**Remerciements**

Nous présentons, premièrement, nos sincères remerciements à nos chères familles pour leur soutien tant moral matériel. Nous ressentons une profonde gratitude pour leurs encouragements qui…. !

Deuxièmement nous disons merci à notre encadreur du projet, MSC NIRAGIRA Ferdinand pour son aide incomparable. Sans lui, Nous n’y aurons peut-être pas arrivé.

**Liste de figures**

Fig. 0.1 Comparaison de l’integration classique et l’integration continue

Fig. 1.1 Détail d’un système d’IC

Fig. 1.2 Gestionnaire de version centralisé [3]

Fig. 1.3 Système de gestion de version distribué [3]

Fig. 1.4 processus d’exécution de builds avec Maven

Fig. 2.1 les opérations dans un système de gestion de version avec GIT

Fig. 2.2 création d’un dépôt sur GitHub

Fig. 2.3 Ajout d’un collaborateur

Fig. 2.4 Réponse a une invitation de collaboration sur un dépôt GitHub

Fig. 2.5 Initiation du dépôt git

Fig. 2.6 Page d’accueil de GitHub Desktop

Fig. 2.7 Choix du dépôt à cloner

Fig. 2.8 Processus de clonage encours

**LISTE DES ABREVIATIONS ET SIGLES**

**IC**: Intégration Continue ou **CI:** Continuous Integration

**VCS:** Version Control System

**DVCS**: Distributed Version Control System

**CVS:** [Concurrent Versions System](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Concurrent_Versions_System&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhiVeQxsGpovkktAtb65fH-0t4P1uQ)

**MOTS CLES**

Builds[[1]](#footnote-1), Commit, Update, Checkout, Intégration, Intégration continue, [Tests](http://dico.developpez.com/html/1139-Gestion-de-projet-tests.php) unitaires, Test de validation, Test d’intégration, Push, Pull, Clone, Repository, Bug,

**CHAPITRE 0 : INTRODUCTION GENERALE**

L’informatique est une science comprenant plusieurs domaines en évolution continue depuis que celui-ci a vu le jour vers les années 1950. L’un des domaines de l’informatique sur lequel on va s’intéresser dans ce travail est le développement de logiciel.

Le développement de logiciel comprend un ensemble des étapes qui se suivent pour la mise en place d’un logiciel de sa création à sa disparition (retrait du logiciel). Ces étapes constituent le cycle de vie du logiciel. Celui-ci comprend généralement à minima les activités suivantes[[2]](#footnote-2):

* **Définition des objectifs**, consistant à définir la finalité du projet et son inscription dans une stratégie globale.
* **Analyse des besoins et faisabilité**, c'est-à-dire l'expression, le recueil et la formalisation des besoins du demandeur (le client) et de l'ensemble des contraintes.
* **Conception générale**. Il s'agit de l'élaboration des spécifications de l'architecture générale du logiciel.
* **Conception détaillée**, consistant à définir précisément chaque sous-ensemble du logiciel.
* **Codage** (Implémentation ou programmation), soit la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.
* **Tests unitaires**, permettant de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.
* **Intégration**, dont l'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de *tests d'intégration* consignés dans un document.
* **Qualification** (ou *recette*), c'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.
* **Documentation**, visant à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.
* **Mise en production**,
* **Maintenance**, comprenant toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel.

Le développement d’un logiciel se fait souvent en équipe. Le partage du code devient donc une tâche difficile et complexe parce qu’on doit, d’une part trouver les erreurs syntaxiques par la compilation de l’ensemble du code combiné et corriger des erreurs syntaxiques d’autre part, pour s’assurer du bon fonctionnement de l’application après la combinaison du code.

Toutes ces tâches de vérification et de correction d’erreurs sont très complexes à tel point qu’elles peuvent prendre un temps à peu près égal au temps de codage si elles sont faites manuellement. D’où l’importance de l’intégration continue parce que cette technique permet d’effectuer toute ces opérations automatiquement. Cela permet l’amélioration de la qualité du logiciel ainsi qu’un très grand gain en temps et en argent.

Dans ce document, il sera abordé les notions d’intégration et d’intégration continue, son fonctionnement et ses intérêts pour un programmeur en général et pour une maison de programmation en particulier. Seront également présenté, les différents outils disponibles sur le marché pour la mise en place d’un système d’intégration continue.

**0 1. JUSTIFICATION DU CHOIX DU SUJET**

Dans la justification du choix, on va parler de ce qui a attiré notre attention à ce sujet et ce qui nous a poussés à le faire comme projet ainsi que son importance pour nous. On parlera aussi de l’intérêt académique du sujet ainsi que l’importance de ce sujet aux bénéficiaires.

1. **1.1. Motivation et intérêt personnel du sujet**

C’est très récemment que nous avons appris l’existence de l’intégration continue, même si cette technique n’est pas du tout jeune. Elle a tout de suite attiré notre attention car Elle coïncide avec l’un de nos besoins de développeurs : automatiser les tâches répétitives au cours du développement de logiciels. On a pris l’opportunité que nous offre le programme de baccalauréat de réaliser un projet de fin du cycle pour entrer en profondeur du sujet.

