

# LOG8371 : Ingénierie de la qualité en logiciel

## **Livraison continue**

### **Hiver 2017**

Fabio Petrillo  
Chargé de Cours



This work is licensed under a Creative  
Commons Attribution-NonCommercial-  
ShareAlike 3.0 Unported License

# Image que....

- Hier votre patron vous a demandé de faire une **démonstration** des **nouvelles fonctionnalités** de ton système à un **client**, mais vous **ne pouvez pas rien monter...**
- Vous avez un **bug** critique dans la production, et vous savez comment le réparer, mais votre temps de le publier est d'une semaine ....
- Ces problèmes sont-ils communs?

# Image que....

- Hier votre patron vous a demandé de faire une démonstration des nouvelles fonctionnalités de ton système à un client, mais vous ne pouvez pas rien monter...
- Vous avez un bug critique dans la production, et vous savez comment le réparer, mais votre temps de le publier est d'une semaine ....
- Ces problèmes sont-ils communs?
  - **Oui**, Mais ils indiquent que quelque chose **ne va pas**!
  - Ils **ne sont pas un résultat inévitable** du processus de développement de logiciels!
- **Combien de temps faudrait-il à votre organisation pour déployer un changement qui ne comporte qu'une seule ligne de code?**
- Quel est le temps de votre cycle de décision que vous devez faire un changement pour l'avoir en production?

# Version communes d'**Anti patrons**

- Déploiement **manuel** du logiciel
  - Les applications de toute taille sont complexes à déployer, impliquant de nombreuses pièces en mouvement.
  - Problème d'erreur (Error-prone)
- Déploiement dans un environnement de production **uniquement** après l'**achèvement** du **développement**
- Gestion **manuelle de la configuration** des environnements de **production**

Alors, pouvons-nous faire mieux?

Déploiement  
**automatisé!**

# Intégration / Livraison **continue**

# Livraison continue

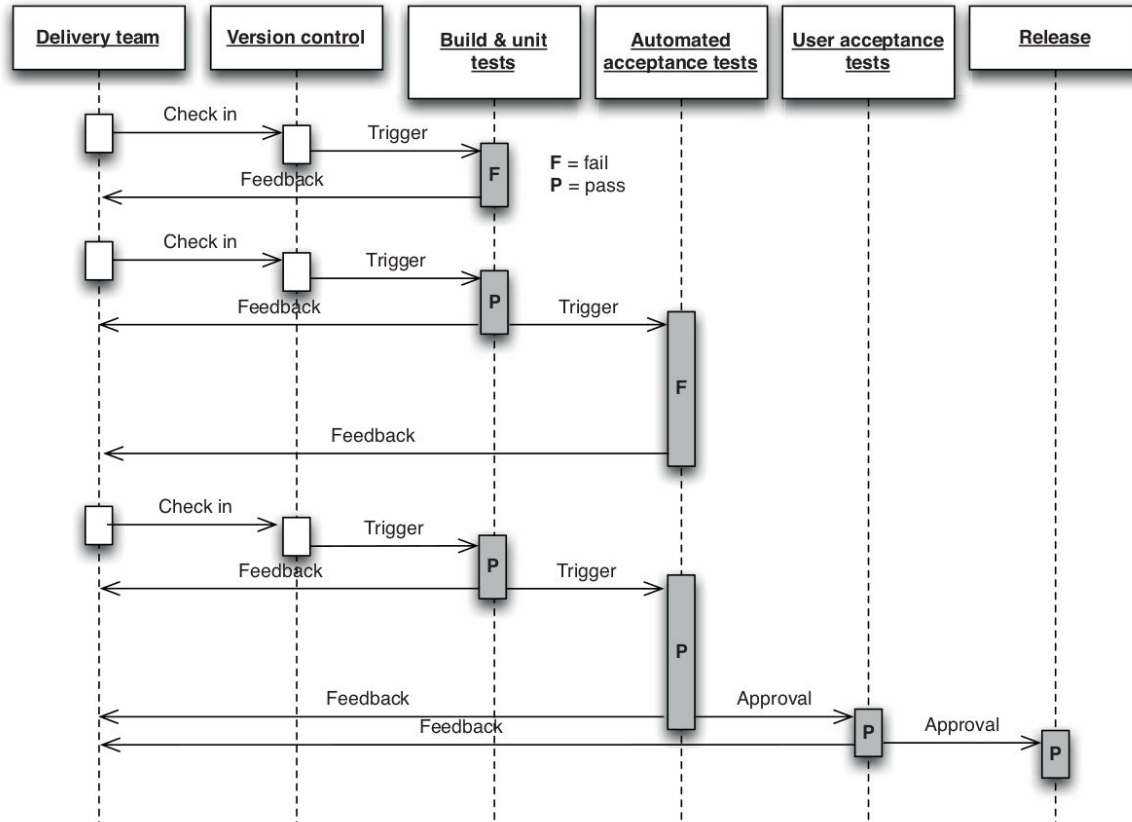
- La livraison continue est un pipeline de déploiement qui fournit une **mise en œuvre automatisée** du **build**, **déploiement**, **test** et **publication** de votre application
- Le **pipeline de déploiement** a ses fondements dans le processus d'intégration continue.
- Toute **modification** apportée à la **configuration**, **code source**, **l'environnement**, ou aux données d'une application, **déclenche** la création d'une nouvelle instance de la pipeline.

