (DB Versioning)

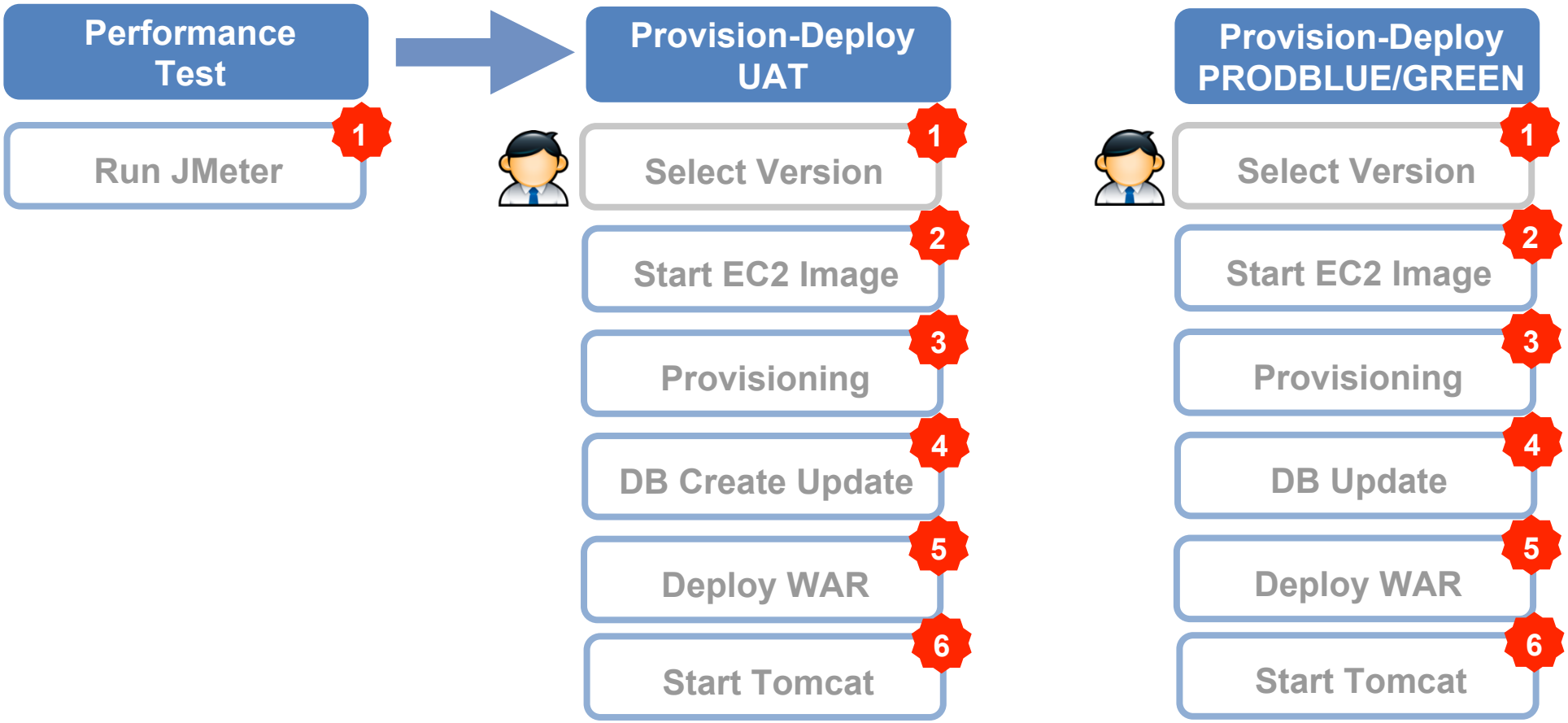
Our Setup


git
https://github.com/...
artifactory
sonar
puppet labs
CD Server
CD Slave
Production Cluster
Blue / Green
MySQL
MySQL
User Acceptance
Test Server
MySQL
Test Server
amazon
web services