Notre plus grande motivation a été de se rendre compte que, malgré l’importance de cette pratique dans le développement de logiciel, on n’en savait presque rien. Cela est peut-être dû au fait que l’intégration continue ne faisait pas partie d’aucun des programme d’enseignement : secondaire ou Universitaire, mais aussi qu’elle n’était pas utiliser dans beaucoup de maisons d’édition de logiciel dans notre pays.

L’intérêt de sujet pour nous est sans précédent. L’intégration continue va radicalement révolutionner notre manière habituelle de réalisation de projet informatique.

1. **1. 2. Pertinence scientifique du sujet : intérêt académique**

L’importance de sujet est capitale dans le domaine informatique en générale et dans le secteur de la programmation en particulier. Notre recherche sur ce sujet va permettre de mettre à disposition un document permettant de découvrir l’IC et pouvant servir de guide pour ceux qui voudront adopter cette pratique. Ce document sera donc une référence pour quiconque faisant de recherche à ce sujet.

* 1. **3. Pertinence sociale du sujet : intérêt des bénéficiaires**

L’intérêt des bénéficiaires est que, pour ceux qui faisaient encore l’intégration manuellement (généralement à la fin du projet), on va proposer, vers la fin de ce document, une solution complète d’intégration continue opérationnelle et fiable. De plus, d’autres outils seront présentés ; à tel point qu’ils pourront faire leur choix. L’adoption de cette pratique facilitera considérablement le processus d’intégration souvent longue et complexe. En effet, le système d’IC exécute automatiquement l’ensemble de taches répétitives, ce qui permet de gagner en argent et en temps. C’est également un moyen fiable et optimal de produire un logiciel de qualité.

* 1. **PROBLEMATIQUE DE LA RECHERCHE**

Le processus de développement d’un logiciel comprend plusieurs étapes et l’une d’elles est l’intégration. Elle se fait souvent après l’étape de codage et consiste à assembler tous les modules constituants le projet. Ses modules sont souvent développés séparément par différents membres d’une équipe de développement. Imagiez un ensemble de modules codes par une équipe de 10 développeurs pendant u mois cela exigera presque une équipe pareille pour pouvoir tout assembler et résoudre les problèmes qui vont avec l’intégration.

Dans beaucoup de maisons d’édition de logiciel, l'intégration peut durer des heures, des jours, des semaines, voire même des mois, tout dépend du volume du projet. La cause est que l’intégration est effectuée manuellement à la fin du codage.

On peut alors se poser ces quelques questions :

* Est-il possible de réduire le temps de l'intégration ?
* Si c’est possible, comment ?
* Avec quels outils ?

C’est à la réponse de ces questions que ce travail sera centré.

* 1. **HYPOTHESES DE LA RECHERCHE**

Nous supposons qu’il est possible de réduire le temps et l’effort que prend l’intégration. Nous faisons également hypothèse que des moyens et outils pratiques existent qui permettent de rendre l’intégration rapide.

* 1. **OBJECTIFS DE LA RECHERCHE**

1. **Objectif global**

L’objectif global de ce projet est d’installer et configurer un serveur regroupant divers outils couramment utilisés dans le cadre d’utilisation des procédures d’intégration continue et de qualité de code.

1. **Objectifs spécifiques**

* Faire un état des lieux sur l’IC
* L’installation et la configuration d’un serveur d’intégration continue
* L’installation et la configuration d’un serveur de gestion de version
* L’installation et la configuration d’un serveur de base de données
* L’installation et la configuration d’un serveur d’application

0. 5. **DELIMITATION DU SUJET**

1. **Délimitation dans le temps :** Nous pensons que l’intégration continue subsistera tant qu’il y aura de la programmation. Le sujet est donc perpétuel.
2. **Délimitation dans l’espace :** les systemes d’IC concernent un programmeur solitaire voire toute une maison d’édition de logiciel. Toute équipe de développement professionnelle, quel que soit sa taille, devrait mettre en œuvre l'IC.
3. **Délimitation dans le domaine :** Ce sujet s’inscrit dans le domaine de l’informatique en générale et dans programmation en particulier.