# L'objectif du déploiement pipeline

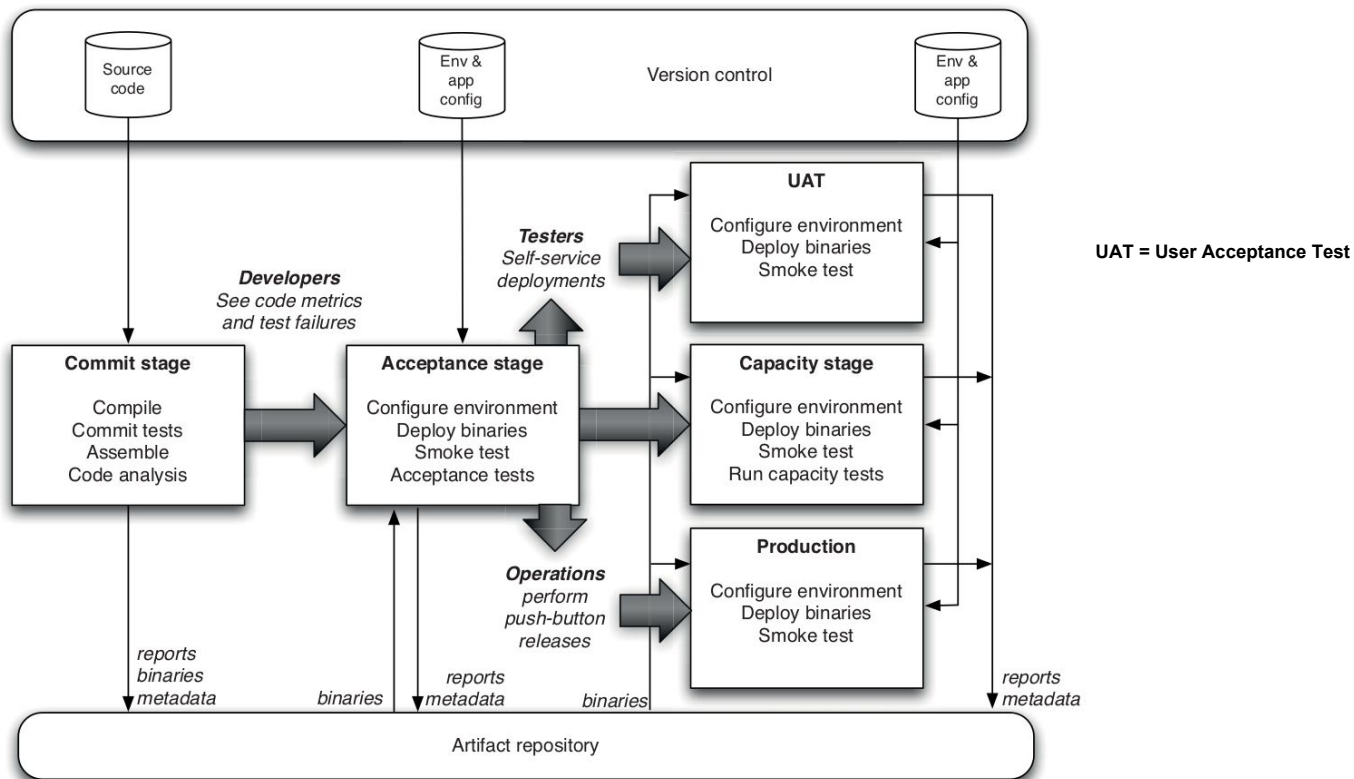
- Cela rend toute partie du processus de construction, de déploiement, de test et de mise en ligne des logiciels **visible pour tous ceux qui participent**, en aidant la collaboration.
- Il améliore les **commentaires** afin que les problèmes soient identifiés et résolus **dès le plus tôt** possible.
- Il permet aux équipes de **déployer** et de **publier** toute version de leur logiciel à n'importe quel environnement à volonté grâce à un **processus entièrement automatisé**.
- **Livraison rapide: déploiement en secondes** ou en minutes, pas des jours ou des **semaines!**



