LOG8371 : Ingénierie de la qualité en logiciel

Livraison continue

Hiver 2017

Fabio Petrillo
Chargé de Cours

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

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?f

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**

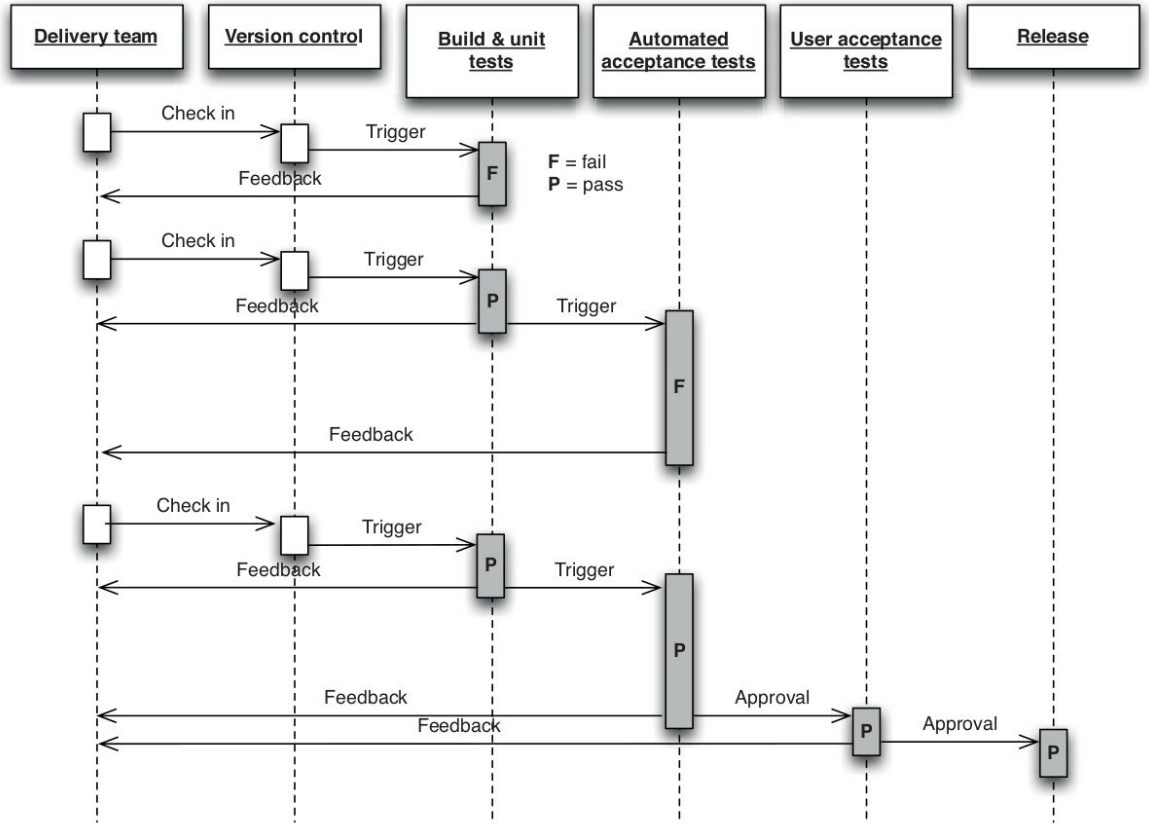
Livraison continue

- 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.