**0. 6. CADRE CONCEPTUEL ET THEORIQUE**

1. **Quelques définitions**

* **Intégration** est definit comme un ensemble des étapes nécessaires à la production de toutes les données nécessaires à un projet et leur assemblage pour rendre le projet fonctionnel.[1]
* **L’intégration continue**: est une pratique de développement de logiciels où les membres d'une équipe intègrent fréquemment leur travail, généralement chaque personne s'intègre au moins une fois par jour, ce qui entraîne des intégrations multiples par jour. [2]
* **Un Logiciel de gestion de** [**versions**](https://fr.wikipedia.org/wiki/Version_d%27un_logiciel) (**VCS**) ou **Un gestionnaire de version** est un système qui enregistre l’évolution d’un fichier ou d’un ensemble de fichiers au cours du temps de manière à ce qu’on puisse rappeler une version antérieure d’un fichier à tout moment.[3]
* [**Tests**](http://dico.developpez.com/html/1139-Gestion-de-projet-tests.php) **unitaires :** Test d’un bloc (unité) de programme (classe, méthode, etc.)[4]
* **Test de validation**: est un type de [test informatique](https://fr.wikipedia.org/wiki/Test_(informatique)) qui permet de vérifier si toutes les exigences client, décrites dans le document de [spécification](https://fr.wikipedia.org/wiki/Sp%C3%A9cification_(informatique)) du [logiciel](https://fr.wikipedia.org/wiki/Logiciel), sont respectées
* **Repostory**: Endroit où sont stockés les différents artefacts, les librairies, etc.
* **bug** ou **bogue** est un défaut de conception d'un programme informatique à l'origine d'un dysfonctionnement [2].
* **Commit** : C’est l’opération qui permet la validation des mises à jour du code source existant sur le répertoire de travail local de la machine du développeur moyennant l’outil de gestion de code source  (tel que SVN). Le commit se fait du répertoire de travail local vers le référentiel de l’outil de gestion de configuration.
* **Update** : C’est l’opération qui permet de la mise à jour à partir du référentiel de l’outil de gestion de configuration du répertoire local.
* **Checkout** : C’est l’opération d’extraction d’une version d’un projet en cours de développement du référentiel du gestionnaire de configuration sur un répertoire de travail local.
* **Pipeline :**

**Apache Maven** est un outil de build très largement utilisé dans le monde Java, avec de nombreuses et puissantes fonctionnalités telles que la gestion déclarative des dépendances, le principe de convention plutôt que configuration et un très grand nombre de plugins. [7]

1. **Approche théorique**

* **L’intégration classique et l’intégration continue**

L’intégration classique intervient dans la dernière phase du développement du logiciel. Cette Phase consiste à assembler les modules constituant le projet. Ces modules sont déjà testés unitairement. C’est après leur assemblage que s’effectue les tests d’intégration puis les tests de validation.

Par contre l’intégration continue fait l’assemblage, les tests unitaires et autres test plusieurs fois durant le processus même de développement. Chaque enregistrement est ensuite vérifié par une construction automatisée, ce qui permet aux équipes de détecter rapidement les problèmes.

**Intégration Continue**

**Intégration classique**

Fig 0.1 Comparaison de l’integration classique et l’integration continue

* **Les composant d’un système d’IC**
* Un serveur d’intégration continue
* Un logiciel de gestion de version du code source
* Une équipe de développeurs

1. **Deroulement du processus d’IC**[[3]](#footnote-3)

**Etape 1**

Le développeur code son module puis il réalise des tests unitaires sur sa machine afin de s'assurer que tout fonctionne correctement dans son environnement. Il récupère la dernière copie du code sur le gestionnaire de code source pour mettre à jour son code. Il fusionne son code avec le code qu'il a récupéré puis résout les conflits, teste de nouveau son code sur sa machine et apporte des corrections éventuelles. Si tout est correct, il publie alors son code via son gestionnaire de code source.

**Etape 2**

Le serveur d'intégration possède un service de détection de modification de code. Suite à la dernière publication du développeur, il prépare une tâche qui consiste à récupérer le dernier fragment de code développé et à l'intégrer dans sa plateforme. Il exécute cette tâche appelée communément Job.

**Etape 3**

Une fois le job terminé, des rapports portant sur la qualité, la stabilité ou encore les divers bugs pouvant être rencontrés, sont générés et transmis à l'ensemble de l'équipe ou juste au développeur.

**Etape 4**

L'équipe peut alors consulter ces rapports, les analyser, traiter des bugs s'il y en a, puis continuer à développer les autres phases du projet.

**Avantages de l'intégration continue [[4]](#footnote-4)**

Les principaux avantages d'une telle technique de développement sont :

* Le test immédiat des modifications ;
* La notification rapide en cas de code incompatible ou manquant ;
* Les problèmes d'intégration sont détectés et réparés de façon continue, évitant les problèmes de dernière minute ;
* Une version est toujours disponible pour un test, une démonstration ou une distribution.
* Cette technique permet de gagner en productivité, en temps, en argent, mais aussi en qualité de code.
* Cela permet d'avoir une bonne vision du logiciel, notamment sur les différents points forts et points faibles du code ou de l'équipe.

