MGL7460 — Automne 2016

Intégration, livraison et déploiement continus

Guy Tremblay
Professeur

Département d'informatique
UQAM
http://www.labunix.uqam.ca/~tremblay

17 novembre 2016

Contenu

1. Introduction : Automatisation de l'assemblage, des tests, . . .

Quelques rappels

Parmi les premières choses à faire, quand on développe du code de façon professionnelle…

«*[Y]ou need to get the development infrastructure environment in order. That means adopting (or improving) the fundamental Starter Kit practices :*

- *Version control*
- *Unit testing*
- *Build automation*

«*Practices of an Agile Developer—Working in the Real World*», Subramaniam & Hunt, 2006.

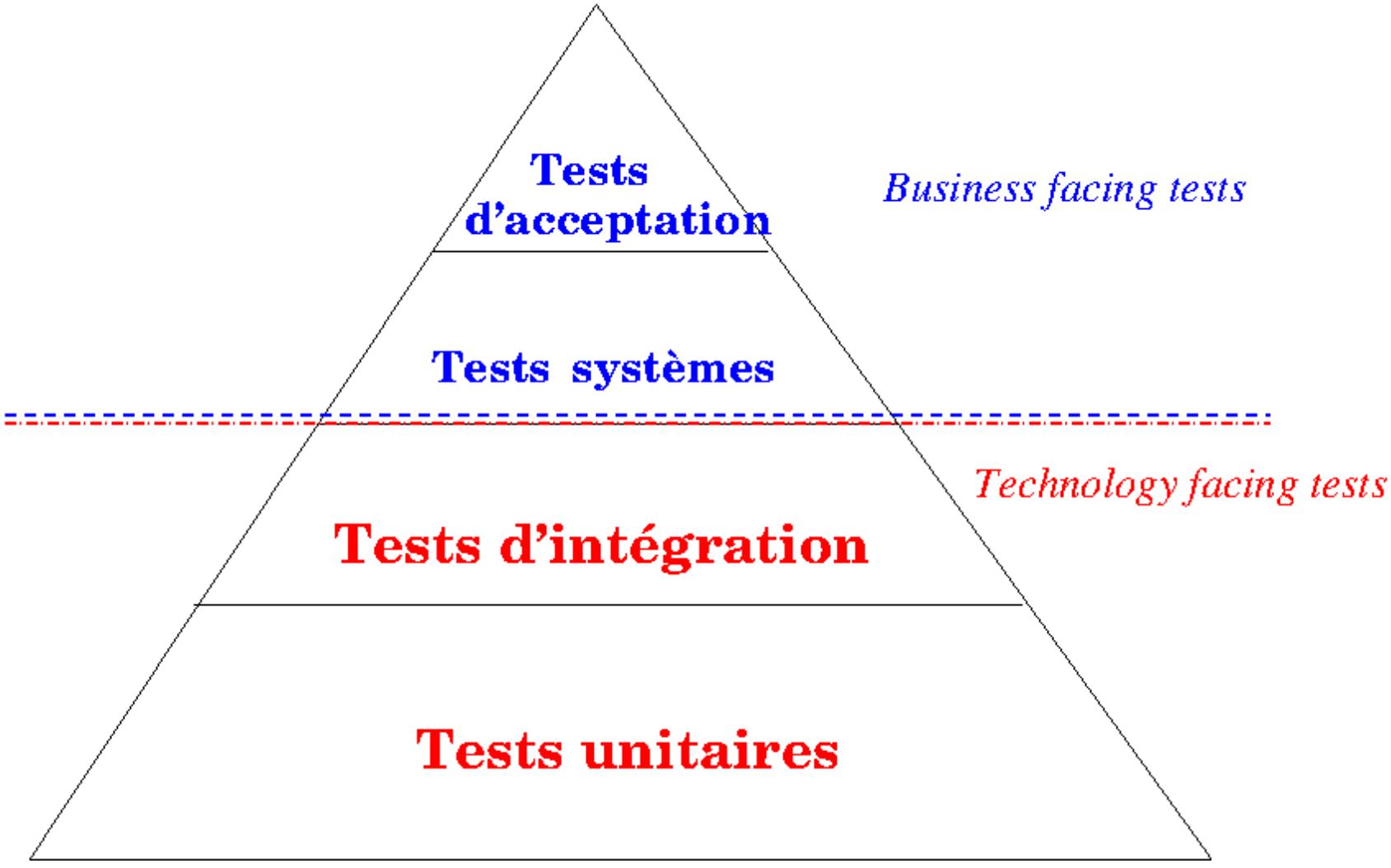
Parmi les premières choses à faire, quand on développe du code de façon professionnelle…

«*[Y]ou need to get the development infrastructure environment in order. That means adopting (or improving) the fundamental Starter Kit practices :*

- *Version control*
- *Unit testing*
- *Build automation*

«*Practices of an Agile Developer—Working in the Real World*», Subramaniam & Hunt, 2006.