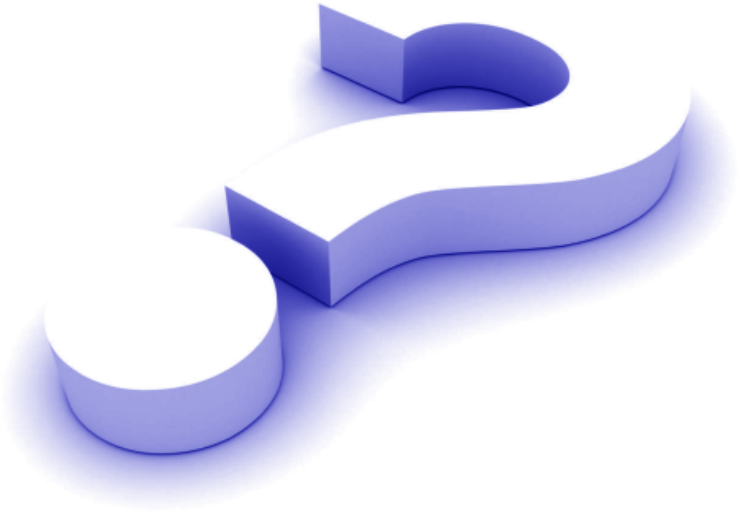
Example – Delivery Pipeline 1/2



Example – Delivery Pipeline 2/2

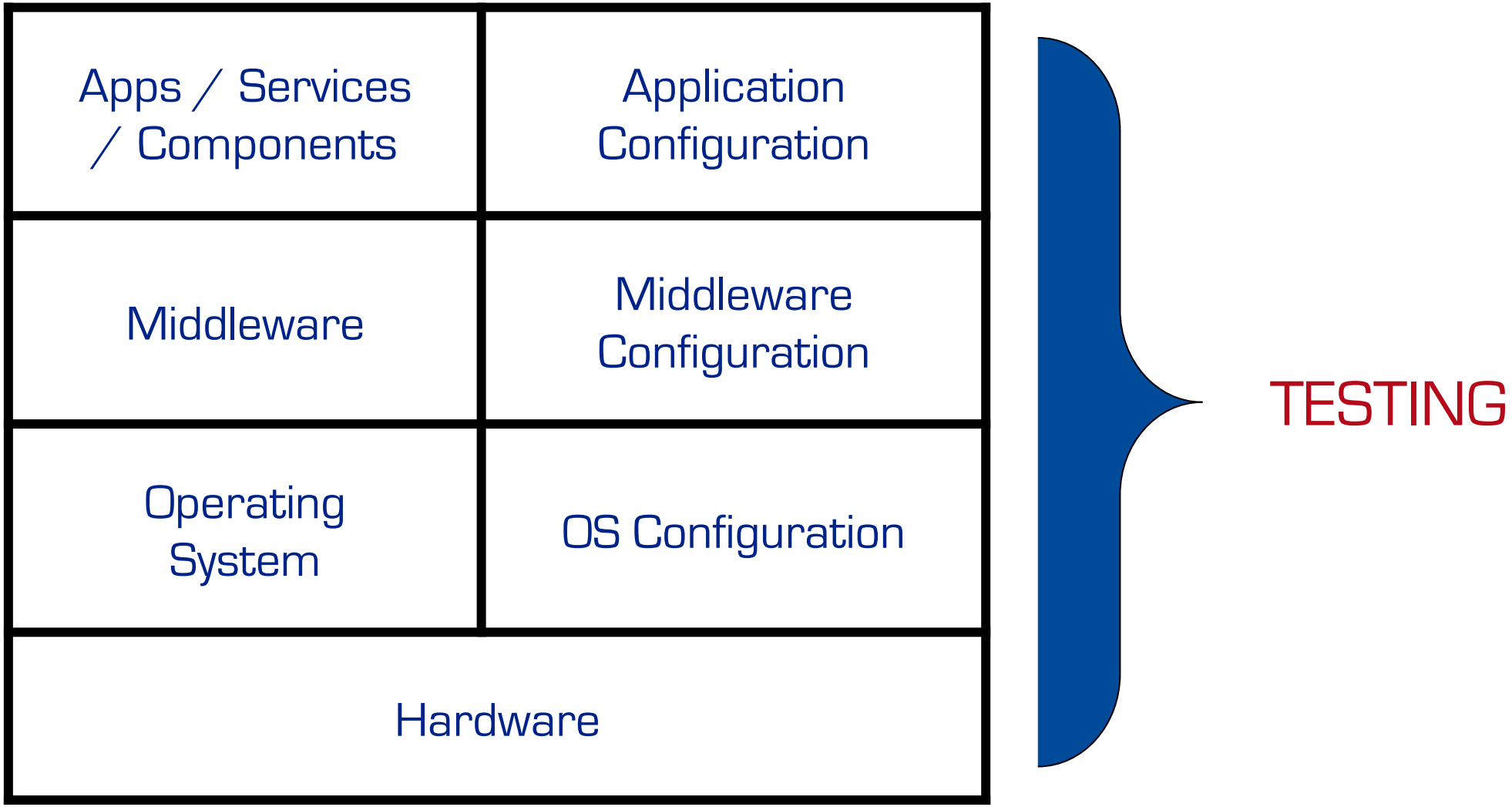