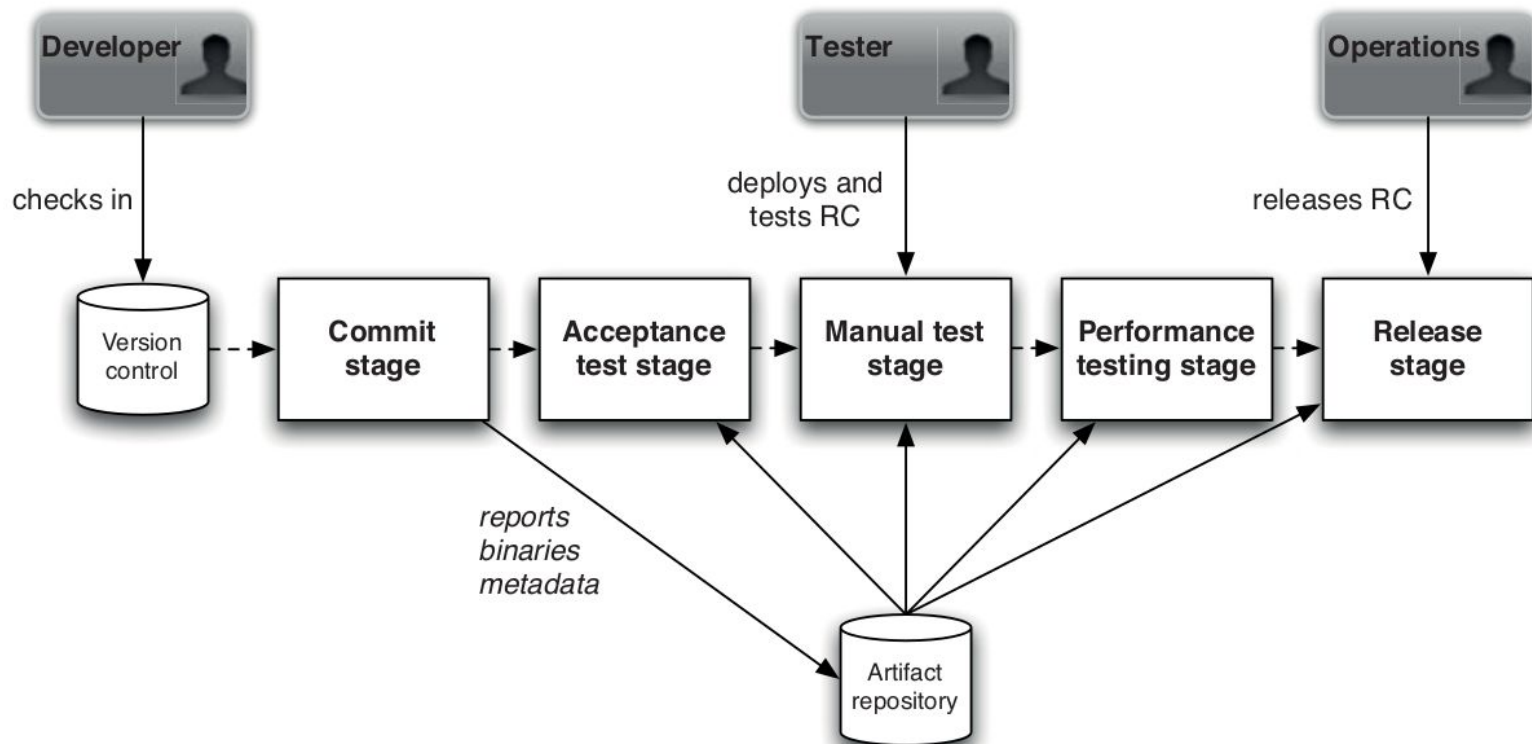
# Changements effectués dans le déploiement pipeline