Fait !


Tests
d'acceptation
Tests systèmes
Tests d'intégration
Tests unitaires
Business facing tests
Technology facing tests

Nothing can stop automation

Nothing can stop automation

Tests unitaires exécutés automatiquement

Tests d'acceptation exécutés automatiquement

Nothing can stop automation

Tests unitaires exécutés automatiquement

Tests d'acceptation exécutés automatiquement

Fait !

Les tests d'acceptation jouent un rôle de tests d'intégration

Cucumber scenarios test entire paths through the app and thus can be acceptance tests or integration tests.

Source*:* «Engineering Software as as Service—An Agile Approach Using Cloud Computing»*, Fox & Patterson*

Mais les tests d'intégration... sont plus que cela

«We use the term integration testing to refer to tests which ensure that each **independent part** *of your application works correctly with the services it depends on.»*

«*Continuous Delivery*», Humble & Farley

Mais les tests d'intégration. . . sont plus que cela

«Integration testing shows that the **major subsystems** *that make up the project work and play well together.»*

«*The Pragmatic Programmer*», Hunt & Thomas

Mais les tests d'intégration. . . sont plus que cela

«*Integration testing shows that the* **major subsystems** *that make up the project work and play well together.*

[I]ntegration [can] become a fertile breeding ground for bugs. In fact, it is often the single largest source of bugs in the system.»

«*The Pragmatic Programmer*», Hunt & Thomas

D’où viennent ces diverses *independent parts*, ces divers *subsystems* ?

Donc, quelle est la prochaine étape ?

«*Practices of an Agile Developer*» : Tip 14

Subramaniam & Hunt, 2006

Tip 14

Integrate early, integrate often.

Code integration is a major source of risk. To mitigate that risk, start integration early and continue to do it regularly.

If It Hurts, Do It More Frequently, and Bring the Pain Forward

This is [. . .] perhaps the most useful heuristic we know of in the context of delivering software, and it informs everything we say.

Integration is often a very painful process. If this is true on your project, integrate every time somebody checks in, and do it from the start of the project. If testing is a painful process that occurs just before release, don't do it at the end. Instead, do it continually from the beginning of the project.

«Continuous Delivery», *Humble & Farley*

2. Intégration continue

Qu'est-ce que l'intégration continue

Intégration continue = *Continuous Integration* = CI

Continuous Integration is a software development practice where **members of a team integrate their work frequently**, *usually each person integrates at least daily—leading to multiple integrations per day.*

Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

Source: *«Continuous Integration», M. Fowler,*
`http://www.martinfowler.com/articles/continuousIntegration.html`

Qu'est-ce que l'intégration continue

Intégration continue = *Continuous Integration* = CI

Continuous Integration is a software development practice where **members of a team integrate their work frequently**, *usually each person integrates at least daily—leading to multiple integrations per day.*

Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible.

Source: *«Continuous Integration», M. Fowler,*
`http://www.martinfowler.com/articles/continuousIntegration.html`

Qu'est-ce que l'intégration continue

Intégration continue = *Continuous Integration* = CI

Continuous Integration is a software development practice where **members of a team integrate their work frequently**, *usually each person integrates at least daily—leading to multiple integrations per day.*

*Each integration is verified by an automated build (***including test***) to detect integration errors as quickly as possible.*

Source*: «Continuous Integration», M. Fowler,*
`http://www.martinfowler.com/articles/continuousIntegration.html`

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

The key practices that make up effective CI

http://www.martinfowler.com/articles/continuousIntegration.html

- *Maintain a Single Source Repository*
- *Automate the Build*
- *Make Your Build Self-Testing*
- *Everyone Commits To the Mainline Every Day*
- *Every Commit Should Build the Mainline on an Integration Machine*
- *Fix Broken Builds Immediately*
- *Keep the Build Fast*
- *Test in a Clone of the Production Environment*
- *Make it Easy for Anyone to Get the Latest Executable*
- *Everyone can see what's happening*

Un serveur d'intégration continue est utile, bien que non strictement nécessaire

Although Continuous Integration is a practice that requires no particular tooling to deploy, we've found that it is useful to use a Continuous Integration server.