**0.7. DEMARCHE SUIVIE POUR LA CONDUITE DU PROJET**

**CONCLUSION DU PREMIER CHAPITRE**

L'Intégration Continue, aussi connue sous le terme IC, est l'un des piliers du développement logiciel moderne. En fait, elle est un véritable tournant quand l'Intégration Continue est mise en place dans une organisation, elle change radicalement la manière dont les équipes pensent le processus de développement. Elle est capable de permettre et d'induire toute une série d'améliorations et de transformations, depuis le build régulier automatisé jusqu'à la livraison continue en production. Une bonne infrastructure d'IC peut fluidifier le processus de développement jusqu'au déploiement, aide à détecter et corriger les bogues plus rapidement, fournit un écran de contrôle très utile aux développeurs mais aussi aux non-développeurs, et poussée à l'extrême, elle permet aux équipes de fournir plus de valeur métier aux utilisateurs finaux. Toute équipe de développement professionnelle, quel que soit sa taille, devrait mettre en œuvre l'IC. [7]

Malgré tout, cette bonne pratique n’est pas d’usage dans la plupart des maisons opérant au Burundi. La cause serait l’ignorance de son existence ou un manque de personnel avec une formation à son sujet, capable de l’appliquer. Nous ferons dans ce travail l’état des lieux de solutions existantes pour une mise en place d’IC.

**CHAPITRE II: SPECIFICATIONS FONCTIONNELLES ET CONCEPTUELLES DE LA PLATEFORME IC**

1. **Détails du processus d’intégration continue [5]**

En s‘appuyant sur le schéma suivant, voici les détails du processus d’intégration continue :

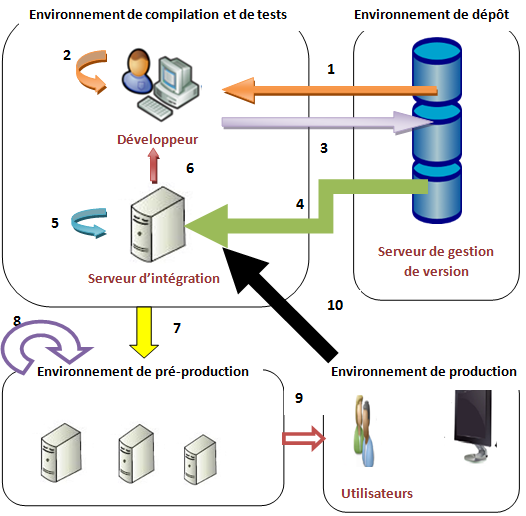


Fig1.1 Détail d’un système d’IC

* + - 1. Chaque développeur, souhaitant apporter des changements à une partie du code source, doit d’abord faire une importation une copie de la dernière version de l’application à partir du *repository*, (effectuer un *check out*)
      2. Une fois les changements apportés au code source, le développeur doit lancer un *build* privé pour s’assurer qu’une une version saine de l’application est conservée.
      3. Suite au succès du *build* privé, le développeur effectue un *commit* pour exporter la nouvelle version vers le *repository*.
      4. Grâce à un contrôle régulier du *repository*, le serveur d’IC détecte l’enregistrement de la nouvelle version comprenant les changements effectués et l’importe.
      5. L’intégrateur lance ensuite un script pour exécuter un *build* d’intégration.
      6. Des feedbacks sont générés par la suite par le serveur d’IC pour avertir les développeurs du succès ou de l'échec du *build*. Dans ce dernier cas, ils sont informés des bugs détectés et doivent procéder à leur correction en reprenant les étapes de 1 à 3.
      7. Suite au succès du *build* d’intégration, l’application sera déployée dans l’environnement de pré-production.
      8. Des tests fonctionnels et de charge de l’application seront effectués dans l’environnement de pré-production.
      9. L’application pourrait finalement être déployée dans l’environnement de production pour tester aussi bien que qualifier l’application.
      10. En cas de détection de bugs ou de problème de performance, une notification sera envoyée aux acteurs de l’environnement de compilation et de test.

1. **Spécifications fonctionnelles et conceptuelles de la plateforme** 
   1. **Fonctionnalités nécessaires pour l’intégration continue**

* **La gestion et contrôle de versions :** Un des outils de production essentiel dans l'[environnement de développement](https://www.pulsar-informatique.com/creation-site-internet/comment-creer-un-site-internet/comment-gerer-son-projet-de-site-web/outils-de-gestion-de-projet-web/environnements-de-dev)est la mise en place d'un outil de contrôle et gestion de version (versioning). En effet quand plusieurs personnes travaillent sur un même projet et manipulent les mêmes fichiers il est courant que certains fichiers soient manipulés en même temps par différents acteurs du projet. Pour éviter des conflits ou des écrasements de fichiers à cause de ce travail collaboratif des outils de gestion version sont apparus. Ils permettent que chacun dispose sur son poste d'une version locale du projet qu'il peut modifier à souhait. Des mécanismes de mise à jour permettent de fusionner les différentes versions de chacun en gérant les conflits. Ces outils de versioning permettent aussi de revenir poste par poste à une version précédente quand un conflit éclate.[[5]](#footnote-5)