# Déploiement de base Pipeline



# Le dépôt d'artefacts



# Exemple de repository artefact - Nexus

The screenshot displays the Sonatype Nexus web interface. On the left is a sidebar with navigation links: Nexus, Artifact Search, Views/Repositories, Build Promotion, Enterprise, Security, Administration, and Help. The main content area shows search results for the keyword 'struts'. A table lists various artifacts from the 'org.apache.struts' group. Below the table, the 'Component Info' tab is selected, showing details for the artifact 'struts2-core' version '2.3.16'. This section includes a tree view of repository contents, a search bar, and a detailed view of the artifact's metadata, license, and security threats. A chart on the right visualizes popularity, license risk, and security alerts across different versions.

| Group             | Artifact                      | Version           | Most Popular Version | Download                           |
|-------------------|-------------------------------|-------------------|----------------------|------------------------------------|
| org.apache.struts | struts2-gxp-plugin            | Show All Versions | 2.3.16.3             | pom, jar, sources.jar, javadoc.jar |
| org.apache.struts | struts2-embeddedjsp-plugin    | Show All Versions | 2.2.1                | pom, jar, sources.jar, javadoc.jar |
| org.apache.struts | struts2-dwr-plugin            | Show All Versions | 2.3.16.3             | pom, jar, sources.jar, javadoc.jar |
| org.apache.struts | struts2-doj-plugin            | Show All Versions | 2.3.16.2             | pom, jar, sources.jar, javadoc.jar |
| org.apache.struts | struts2-core                  | Show All Versions | 2.3.16.3             | pom, jar, sources.jar, javadoc.jar |
| org.apache.struts | struts2-convention-plugin     | Show All Versions | 2.3.16.3             | pom, jar, sources.jar, javadoc.jar |
| org.apache.struts | struts2-continuations-plugin  | Show All Versions | 2.0.5                | pom, jar                           |
| org.apache.struts | struts2-config-browser-plugin | Show All Versions | 2.3.16.3             | pom, jar, sources.jar, javadoc.jar |

Displaying Top 113 records × Clear Results

Viewing Repository: Central

Application: LSS

Group: **org.apache.struts**  
Artifact: **struts2-core**  
Version: **2.3.16**  
Declared License: **Apache-2.0**  
Observed License: **Apache-2.0**  
Effective License: **Apache-2.0**  
Highest Policy Threat: **7**  
Highest Security Threat: **5** within 2 security issues  
Cataloged: **9 months ago**  
Match State: **exact**  
Identification Source: **Sonatype**  
[View Details](#)

Popularity, License Risk, Security Alerts chart showing Older, This Version, and Newer versions.