Source: *«Continuous Integration», M. Fowler,*

`http://www.martinfowler.com/articles/continuousIntegration.html`

Quelques exemples de serveurs d'intégration continue

Quelques exemples de serveurs d'intégration continue

Note : Le site suivant en présente…

`https://xebialabs.com/the-ultimate-devops-tool-chest/continuous-integration`

Quelques exemples de serveurs d'intégration continue

Note : Le site suivant en présente. . . deux douzaines

`https://xebialabs.com/the-ultimate-devops-tool-chest/continuous-integration`

Le labo que nous ferons tantôt…

Travis CI

Le labo que nous ferons tantôt…

Travis CI

- Logiciel libre
- Service en ligne
- Écrit en Ruby, mais utilisable avec de nombreux langages

Travis CI supporte divers langages


Clojure


Node

PHP


Xcode


Ruby

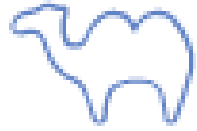

Python


Java


Erlang


Go


Scala

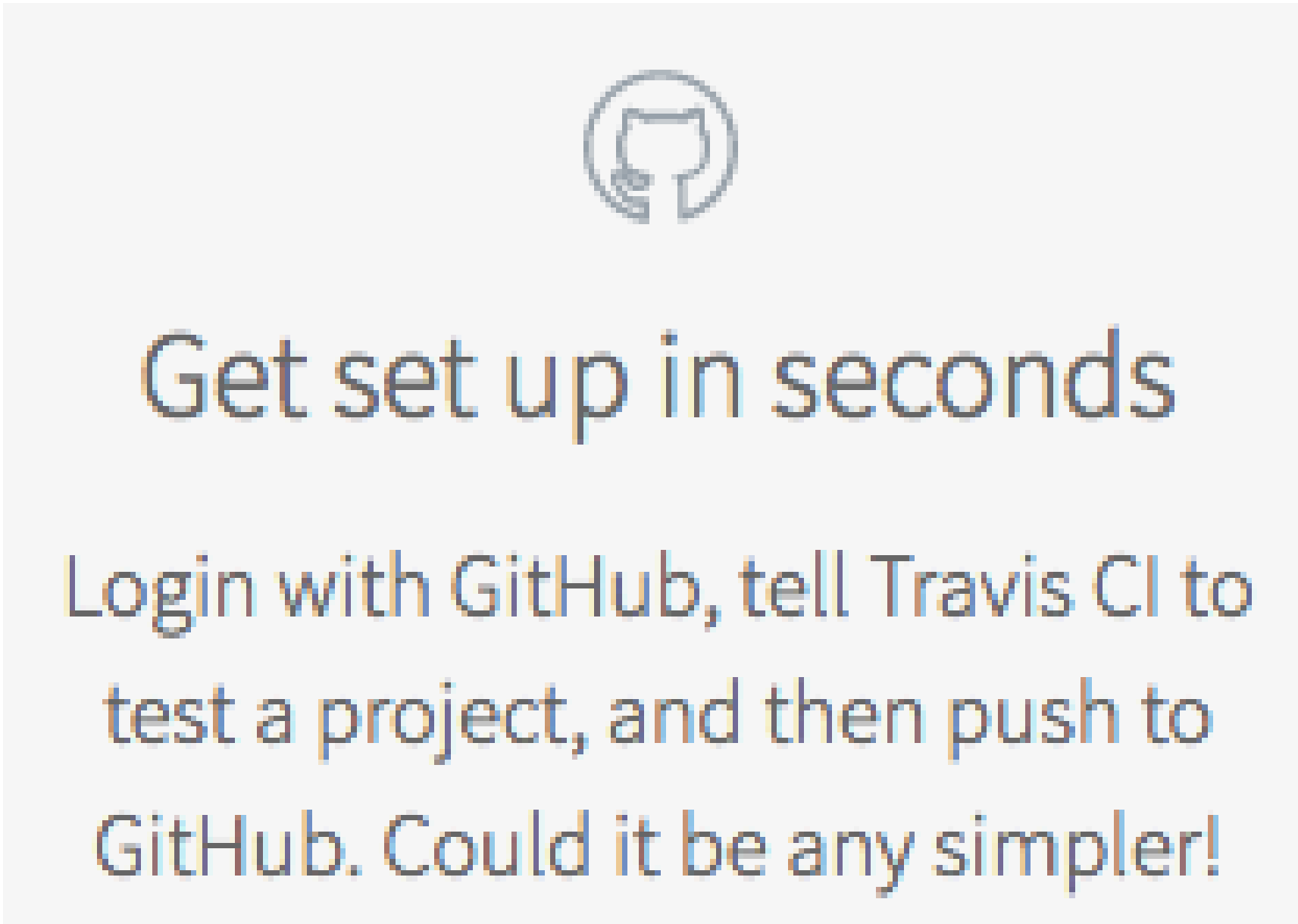

Perl

Source: `https://travis-ci.com/`

Features created to help your projects and teams

- ✓ Watch your tests as they run
- ✓ Keep your config with your code
- ✓ Slack, HipChat, Emails and more
- ✓ A clean VM for every build
- ✓ Run your tests in parallel
- ✓ Linux and Mac (and iOS) supported
- ✓ Great API and command line tool

Source: `https://travis-ci.com/`



Source: `https://travis-ci.com/`

3. Livraison continue

La livraison continue va au-delà de l'intégration continue

Continuous Delivery is a software development discipline where you build software in such a way that the software **can be released to production** *at any time.*

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Quand fait-on de le livraison continue ?

- *Your software is deployable throughout its lifecycle*
- *Your team prioritizes keeping the software deployable over working on new features*
- *Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them*
- *You can perform* **push-button** *deployments of any version of the software to any environment on demand*

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Quand fait-on de le livraison continue ?

- *Your software is deployable throughout its lifecycle*
- *Your team prioritizes keeping the software deployable over working on new features*
- *Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them*
- *You can perform* **push-button** *deployments of any version of the software to any environment on demand*

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Quand fait-on de le livraison continue ?

- *Your software is deployable throughout its lifecycle*
- *Your team prioritizes keeping the software deployable over working on new features*
- *Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them*
- *You can perform* **push-button** *deployments of any version of the software to any environment on demand*

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Quand fait-on de le livraison continue ?

- *Your software is deployable throughout its lifecycle*
- *Your team prioritizes keeping the software deployable over working on new features*
- *Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them*
- **You can perform push-button deployments of any version of the software to any environment on demand**

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Comment fait-on de le livraison continue ?

You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems.

Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production.

To do this you use a DeploymentPipeline.

Note : *check-in* = **push**

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Comment fait-on de le livraison continue ?

You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems.

Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production.

To do this you use a DeploymentPipeline.

Note : *check-in* = **push**

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Comment fait-on de le livraison continue ?

You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems.