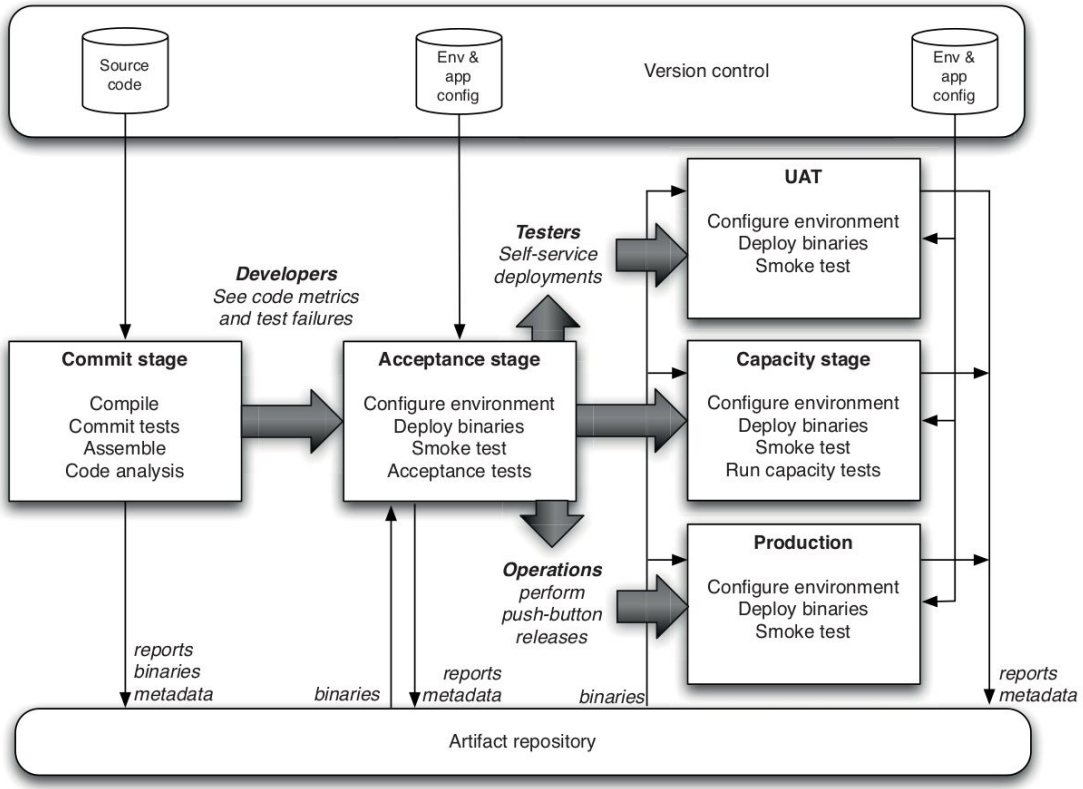
L'objectif du déploiement pipeline

- 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

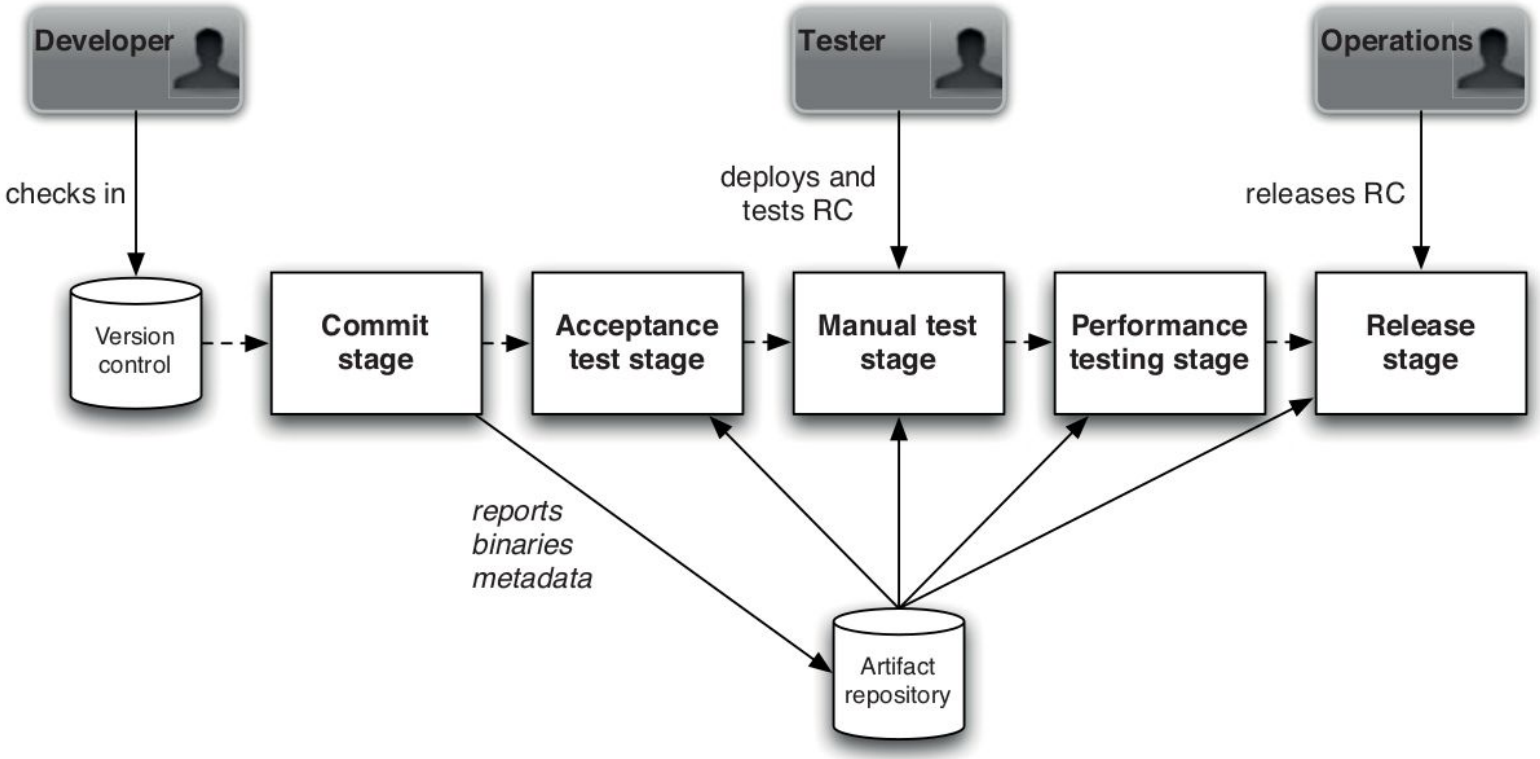


Déploiement de base Pipeline