L'une des caractéristiques des systèmes de contrôle de version est qu'ils vous permettent de créer plusieurs branches pour gérer différents flux de développement. C'est une fonctionnalité utile, voire essentielle, mais elle est souvent surutilisée et provoque des problèmes. [2]

En gros, Il permet de ramener un fichier à un état précédent, ramener le projet complet à un état précédent, comparer les changements au cours du temps, voir qui a modifié quelque chose qui pourrait causer un problème, qui a introduit un problème et quand, et plus encore. Utiliser un VCS signifie aussi généralement que si vous vous trompez ou que vous perdez des fichiers, vous pouvez facilement revenir à un état stable. Deplus, vous obtenez tous ces avantages avec une faible surcharge de travail.[3]

* **L’exécution des builds :** Une grande construction prend souvent du temps, vous ne voulez pas faire toutes ces étapes si vous avez seulement fait un petit changement. Un bon outil de construction analyse donc ce qui doit être changé dans le cadre du processus. La façon courante de procéder consiste à vérifier les dates des fichiers source et objet et à compiler uniquement si la date source est postérieure. Les dépendances deviennent alors difficiles: si un fichier d'objet change ceux qui en dépendent, il peut aussi avoir besoin d'être reconstruit. Les compilateurs peuvent gérer ce genre de chose, ou ils ne le peuvent pas. [2]
* **Le suivi et qualification des performances :** cette fonctionnalité permet de dégager des métriques concernant l’analyse du code source. Il est possible, par exemple, de vérifier que les lignes n'excèdent pas une certaine longueur, que les standards de nommage sont bien respectés et que le code est bien commenté. Il permet également d’obtenir des rapports d’exécution des tests de performances, des indicateurs sur la couverture du code et même de pouvoir détecter des codes morts (non-utilisés).[[6]](#footnote-6)
* **La gestion des bugs**[[7]](#footnote-7)**:** La gestion des bugs, ou plus exactement des défauts, varie selon les projets. Même si l'objectif ultime avec une méthode agile est de ne pas avoir de défauts dans le code, dans la vraie vie des projets il y a toujours des défauts. Et il faut s'en occuper, en gardant à l'idée que c'est moins cher de les corriger tôt que tard.
  1. **Etude des composants de la plateforme pour la solution**

1. **Outils de contrôle et gestion des versions**
2. **Classification**

* **Système de gestion de version centralisé**

Ils fonctionnent suivants le modèle **client-serveur**. Les développeurs utilisent un référentiel unique partagé.

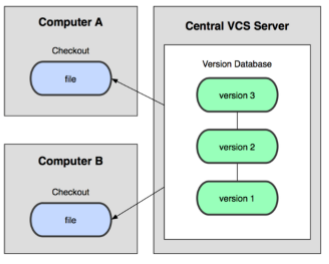


Fig1.2 Gestionnaire de version centralisé [3]

Le défaut le plus visible de ce système est le point unique de panne que le serveur centralisé représente. Si ce serveur est en panne pendant une heure, alors durant cette heure, aucun client ne peut collaborer ou enregistrer les modifications issues de son travail. Si le disque dur du serveur central se corrompt, et s’il n’y a pas eu de sauvegarde, vous perdez absolument tout de l’historique d’un projet en dehors des sauvegardes locales que les gens auraient pu réaliser sur leur machines locales. [3]

Il est aujourd’hui disponible sur le marché beaucoup de logiciels de contrôle et de gestion de versions open source(Subversion, CVS, [CVSNT](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/CVSNT&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg-klcnnaYRPCoDUZ1_IJFNnJz0sg), [OpenCVS](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/OpenCVS&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhl8a24Sddh4JiO1ZS5a6eaWYk-sw), [Vesta](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Vesta_(Software_configuration_management)&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjiYlIhLVSjZMNb1auAAg9bCJuwzQ), etc.…) et d’autres propriétaires([IC Gérer](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/IC_Manage&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhj1njsxb2uPrenb_JzULZH1j4k7wA), [PTC Integrity](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/PTC_Integrity&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhiSKdd5bXk2vEKEG44ByrtZeovwyw), [StarTeam](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/StarTeam&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjdP4DpYsXzRUlxulXjpRHOg3gyQg), [AccuRev](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/AccuRev_SCM&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjttvu3nuqNPFjK8DvFP3j9pYVi5A), [Panvalet](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Panvalet&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg0DzKA9Hpf2tuWqAdc2zoQ1xNoJw) , [Team Foundation Server](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Team_Foundation_Server&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhiNflzf74LOqjoaJgWYYftNWQy0zA) (TFS), etc.)

* **Système de gestion de version décentralisé(Distribué)**

Dans ce genre de système chaque développeur travaille directement avec son propre référentiel local. Ce référentiel est en réalité une copie du référentiel central existant sur le serveur. Les modifications sont partagées entre sur le dépôts central et les membres doivent mettre à jour leur référentiel en dupliquant le dépôt sur serveur central.