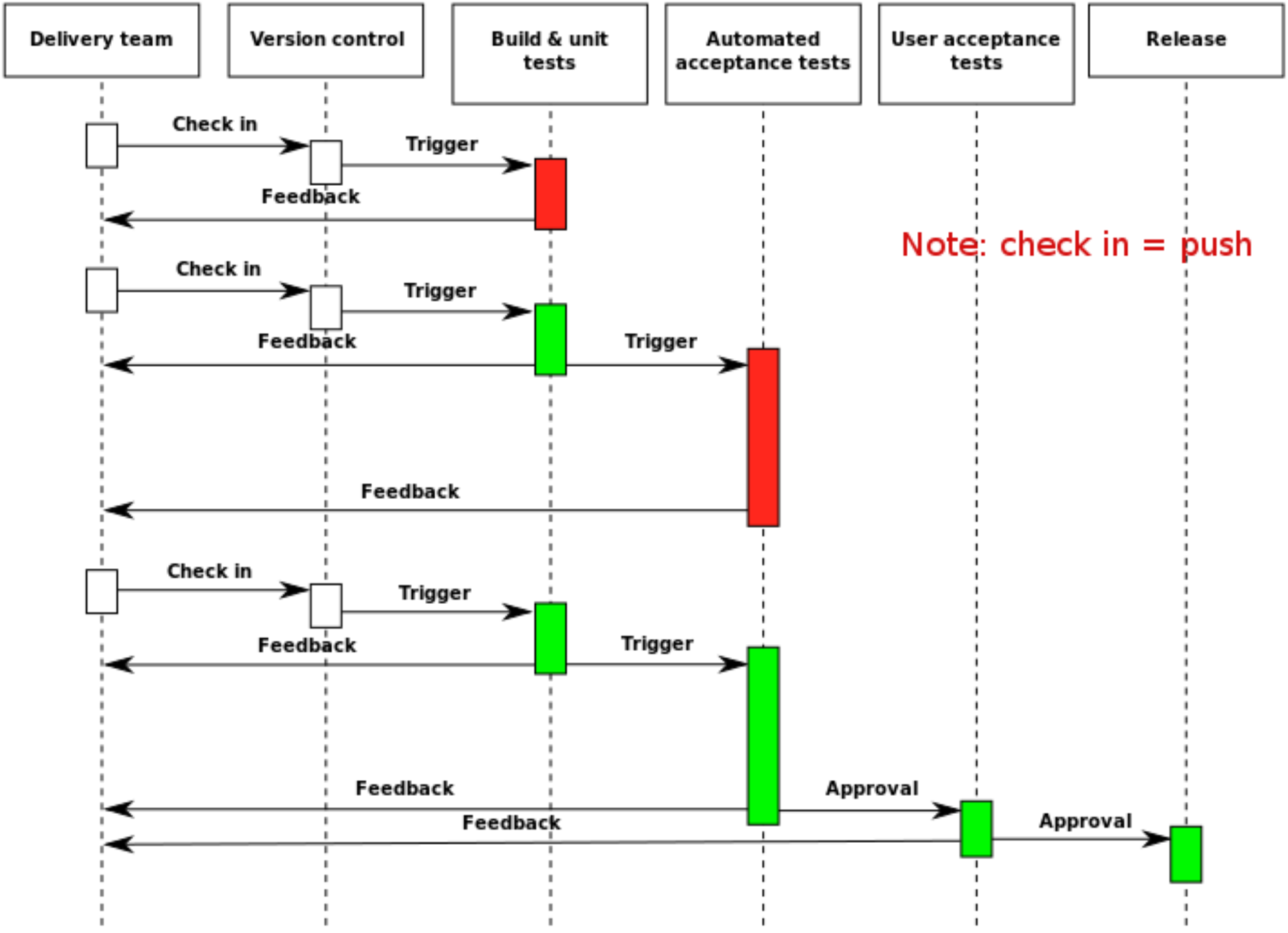
Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production.

To do this you use a DeploymentPipeline.

Note : *check-in* = **push**

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

https://en.wikipedia.org/wiki/Continuous_delivery



Les principaux bénéfices de la livraison continue

- *Reduced Deployment Risk : since you are deploying smaller changes, there's less to go wrong and it's easier to fix should a problem appear.*

Les principaux bénéfices de la livraison continue

- *Reduced Deployment Risk : since you are deploying smaller changes, there's less to go wrong and it's easier to fix should a problem appear.*
- *Believable Progress : many folks track progress by tracking work done. If "done" means "developers declare it to be done" that's much less believable than if it's deployed into a production [. . .] environment.*