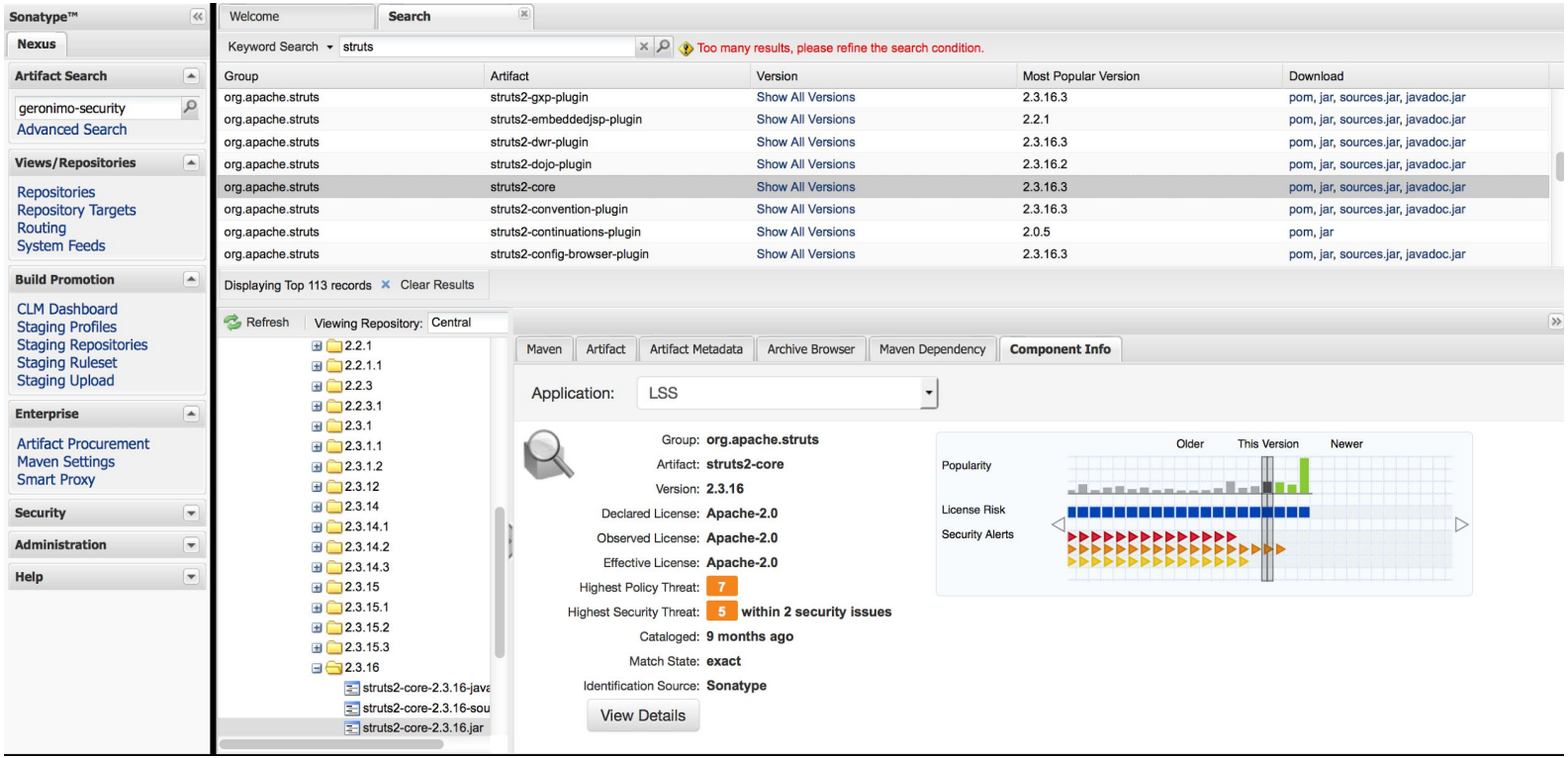


UAT = User Acceptance Test

Le dépôt d'artefacts



Exemple de repository artefact - Nexus



http://www.sonatype.org/nexus/

