Intégration Continue Avec Jenkins

- Dans le chapitre précédent, nous avons vu comment le processus de développement est suivi par les développeurs pour créer des logiciels et même maintenir ces logiciels.
- Ainsi, lorsque les développeurs écrivent du code pour créer des logiciels, ils fusionnent également tout ce code dans un référentiel centralisé ou un système de contrôle de version comme `Github`.
- Ce code est transféré dans les référentiels plusieurs par jour, et au fil du temps, tout le code est fusionné.
- Les méthodes traditionnelles de développement de logiciels ne dictent pas la fréquence ou la fréquence à laquelle vous intégrez toutes les sources d'un projet.
- Les programmeurs peuvent travailler séparément pendant des heures, des jours, voire des semaines sur la même source sans se rendre compte du nombre de conflits (et peut-être de bugs) qu'ils génèrent.

L'Intégration est Douloureuse.

- Les équipes agiles produisent du code exploitable et robuste à chaque itération.
- Tout ce code, s'il est construit et évalué renvoie de nombreux conflits, bugs et erreurs.
- Les développeurs doivent résoudre ces conflits et problèmes avant de passer à l'itération suivante.
- Plus les programmeurs partagent du code, plus cela devient problématique.
- Pour ces raisons les équipes agiles choisissent donc souvent `L'Intégration Continue`.

Quelques Terminologies.

Code Source

- Tout le code que les développeurs écrivent pour créer le logiciel est appelé `Code Source`.

Le Processus De Construction

- Il s'agit d'un processus par lequel le code source est converti en une forme autonome pouvant être exécutée sur un ordinateur.
- Par exemple, un code source écrit pour développer un logiciel `Windows` une fois construit créera un fichier `.exe ou .msi`.
- Un autre exemple, si un code `Java` est construit, il peut créer un fichier `.jar, .war, .ear`.
- Ce logiciel déployable est appelé `Artefact`.
- Le code peut être empaqueté et déployé manuellement.
- Mais il existe des outils de construction qui facilitent la vie des développeurs lorsqu'il s'agit de créer des `artefacts` ou même de les déployer.
- Ceux-ci sont appelés `Outils d'Automatisation de Construction`.

Quelques outils de build

- `Ant`
- `Maven`
- `Gradle`
- `MsBuild`
- `Nant`

Tests Unitaires

- `Les tests unitaires` vérifient que chaque unité de code (principalement des fonctions) fonctionne comme prévu.
- Le développeur, en plus d'écrire du code, écrira les cas de test qui peuvent être exécutés au moment de la construction.
- Certains cas de tests peuvent être générés automatiquement.
- L'objectif de tests unitaires est d'isoler une session de code (unité) et de vérifier son exactitude.

Qu'est-ce que l'intégration continue

- `L'Intégration Continue (CI)` est le processus d'automatisation de la création et des tests du code chaque fois qu'un membre de l'équipe valide des modifications dans le contrôle de version.
- `CI` encourage les développeurs à partager leur code et leurs tests unitaires en fusionnant leurs modifications dans un référentiel de version partagé après chaque petite tâche terminée.
- La validation du code déclenche un système de génération automatisé pour récupérer le dernier code du référentiel partagé et pour créer, tester et valider la branche principale complète (également appelée `tronc ou main`).

Le Problème

- Les développeurs écriront le code et construiront dans un système local.
- Uen fois que les développeurs ont testé le code et vérifié localement, ils le transfèrent vers un référentiel centralisé comme `github`.
- De même, tous les développeurs transmettent leur code `VCS` plusieurs par jour.
- Les développeurs travailleraient dans leurs propres silos ou grottes et continueraient à écrire le code jusqu'à ce qu'ils terminent une tâche particulière ou le projet.
- Désormais, tout le code que les développeurs ont inséré dans le `VCS`, s'il est construit et testé, renverra de nombreux conflits, erreur à cause de laquelle la construction échouera.

La Solution

- Pour contourner ce problème, chaque fois que le développeur envoie le code au `VCS`, il doit être récupéré, construit et testé par un serveur de `build` en même temps.