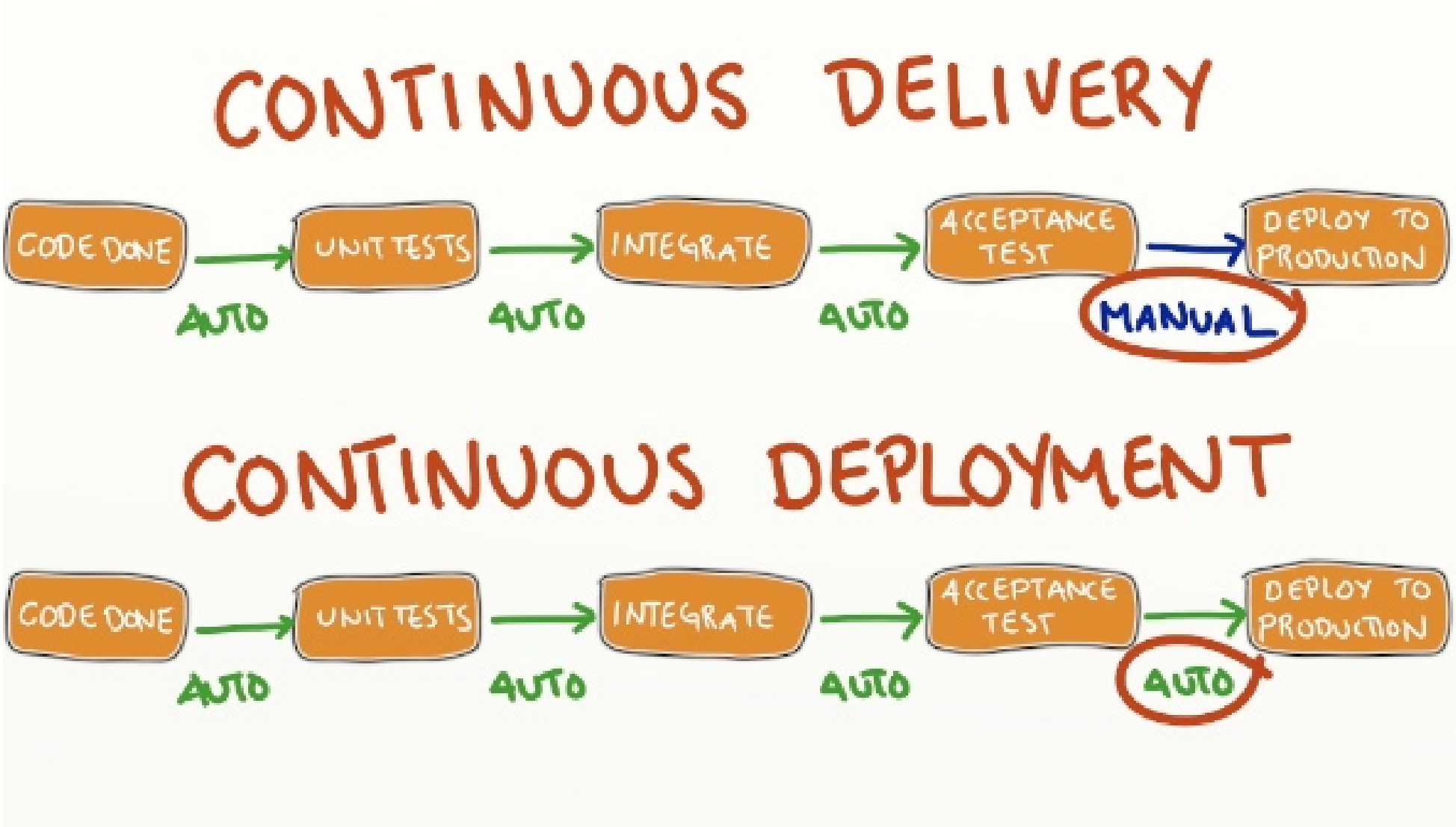
Les principaux bénéfices de la livraison continue

- *Reduced Deployment Risk : since you are deploying smaller changes, there's less to go wrong and it's easier to fix should a problem appear.*
- *Believable Progress : many folks track progress by tracking work done. If "done" means "developers declare it to be done" that's much less believable than if it's deployed into a production [. . .] environment.*
- *User Feedback : the biggest risk to any software effort is that you end up building something that isn't useful. The earlier and more frequently you get working software in front of real users, the quicker you get feedback to find out how valuable it really is [. . .].*

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

4. Déploiement continu

http://codeahoy.com/2016/06/18/continuous-delivery-automating-the-release-process/



Livraison continue vs. Déploiement continu

Continuous delivery vs. continuous deployment

Continuous Delivery is sometimes confused with Continuous Deployment.

Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day.

Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment.

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Livraison continue vs. Déploiement continu

Continuous delivery vs. continuous deployment

Continuous Delivery is sometimes confused with Continuous Deployment.

Continuous Deployment means that every change goes through the pipeline and **automatically gets put into production**, *resulting in many production deployments every day.*

Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment.

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

Livraison continue vs. Déploiement continu

Continuous delivery vs. continuous deployment

Continuous Delivery is sometimes confused with Continuous Deployment.

Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day.

Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment.

Source: *«Continuous Delivery», M. Fowler,* `http://martinfowler.com/bliki/ContinuousDelivery.html`

5. Conclusion : La prochaine étape. . .

Après l’intégration continue, la livraison continue et le déploiement continu, quelle est la nouvelle «tendance» ?