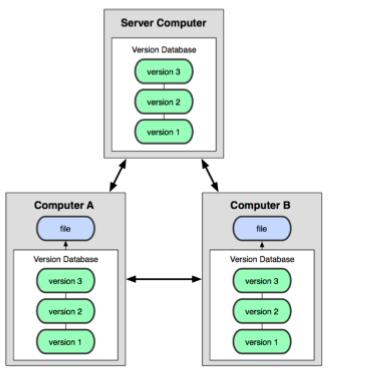


Fig1.3 Système de gestion de version distribué [3]

Il y a de DVS gratuits ([Git](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Git_(software)&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg3SBQq09KcQGQ66ERprFlpeJsr2Q), [Mercurial](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Mercurial_(software)&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhilpzIyl3e13bxvu1TUXo0nRckgSg), [Bazar](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Bazaar_(software)&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhlIeq4wLB4OAlWHXSTbTgTwEkBpQ), [BitKeeper](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/BitKeeper&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjuFhw3UFbFUCK6AkE05w_8nUxA4A), [Codeville](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Codeville&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjQsA9ByNHf-UX7_vwJI5-qweEGwA), [DCVS](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Distributed_Concurrent_Versions_System&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhh2_mRlXNYiHaAiAqxKKwVKoaXM0Q) - décentralisé et basé sur CVS, [GNU arch](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/GNU_arch&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjDfI7hAfaJWTb03EKKdR61NHVEZA), [Veracity](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Veracity_(software)&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjq1nk43NoKFz1nZV9Wx012RTFi2Q)) et d’autres payants([Code Co-op](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Code_Co-op&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhR2NBgZzqPrvpvmff_vzCqTwpSGg), [Sun WorkShop TeamWare](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Sun_WorkShop_TeamWare&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg4UNw7O9biOeQX8isDLWOQL7CSJg), [Plastic SCM](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Plastic_SCM&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhiNgUa2Je6PH1tkPms5xjpcDI1zlw), [Visual Studio Team Services](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://en.wikipedia.org/wiki/Visual_Studio_Team_Services&xid=17259,15700022,15700105,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhZ28Y1bh968XB6ISjWsuoeQXph4Q))

1. **Description de quelques-uns des outils de gestion de version[[8]](#footnote-8)**

### Git



Git est l’un des logiciels de gestion de versions qui permettent d’offrir aux développeurs des fonctionnalités pour le travail collaboratif. Il a été Créé [Linus Torvald](https://fr.wikipedia.org/wiki/Linus_Torvalds), en 2005.Git est populaire en partie grâce au site Internet [GitHub](https://www.github.com/). Il permet le partage des informations en proposant un système de gestion de bugs (ou issue tracking system), des propositions de modification (ou pull request), un gestionnaire de tâche, un wiki, des graphiques (soumission, contribution, fréquences, …), etc. De 2008, l’année de son lancement a 2016, GitHub possède 14 millions d’utilisateurs et pas moins de 35 millions de dépôts, ce qui en fait le plus grand répertoire de code source au monde.

### Mercurial SCM



Tout comme Git, Mercurial SCM est un logiciel de gestion de versions qui monte. Disponible sur Linux et Windows, il est publié sous [licence GNU / GPL](https://fr.wikipedia.org/wiki/Licence_publique_g%C3%A9n%C3%A9rale_GNU) depuis 2005, l’année de son lancement. Mercurial est basé sur le même principe de décentralisation que

Git. Il propose des fonctionnalités somme toute proches de ses concurrents, tout en invoquant sa grande simplicité et rapidité d’utilisation. Mercurial est capable de gérer de gros projets, sans avoir à utiliser de serveur mais plutôt son interface web, dans sa gestion des branches et des fusions.

* Apache Subversion, ou SVN



Subversion est un logiciel libre développé par l’[Apache Software Foundation](https://www.apache.org/) depuis 2000. Remplaçant de l’illustre logiciel [Concurrent Versions System](https://fr.wikipedia.org/wiki/Concurrent_versions_system) ou CVS, Subversion propose en substance les mêmes fonctionnalités que Git et Mercurial pour le travail en équipe.

### Git vs. Mercurial vs. Subversion[[9]](#footnote-9)

1. **La centralisation :** Git et Mercurial sont décentralisés. Lorsque vous les utilisez, tout est hébergé à la fois sur le serveur mais aussi sur votre ordinateur. Vous avez accès à toutes les données du dépôt puisqu’elles sont téléchargées en intégralité sur votre machine. Pour Subversion qui est un système centralisé, ce n’est pas le cas. Vous devez vous connecter au serveur de dépôt.
2. **Le stockage :** chaque système gère et stocke ses données d’une manière différente. Tandis que Git prône l’utilisation des métadonnées dans un répertoire nommé .git, où sont situées toutes les ressources téléchargées (tags, branches, historique des versions, etc.), SVN lui ne stocke que les fichiers. SVN utilise donc une grande espace
3. **Les branches :** l’utilisation des branches est de loin plus facile avec Git et Mercurial qu’avec Subversion. Git et Mercurial gèrent tout et savent quel fichier doit être fusionné ou non. Pour Subversion en revanche, les branches ne sont que la résultante d’un répertoire sur le dépôt. Pour fusionner des branches, vous devrez utiliser des commandes spéciales.
4. **Les révisions :** Subversion et Mercurial utilisent une numérotation unique pour chaque révision des fichiers. Ce n’est pas le cas pour Git.
5. **La protection :** le contenu présent dans Git et Mercurial est crypté grâce à l’algorithme SHA-1 (pour Secure Hash Algorithma ou algorithme de hachage sécurisé). Cela permet d’éviter la corruption d’un dépôt à la suite d’un problème d’envoi par Internet ou le crash d’un disque dur. Ce n’est pas le cas pour Subversion.