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 lancé pour le vérifier.
- Si la **compilation** ou le processus de test **échoue**, l'équipe de développement arrête tout ce qu'il font et corrigent 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

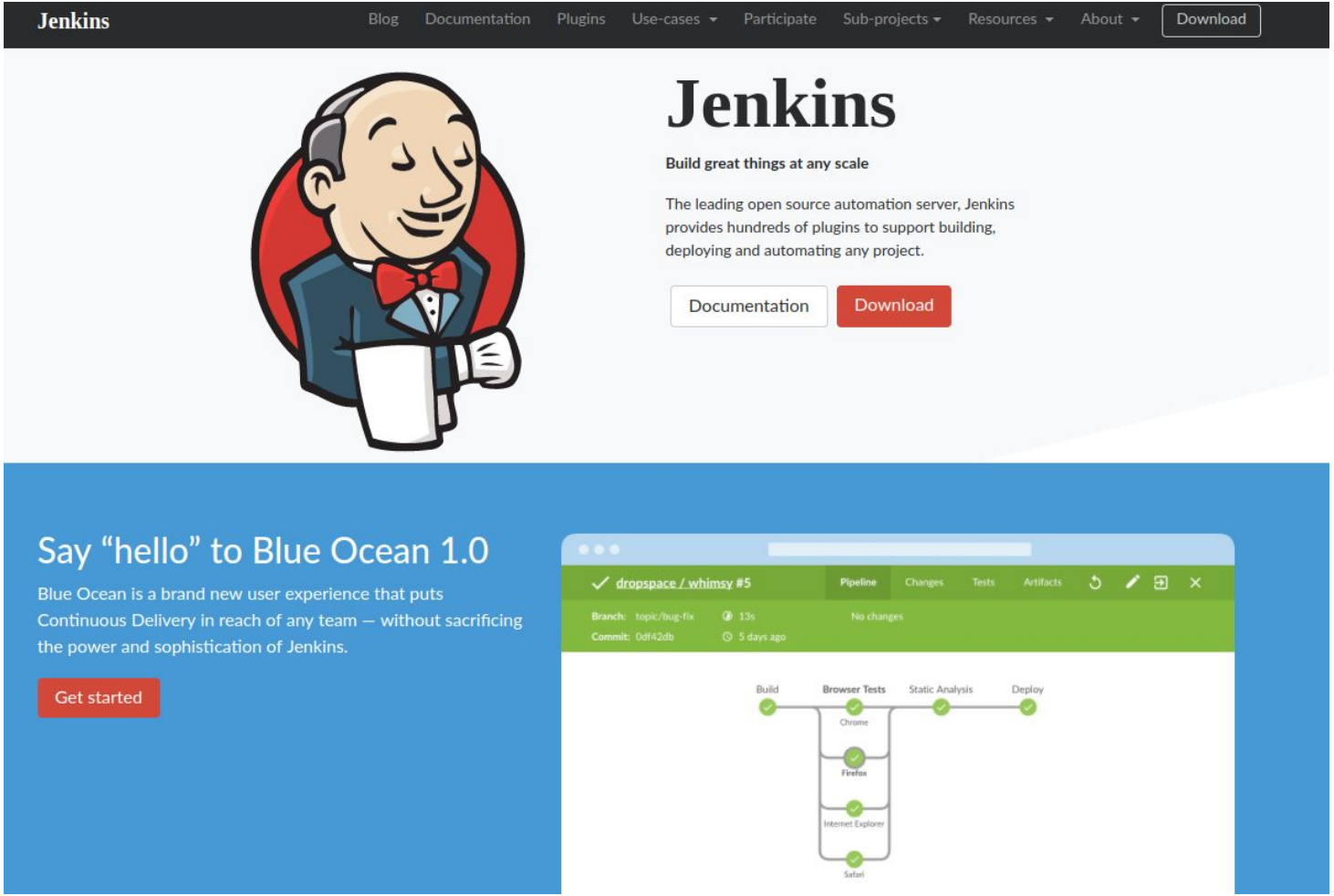
- Accord de l'équipe
- Contrôle de version
- Un outil de construction automatisé
- Un système d'intégration