# Avantages du déploiement pipeline

- Empowering Teams

- Les testeurs peuvent sélectionner d'anciennes versions d'une application pour vérifier les changements de comportement dans les versions plus récentes.
- Le personnel de support peut déployer une version publiée de l'application dans un environnement pour reproduire un défaut.
- Le personnel des opérations peut sélectionner une bonne version compilé connue pour déployer sa production dans le cadre d'un exercice de récupération après un sinistre.
- Les rejets peuvent être effectués en appuyant sur un bouton.

- Réduire les erreurs

- Réduction du stress

- Flexibilité du déploiement

- La pratique mène au parfait

# Principes de livraison de logiciels

- Créer un processus **répétable** et fiable pour la publication de logiciels
- **Automatiser** presque tout
- garder **absolument** tout dans la **version de contrôle**
  - Mais ne vérifiez pas les mots de passe dans le source de contrôle ou de les hardcoder dans votre application!
- Construisez la qualité dans: le test n'est pas une phase!
- **Compléter** signifie **publication**
- **Ne jamais aller à la maison en laissant une version erronée!**
- **Tout le monde** est **responsable** du processus de **livraison**
- Soyez toujours prêt à **revenir** à la version précédente
- Amélioration continue

# Intégration continue (CI)

- Si l'**intégration régulière** de votre codebase est **bon**, pourquoi ne pas le faire **tout le temps**?
- L'intégration continue est lorsque quelqu'un fait un **commit** pour n'importe quel changement, l'ensemble de l'application est compilée et un ensemble de tests automatisés sont lancés pour le vérifier.
- Si la **compilation** ou le processus de test **échoue**, l'équipe de développement arrête tout ce qu'il font et corrige le problème immédiatement.
- Le but de l'intégration continue est que le logiciel dans un **état de fonctionnement tout le temps**.
- L'intégration continue est une pratique de programmation extrême

# Mise en œuvre une intégration continue

- Accord de l'équipe
- Contrôle de version
- Un outil de construction automatisé
- Un système d'intégration



# Outils de build automatisés

- Construisez des outils pour automatiser les tâches en tant que compilation, effectuer des analyses statiques, exécuter des tests, gérer les dépendances, l'emballage, le déploiement, ...
- Exemples:
  - Make: C/C++ (typically)
  - Ant, Maven, **Gradle**: Java
    - Maven -> solution à la dépendance classpath (or jar) hell
  - Rake: Ruby
  - npm: NodeJs

# Créer un projet Java à partir de Scratch à l'aide de Gradle

- Installer Gradle  $\geq 3.4.1$  ( <https://gradle.org/install> )
- Faire un dossier de projet
  - `mkdir myproject`
  - `cd myproject`
- Projet initial
  - `gradle init --type java-library`
- Configuration pour eclipse
  - Ajouter dans le fichier `build.gradle`
    - `appliquer plugin: eclipse'`
  - `gradle eclipse`
- Exécuter des tests
  - `test le gradle`