**Tenant comptent de tous les facteurs vu ci-dessus et de ce dont on a besoin pour ce projet, on a choisi d‘utiliser Git.**

1. **Outils de tests automatisés et d’exécution des builds** 
   1. **Les outils de test de performance [5]**

* **TestMaker** est un outil approprié pour tester les applications Web, les systèmes de courrier électronique (e-mail), les applications basées sur l’architecture SOA (Service Oriented Architecture) et les applications Java. Il supporte de multiples protocoles notamment HTTP, HTTPS, SOAP.
* **JMeter** permet de tester les performances (temps de réponse et fiabilité des réponses aux requêtes) de sites Web, de serveurs FTP, TCP/IP et de services Web, de bases de données accessibles via jdbc, de scripts Perl et d’objets JAVA (Applets). Il fait partie de la suite Apache JAKARTA. Il est entièrement écrit en Java, ce qui lui permet d'être utilisé sur tout système d'exploitation supportant une machine virtuelle Java (JVM).
* **Dieseltest** est un outil de test de charge permettant de simuler plusieurs utilisateurs sur une application ou un site Web. L'enregistrement des scripts par cet outil est facile et les résultats sont fournis en temps réel.
* **OpenSTA** est un logiciel dédié aux tests de performance développé en C++. Il permet de simuler des centaines d'utilisateurs virtuels d’un site ou d’une application Web et de recueillir ensuite des mesures de performance et des analyses fines du système testé.

1. **Outil de test unitaire JUnit : [5]**

**JUnit** est une bibliothèque de test unitaire pour le langage Java. Il est intégré par défaut dans les environnements de développement intégré Java tels que BlueJ, Eclipse et Netbeans.

Cet outil définit deux types de fichiers de tests :

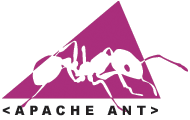
* Les TestCase sont des classes Java héritant de la classe junit.framework. Ils contiennent un certain nombre de méthodes de tests et servent généralement à tester le bon fonctionnement d'une classe.
* Une TestSuite permet d'exécuter un certain nombre de TestCase déjà définis.

1. **Outils d’exécution des builds[[10]](#footnote-10)**

Le langage Java est très répandu, et par conséquence, le nombre d'outils pour réaliser telle ou telle tâche est généralement élevé. Pour ce qui est de l’execution de buils,on peut citer : Ant, Maven, Gradle, GAnt, EasyAnt, etc. Toutefois, seule une poignée d'entre eux sont vraiment connus et réputés.

* 1. **Présentation de quelques outils d’exécution de builds**

# Ant



## Présentation

[Ant](http://ant.apache.org/) est un projet open-source de la [fondation Apache](http://www.apache.org/). Ant repose sur un fichier XML (le build.xml) qui est un ensemble de cibles (ou targets) ayant chacune un rôle bien précis dans la construction du projet : nettoyage de répertoires, initialisation, compilation, transfert de fichiers, etc. Chaque cible est décomposée en un ensemble de tâches, chaque tâche étant destinée à réaliser une opération particulière. Ainsi, la tâche <javac .../> a pour rôle de compiler des classes Java, la tâche <copy .../> copiera un ensemble de fichiers, etc.

## Avantages

* Intégré dans la quasi-totalité des IDE.
* Supporté nativement par les outils d'Intégration Continue.
* Connu pratiquement par tout le monde, il est aussi enseigné dans les écoles.
* De nombreuses tâches sont disponibles.
* On peut tout faire avec Ant.
* Plutôt facile à prendre en main, relativement intuitif.
* Grande flexibilité.

## Inconvénients

* Devient vite verbeux.
* Au final, cela reste un script (amélioré, certes).
* Pas de conventions dans l'écriture du XML, ce qui amène de la complexité dans le script.
* Aucun mécanisme de gestion des dépendances.
* Création complexe et répétitive d'un processus complet de création d'un artefact JAR ou WAR (compilation, tests, package...).

# Ivy



## Présentation

Le projet [Apache Ivy](http://ant.apache.org/ivy/index.html) est en réalité une extension à **Ant** permettant avant tout de disposer d'un véritable mécanisme de gestion des dépendances.