Ask yourself



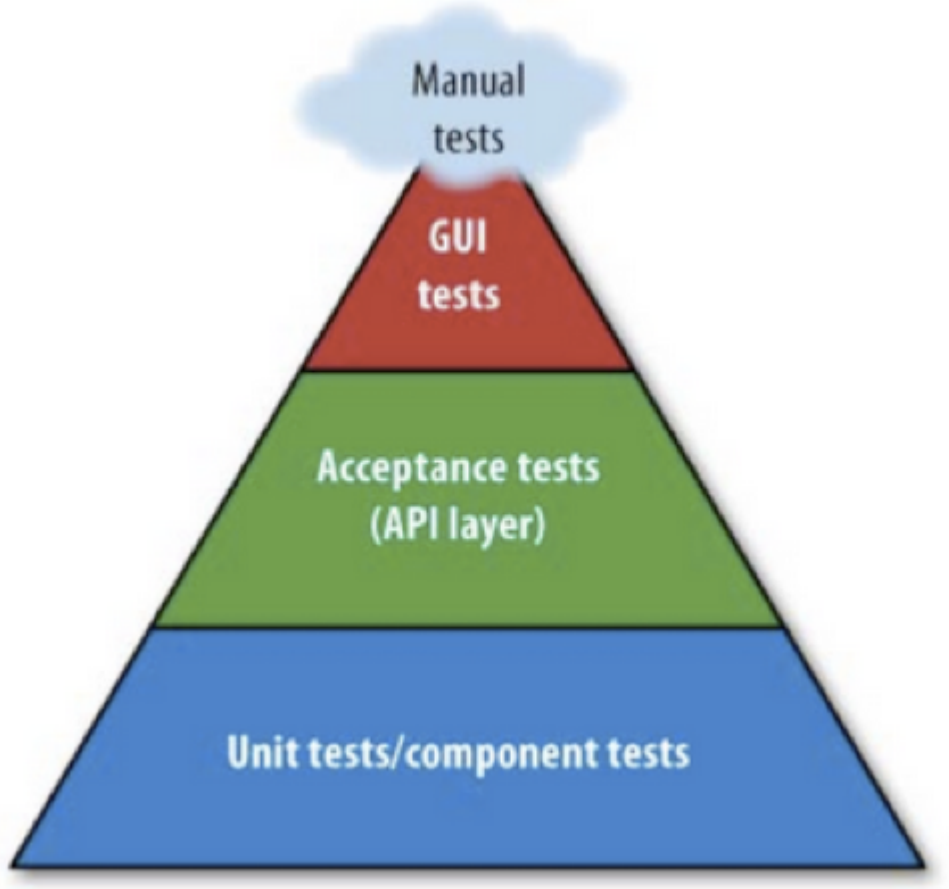
- Which parts are tested how?
- What remains to be tested?