# Jenkins

Build great things at any scale

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

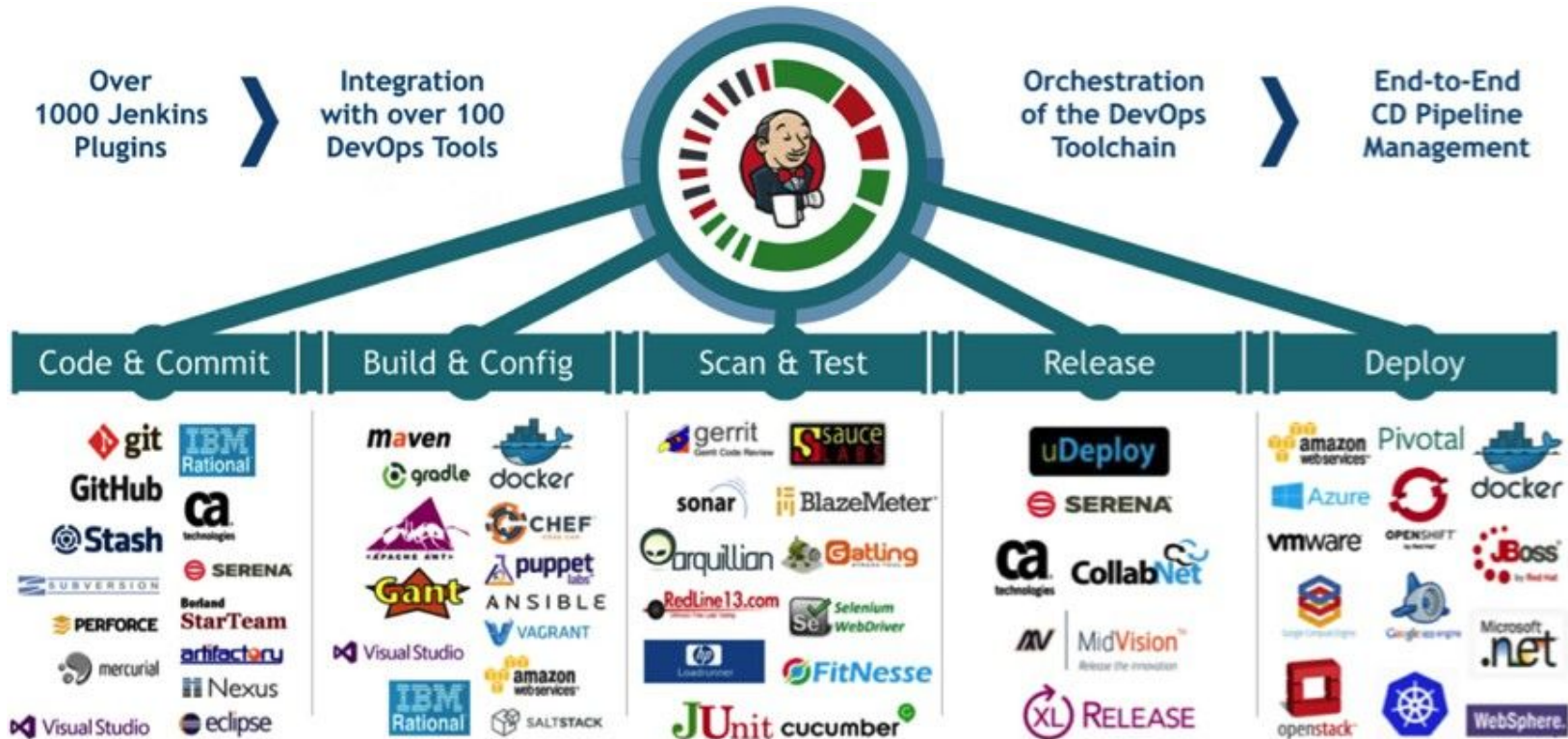
[Documentation](#)[Download](#)