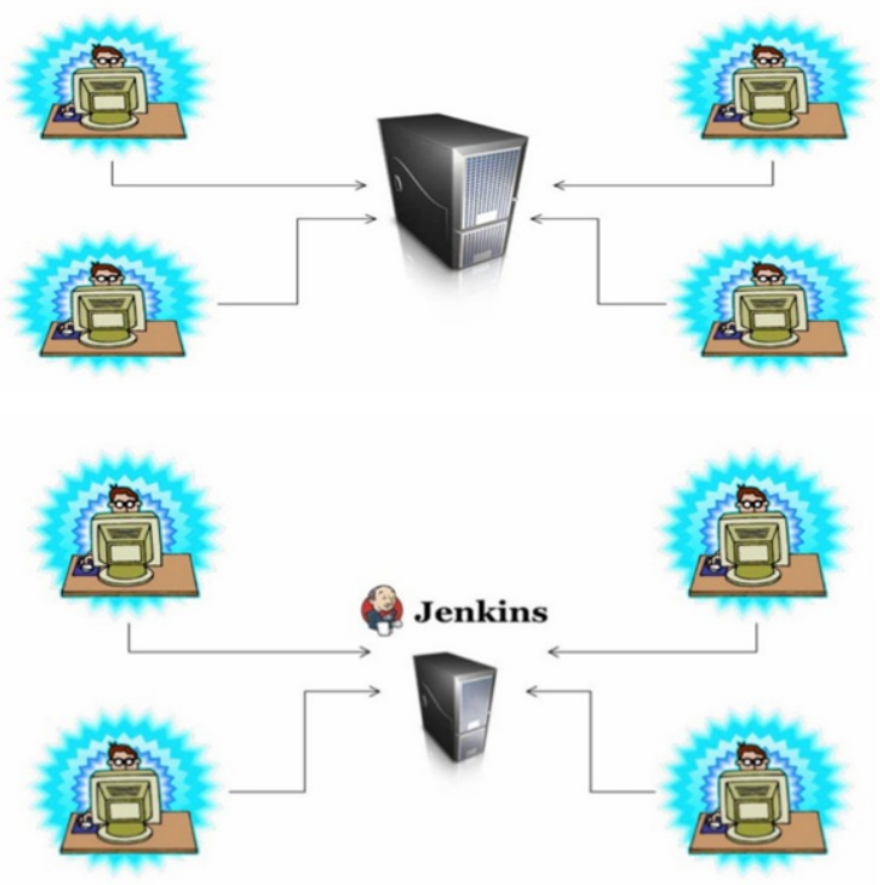

Figure 1: Alt Text

- Ce processus répété plusieurs fois par semaine ou par jour ou quotidiennement une fois est appelé `Intégration Continue`.
- Le code des développeurs est continuellement intégré, de sorte qu'à tout moment nous disposons d'un logiciel fonctionnel, s'il y a un problème dans le `processus de construction`, les développeurs seront informés par `e-mail` et résoudront le problème.

Qu'est-ce que Jenkins

- `Jenkins` est un `serveur d'intégration continue` qui récupérer le dernier code de `VCS`, le construire, le tester et le notifier aux développeurs.
- `Jenkins` peut faire bien plus de choses en plus d'être simplement un `serveur CI`.
- Il s'appelait à l'origine `Hudson`, `Oracle Inc.`possède désormais `Hudson`
- `Jenkins` est un projet open source écrit par `Koshuke Kawagauchi`.
- `Jenkins` est un serveur d'applications `Web basé sur Java`.
- Comme condition préalable, nous devons d'abord configurer `Java` sur des machines pour ex&cuter le serveur `Jenkins`.

Caractéristiques de Jenkins

Open Source

- Comme `Jenkins` est open source, il y a de nombreuses contributions partout dans le monde au logiciel `Jenkins`.
- Il possède toutes les fonctionnalités les plus récentes et les plus intéressantes que les développeurs y intègrent régulièrement.

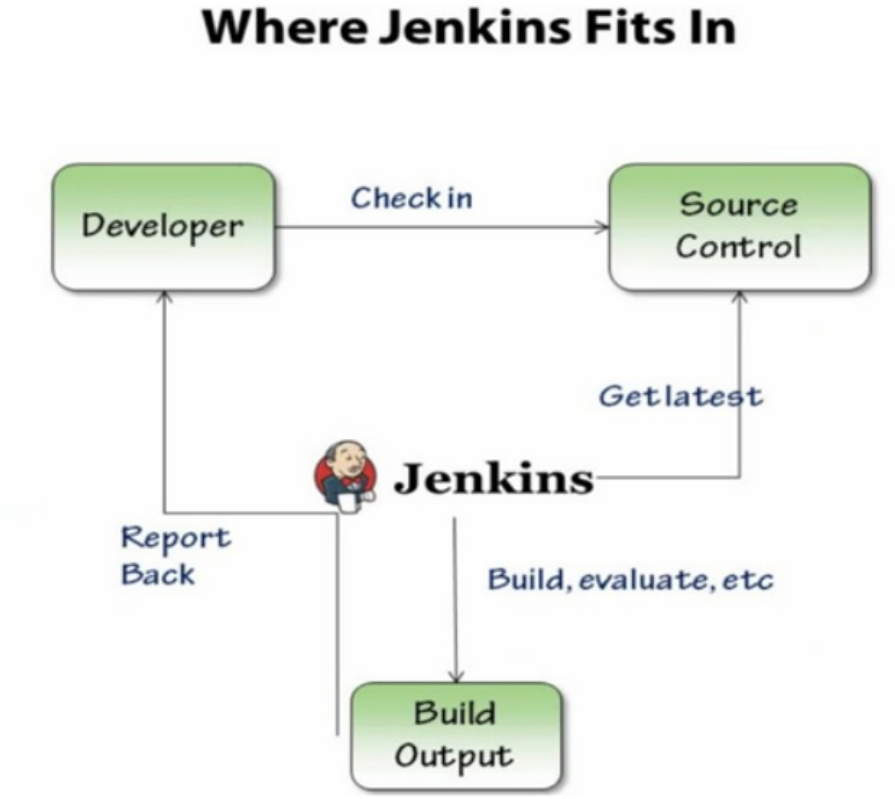

Figure 2: Alt Text