Application Stack



What do we test where and when?

- Automation is key
- Do what we do best
- Team effort

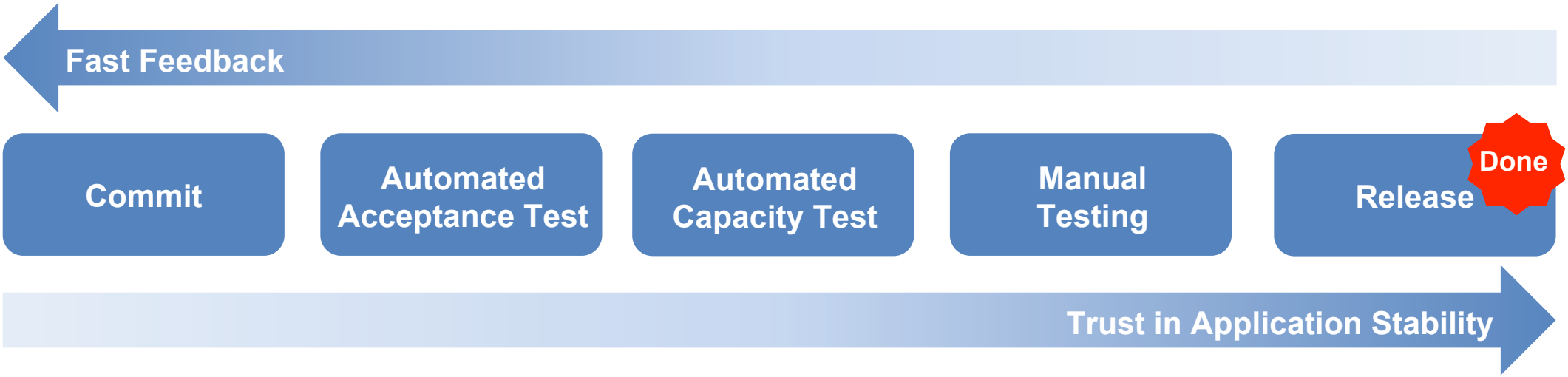


Manual testing

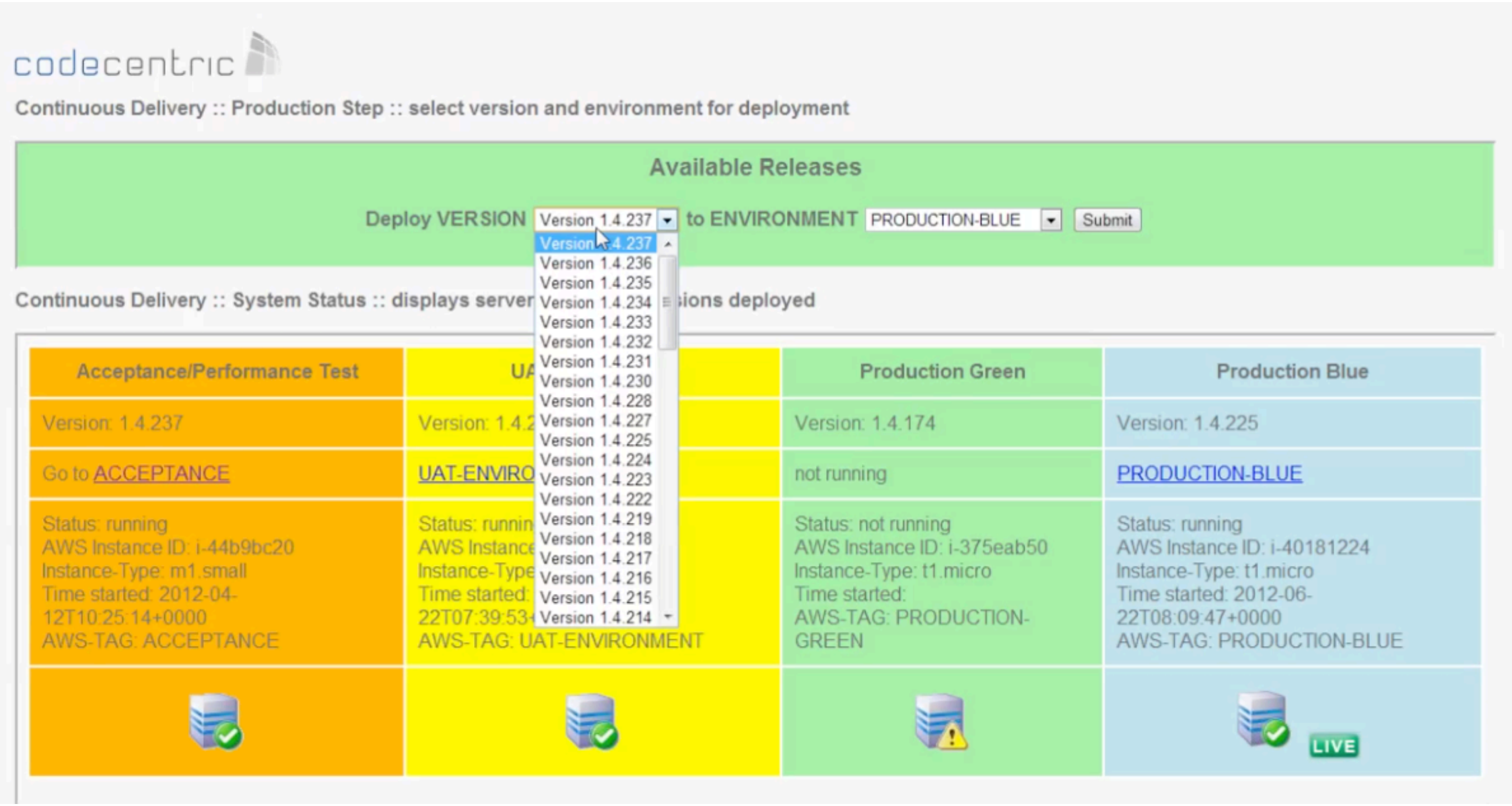
Still required:

- “Show me”
- Exploratory testing
- User acceptance testing

Anatomy of a Deployment pipeline



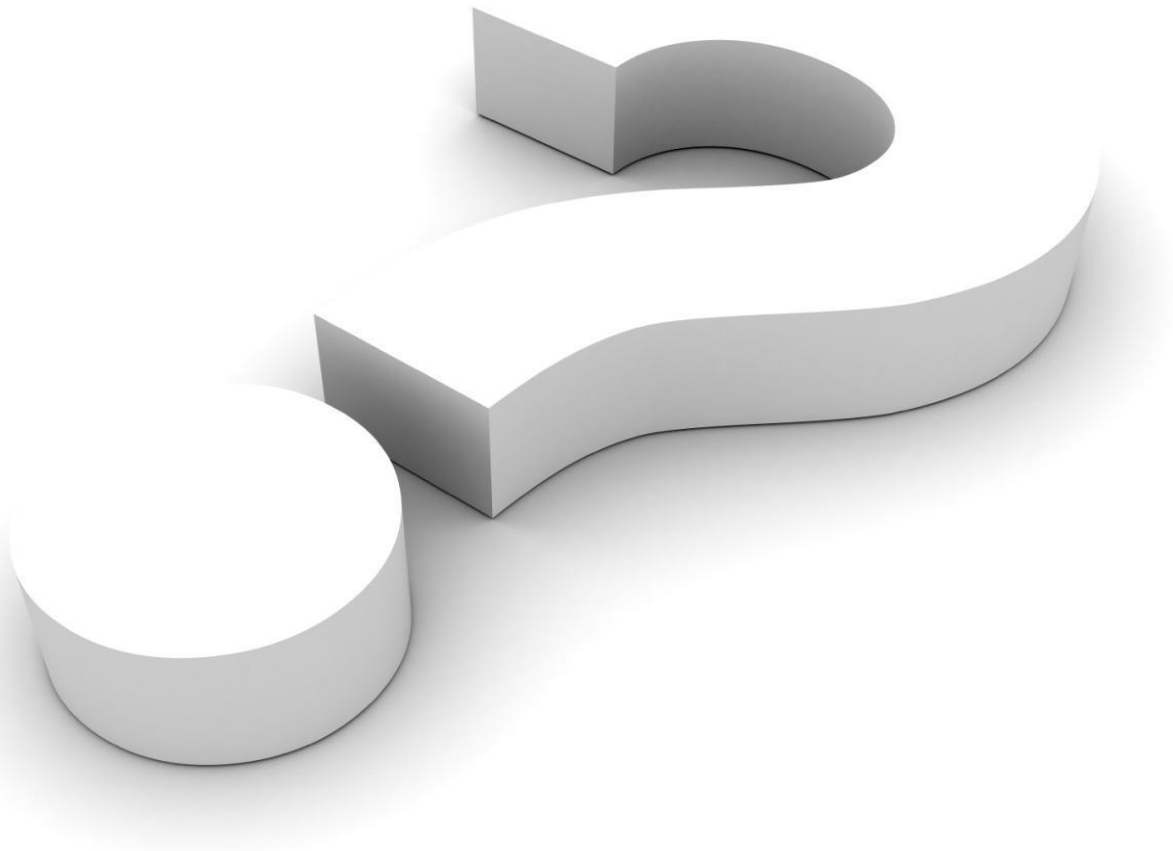
- Every change results in a trigger of the deployment process
- Software is build once and only once
- The same deployment process for every environment
- Deployment in production-like environments


codecentric
Continuous Delivery :: Production Step :: select version and environment for deployment
Available Releases
Deploy VERSION Version 1.4.237 to ENVIRONMENT PRODUCTION-BLUE Submit
Version 1.4.237
Version 1.4.236
Version 1.4.235
Version 1.4.234
Version 1.4.233
Version 1.4.232
Version 1.4.231
Version 1.4.230
Version 1.4.228
Version 1.4.227
Version 1.4.225
Version 1.4.224
Version 1.4.223
Version 1.4.222
Version 1.4.219
Version 1.4.218
Version 1.4.217
Version 1.4.216
Version 1.4.215
Version 1.4.214
Continuous Delivery :: System Status :: displays server
ions deployed
Acceptance/Performance Test
Production Green
Production Blue
Version: 1.4.237
Version: 1.4.174
Version: 1.4.225
Go to ACCEPTANCE
not running
PRODUCTION-BLUE
Status: running
AWS Instance ID: i-44b9bc20
Instance-Type: m1.small
Time started: 2012-04-12T10:25:14+0000
AWS-TAG: ACCEPTANCE
Status: not running
AWS Instance ID: i-375eab50
Instance-Type: t1.micro
Time started:
AWS-TAG: PRODUCTION-GREEN
Status: running
AWS Instance ID: i-40181224
Instance-Type: t1.micro
Time started: 2012-06-22T08:09:47+0000
AWS-TAG: PRODUCTION-BLUE
LIVE

Summary

- Continuous delivery is a way to create a stable release process
- Automated testing makes continuous delivery possible
- Manual testing is (still) necessary
- Pull vs Push
- Dashboard

THANK YOU FOR YOUR ATTENTION – ANY QUESTIONS?



Contact Information

huib.schoots@codecentric.nl
+31 (0) 6 24 64 10 33
@huibschoots

miel.donkers@codecentric.nl
+31 (0) 6 51 19 77 38
@mieldonkers

See our website for more information:

http://www.codecentric.nl/portfolio/continuous-delivery/