## Say “hello” to Blue Ocean 1.0

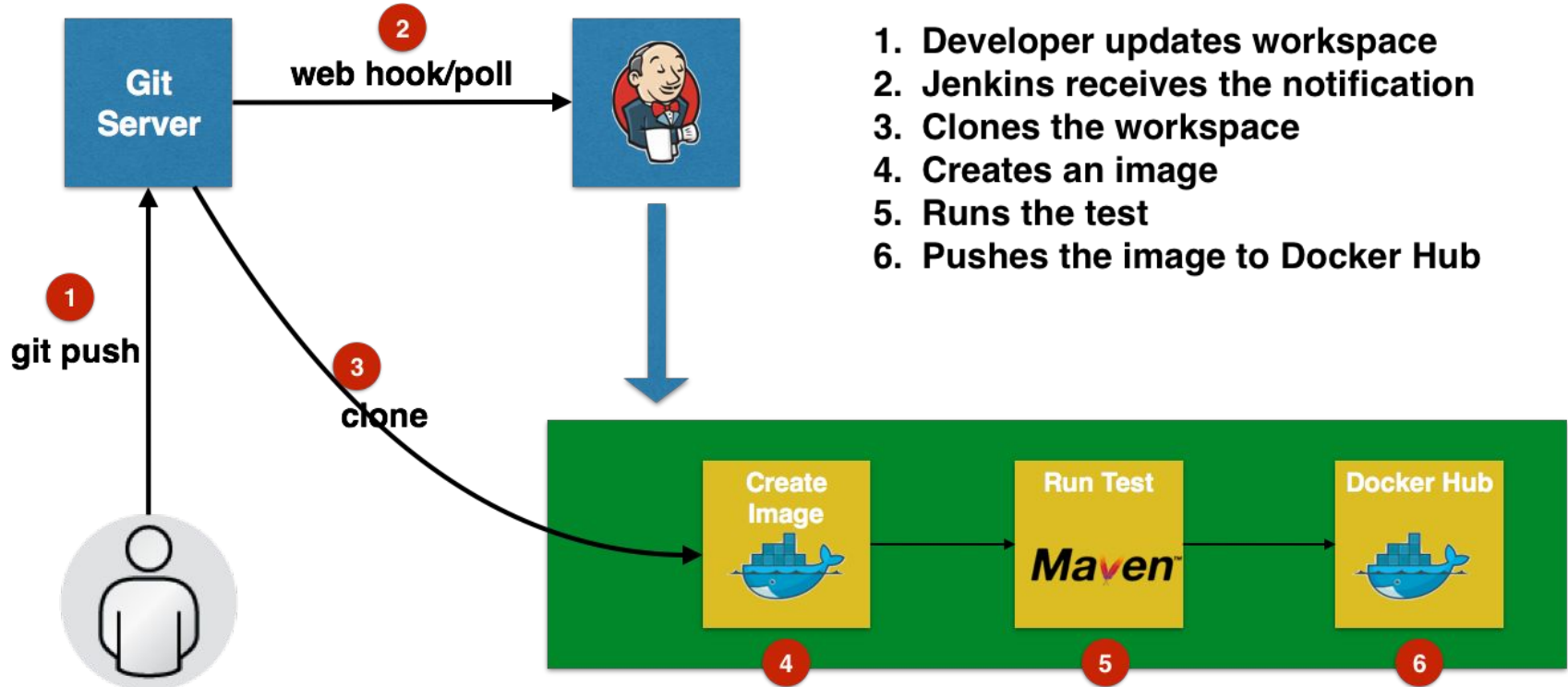
Blue Ocean is a brand new user experience that puts Continuous Delivery in reach of any team — without sacrificing the power and sophistication of Jenkins.

[Get started](#)