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, ...
- Examples:
  - Make: C/C++ (typically)
  - Ant, Maven, **Gradle**: Java
    - Maven -> solution à la dépendance classpath (or jar) hell
  - Rake: Ruby
  - npm: NodeJs

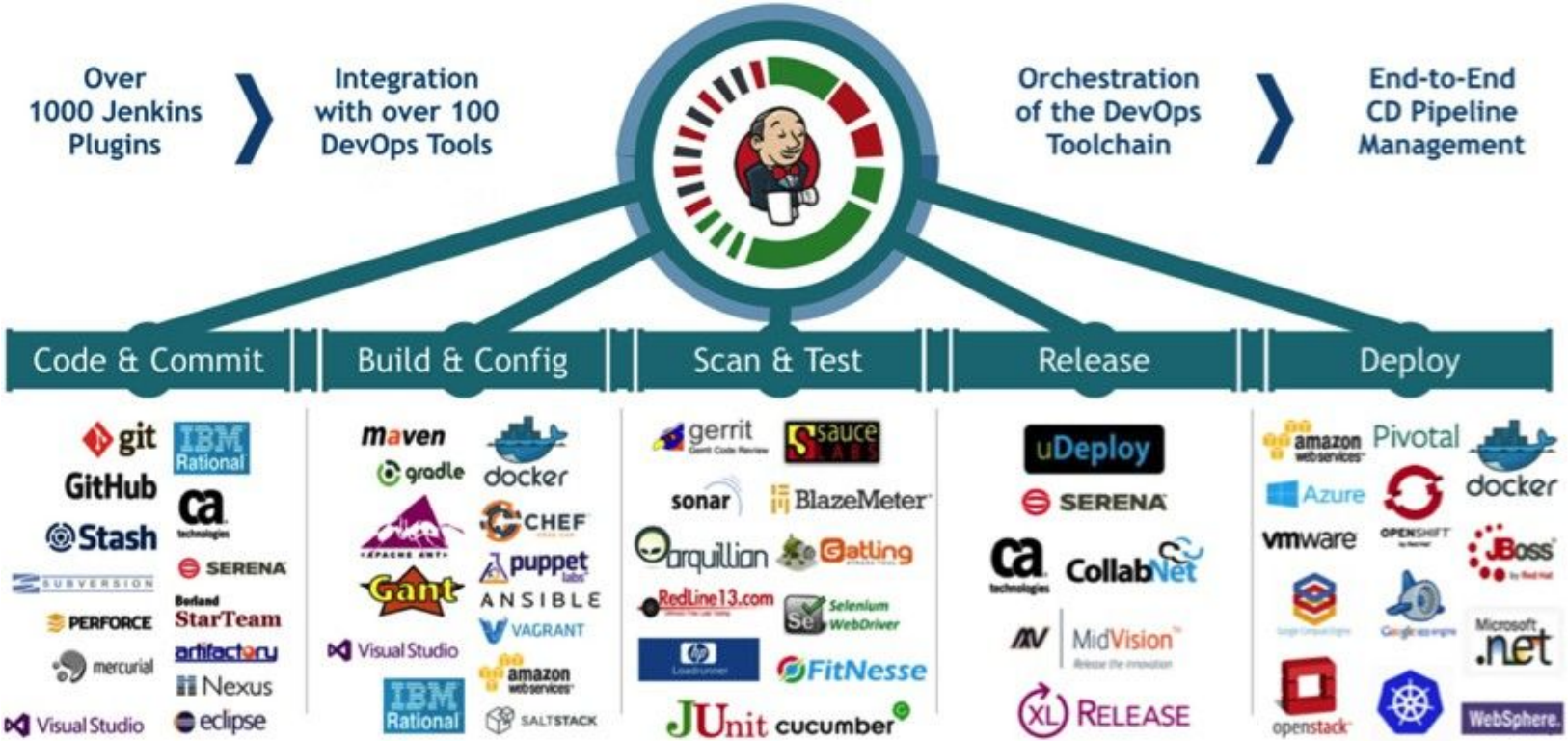
Créer un projet Java à partir de Scratch à l'aide de Gradle