Article tiré de «La Presse+», 19 novembre 2016

TECHNO

CINQ EMPLOIS BRANCHÉS EN 2017

Quels seront les employés les plus recherchés en 2017 ? Deux cabinets spécialisés en recrutement nous font part des tendances qu'ils observent sur le marché du travail. Devinez quoi : elles sont liées aux technologies de l'information.

Quel est l'emploi en 1ère position ?

En première position = *DevOps*

Source : «La Presse+», édition du 19 novembre

Devops, dites-vous ? Contraction des mots anglais *development* et *operations*, le terme désigne une approche globale pour un déploiement plus efficace des systèmes informatiques dans une organisation.

Ce processus vise à coordonner le travail des développeurs de logiciels et des responsables de l'exploitation au sein de l'entreprise — deux fonctions traditionnellement indépendantes. Ce courant, apparu vers 2009, s'est répandu comme une traînée d'électrons.

L'approche a donné son nom à son praticien — devops — que nous traduirons ici par le nettement moins exotique «spécialiste en processus de développement et d'exploitation».

«C'est le poste le plus tendance, à l'heure actuelle», constate Emily Woods, gestionnaire pour les emplois en technologie de l'information chez Groom et associés. «Cette personne doit connaître le cloud, l'infrastructure, les processus. Les grandes entreprises comme Amazon et Netflix engagent beaucoup de devops. Ils sont également très recherchés à Montréal : c'est la grande tendance.»

En première position = *DevOps*

Source : «La Presse+», édition du 19 novembre

Devops, dites-vous ? Contraction des mots anglais *development* et *operations*, le terme désigne une approche globale pour un déploiement plus efficace des systèmes informatiques dans une organisation.

Ce processus vise à coordonner le travail des développeurs de logiciels et des responsables de l'exploitation au sein de l'entreprise — deux fonctions traditionnellement indépendantes. Ce courant, apparu vers 2009, s'est répandu comme une traînée d'électrons.

L'approche a donné son nom à son praticien — devops — que nous traduirons ici par le nettement moins exotique «spécialiste en processus de développement et d'exploitation».

«**C'est le poste le plus tendance, à l'heure actuelle**», constate Emily Woods, gestionnaire pour les emplois en technologie de l'information chez Groom et associés. «Cette personne doit connaître le cloud, l'infrastructure, les processus. Les grandes entreprises comme Amazon et Netflix engagent beaucoup de devops. Ils sont également très recherchés à Montréal : c'est la grande tendance.»

Qu'est-ce que le «*DevOps*» ?

DevOps [. . .] is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

It aims at establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably.

Source: `https://en.wikipedia.org/wiki/DevOps`

Qu'est-ce que le «*DevOps*» ?

DevOps [. . .] is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

It aims at establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably.

Source: `https://en.wikipedia.org/wiki/DevOps`

Qu'est-ce que le «*DevOps*» ?

DevOps [. . .] is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

It aims at establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably.

Source: `https://en.wikipedia.org/wiki/DevOps`

Qu'est-ce que le «*DevOps*» ?

DevOps is a development methodology with a set of practices aimed at bridging the gap between Development and Operations, emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment.

Source: *«What is DevOps ? : A Systematic Mapping Study on Definitions and Practices», Jabbari* et al., *XP '16 Workshops*

Continuous Delivery vs. *DevOps*

Continuous delivery and DevOps are similar in their meanings and are often conflated, but they are two different concepts.

DevOps has a broader scope, and centers around the cultural change, specifically the collaboration of the various teams involved in software delivery (developers, operations, quality assurance, management, etc.), as well as automating the processes in software delivery.

Continuous Delivery, on the other hand, is an approach to automate the delivery aspect, and focuses on bringing together different processes and executing them more quickly and more frequently.

Source: `https://en.wikipedia.org/wiki/DevOps`

Références

M. Fowler.

Continuous integration.

http ://www.martinfowler.com/articles/continuousIntegration.html, May 2006.

J. Humble and D. Farley.

Continuous Delivery—Reliable Software Releases Through Build, Test, and Deployment Automation.

Addison-Wesley, 2011.

J. Richardson and W.A. Gwaltney.

Ship it ! A Practical Guide to Successful Software Projects.

The Pragmatic Bookshelf, 2005.

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

Foreword by Martin Fowler

What is DevOps ?

https://theagileadmin.com/what-is-devops/