# Integrations Jenkins (Plugins)



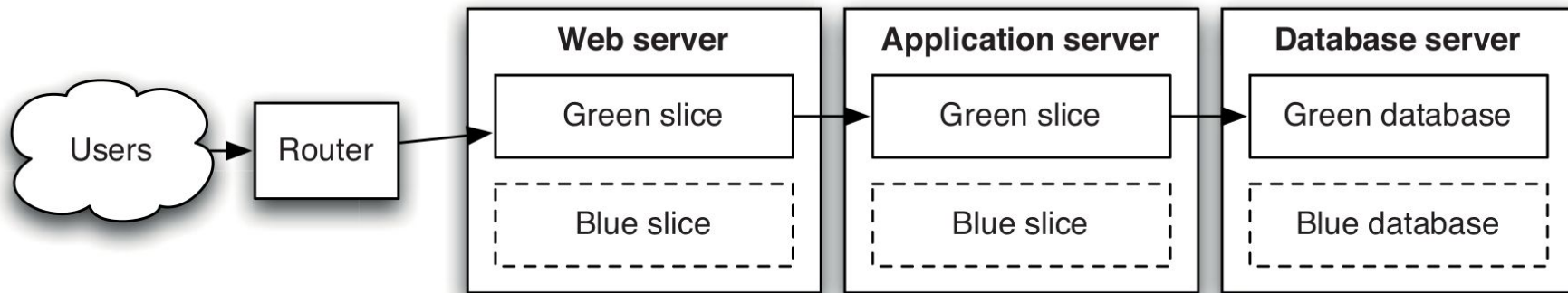
# déploiement typique de Pipeline



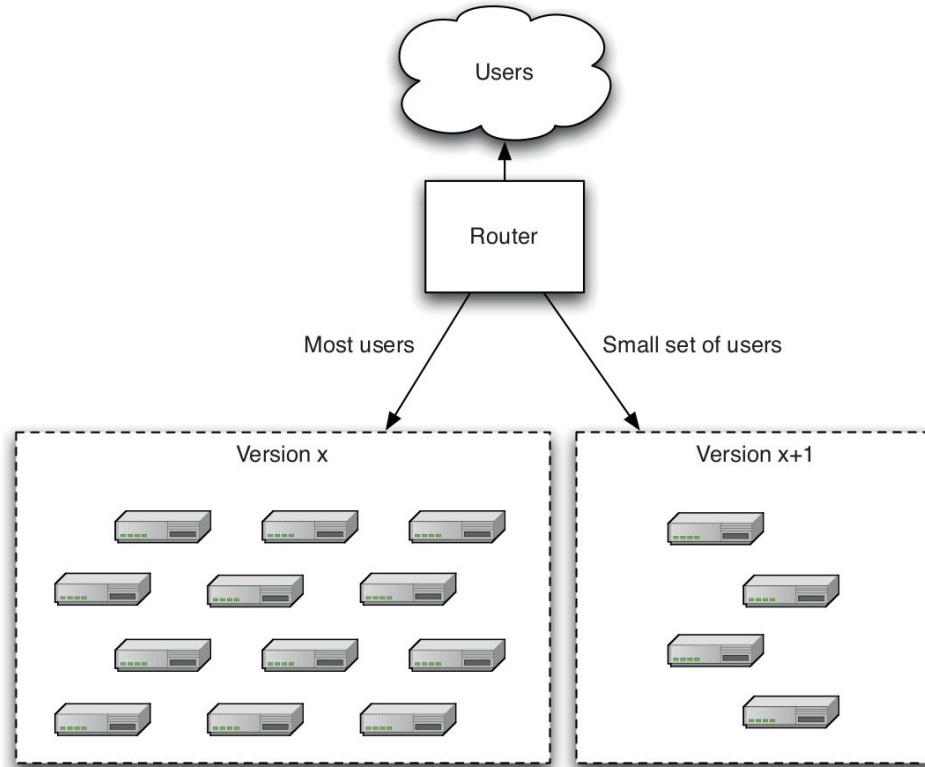
# Stratégies de publication

- Déploiements bleu-vert
- Libération des Canaries

# Déploiements bleu-vert



# Libération des Canaries





*The Addison-Wesley Signature Series*



# CONTINUOUS DELIVERY

RELIABLE SOFTWARE RELEASES THROUGH BUILD,  
TEST, AND DEPLOYMENT AUTOMATION

JEZ HUMBLE  
DAVID FARLEY



*Foreword by Martin Fowler*

Prochain cours  
revue du cours