- Installer Gradle >= 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

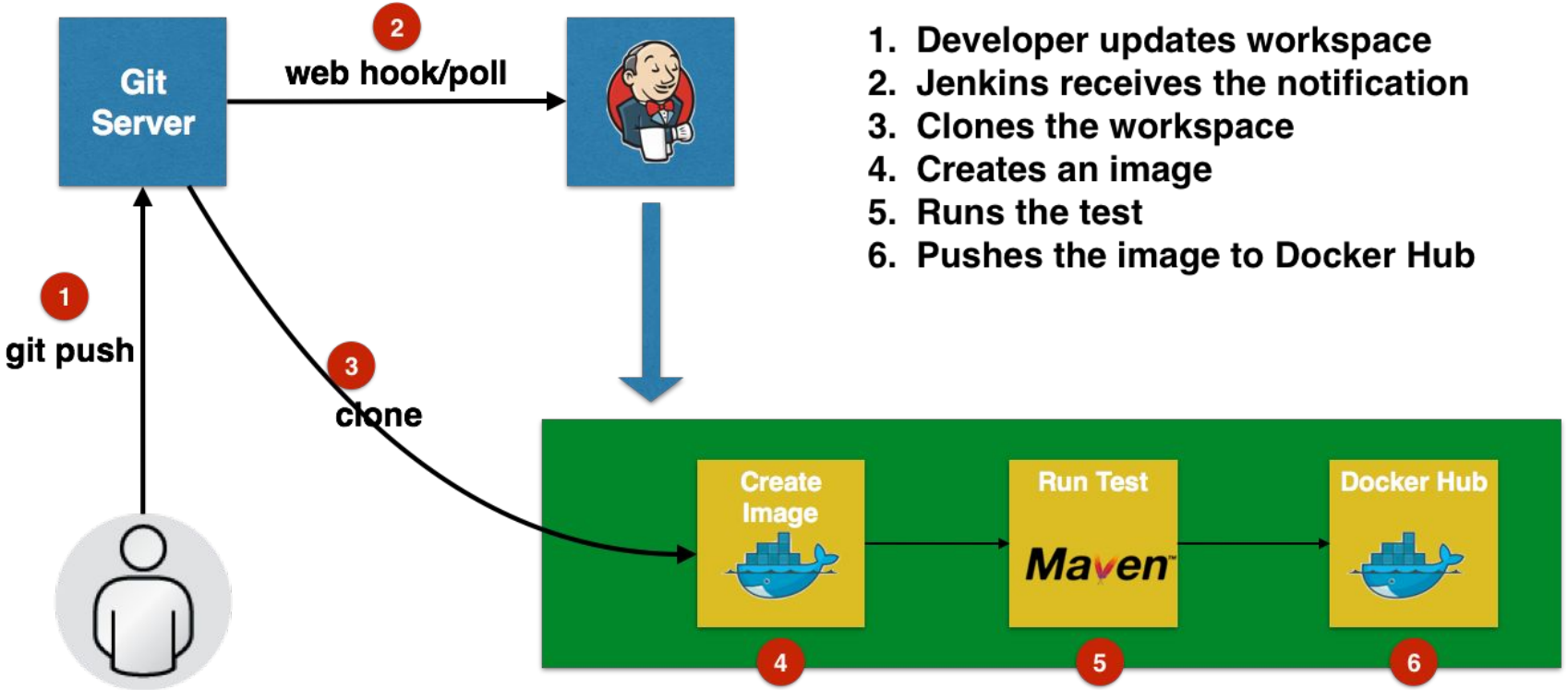


https://jenkins.io/

Integrations Jenkins (Plugins)



déploiement typique de Pipeline



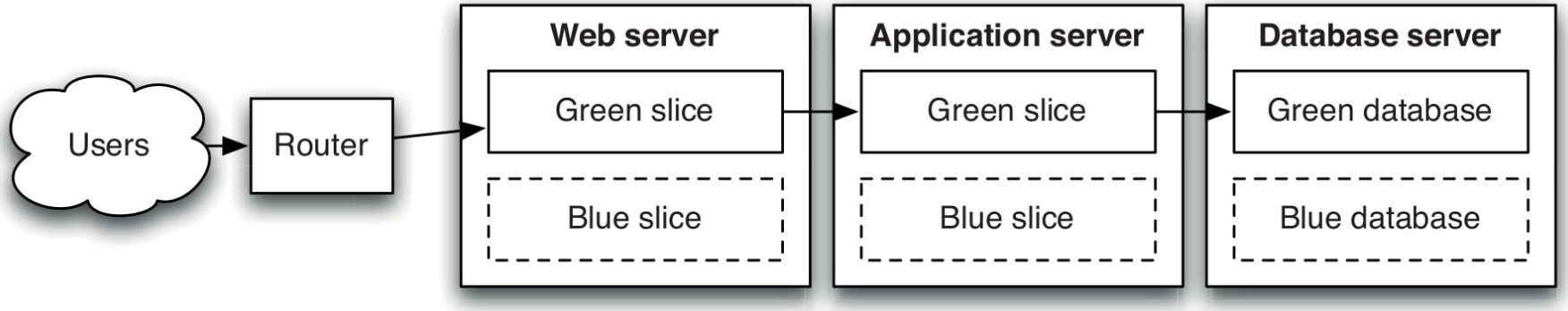
1. Developer updates workspace
2. Jenkins receives the notification
3. Clones the workspace
4. Creates an image
5. Runs the test
6. Pushes the image to Docker Hub

http://blog.arungupta.me/deployment-pipeline-docker-jenkins-java-couchbase/

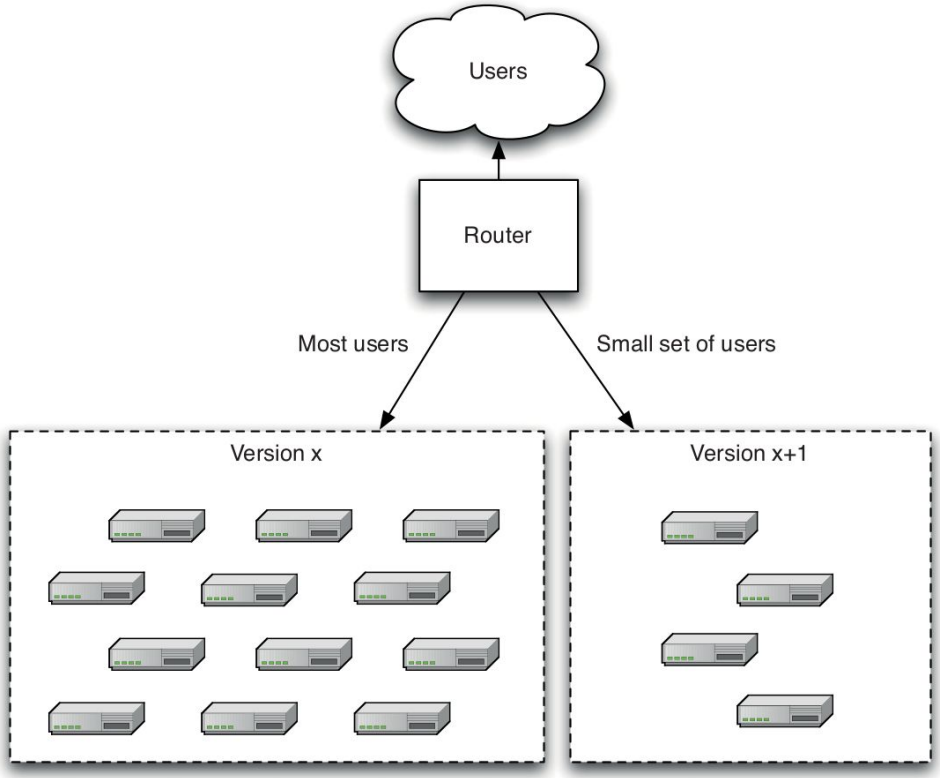
Stratégies de publication

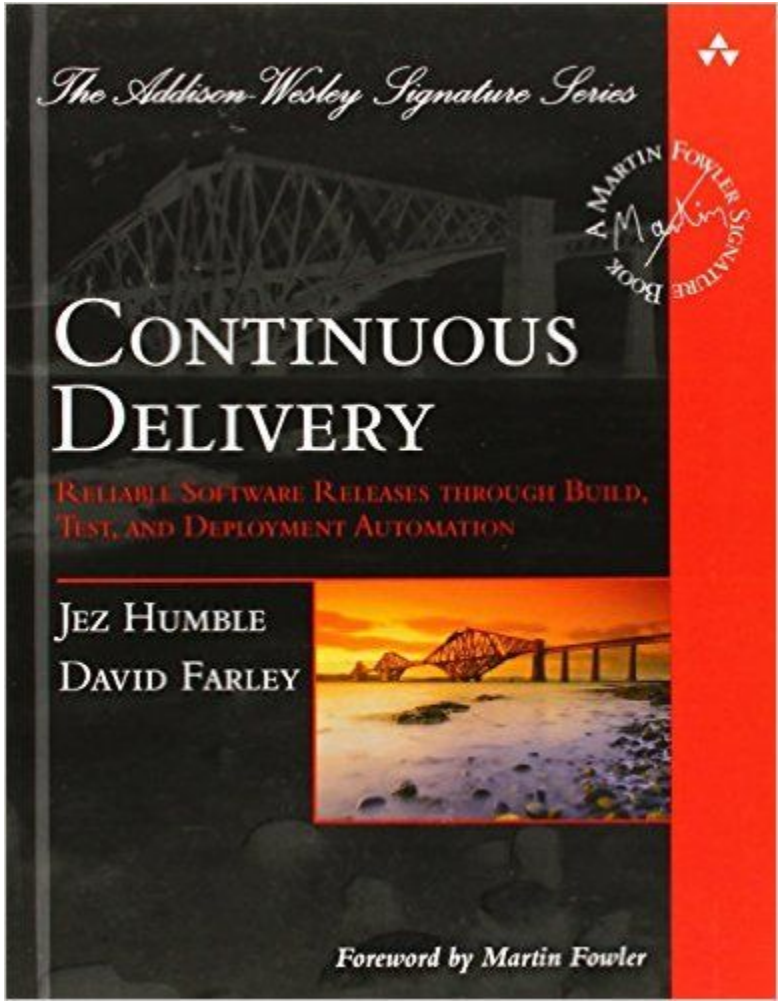
- Déploiements bleu-vert
- Libération des Canaries

Déploiements bleu-vert



Libération des Canaries




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

Prochain cours
revue du cours