* **Avantages**
* Se base sur **Ant**, l'un des outils les plus utilisés et connus en Java.
* Moins verbeux que **Maven 2** (en ce qui concerne la déclaration des dépendances).
* Permet de générer des rapports.
* Gère les dépendances transitives.
* Peut utiliser le repository de **Maven 2**.
* **Ivy** peut être invoqué par des API, et donc hors d'**Ant**.
* Propose des services supplémentaires par rapport à **Maven** en fournissant par exemple une gestion intelligente des exclusions, la possibilité de désactiver la transitivité des dépendances (y compris pour les dépendances **Maven**), ainsi qu'un algorithme de résolution de conflits plus avancé.
* Contrairement à **Maven**, il est possible de générer plusieurs artefacts différents pour un même projet.

## Inconvénients

* Ne se charge que des dépendances et ne résout pas les autres lacunes d'**Ant**. Toutefois, ce ne sont pas là les objectifs d'**Ivy**...
* Faible popularité, et évolution relativement lente.

# Maven



## Présentation

### Historique

* + **Maven 1**, développé en 2002 en tant que sous-projet du projet [Apache Turbine](http://turbine.apache.org/), est devenu un projet Apache à part entière en 2003. Toutefois, cette version n'a pas rencontré un succès notoire.
  + **Maven 2**, est apparu fin 2005. Reprenant les bases de **Maven 1**, il a été complètement refondu et a corrigé nombre de lacunes de son prédécesseur. Il est depuis devenu l'un des outils les plus utilisés pour réaliser la construction de projets Java. C'est d'ailleurs généralement l'outil adopté lors du démarrage d'un nouveau projet.
  + Le processus est celui-ci :[[11]](#footnote-11)

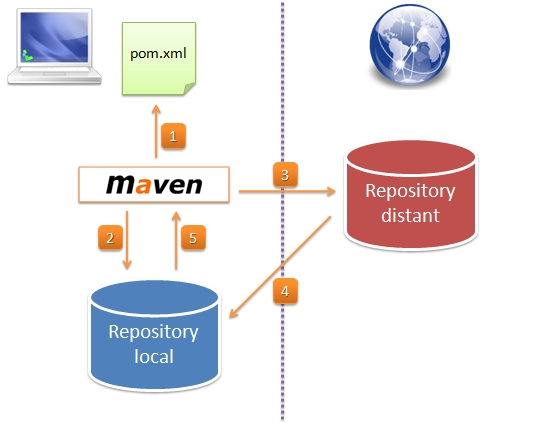
[](http://linsolas.developpez.com/articles/java/outils/builds/images/maven-dependencies.jpg)

Fig1.4 processus d’exécution de builds avec Maven

1. Maven commence par définir la liste des dépendances nécessaires au projet, via la lecture du pom.xml du projet.
2. Maven interroge alors le repository local afin de trouver les dépendances utilisées.
3. Si la dépendance n'est pas trouvée, alors Maven va interroger les repositories distants.
4. Les dépendances absentes du repository local sont alors téléchargées depuis les repositories distants, de telle façon à ce qu'elles soient disponibles lors des prochains builds.
5. Maven peut alors utiliser la dépendance pour la construction du projet.

## Avantages

* Utilisation des conventions **Maven**, permettant de raccourcir le pom.xml.
* La gestion des dépendances, en particulier les dépendances transitives.
* L'héritage de projets.
* De nombreux plugins sont aujourd'hui disponibles.
* Possibilité de réaliser de nombreuses tâches avec un pom.xml simple (compilation, packaging, site, rapports).
* Supporté par la majorité des outils d'Intégration Continue.
* Relative facilité de création de nouveaux plugins.
* Les archetypes, assez nombreux, facilitant la création de nouveaux projets.
* Il existe plusieurs gestionnaires de repositories locaux, les plus utilisés étant [Nexus](http://nexus.sonatype.org/), [Archiva](http://archiva.apache.org/) et [Artifactory](http://www.jfrog.org/products.php).

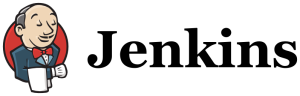
## Inconvénients

* Dès que l'on veut sortir un peu du système **Maven 2**, que l'on veut faire des choses en dehors du cadre de l'outil, ça devient vite complexe.
* Le pom.xml est trop verbeux, par exemple pour la définition dépendances.
* Trop grande utilisation de plugins, y compris pour réaliser certaines tâches simples.
* Documentation "officielle" peu fournie et assez fouillie. Heureusement, plusieurs ouvrages complets (et pour certains gratuits) existent !
* XML parfois redondant, même en utilisant l'héritage.
* Support pas encore parfait dans les IDE, bien que les choses s'améliorent.
* Développement et support des plugins "officiels" inégaux.
* Manque de souplesse, de rigueur sur certains principes (difficile de sortir du cycle de vie par exemple).

1. **Serveurs d’IC**

Il existe aujourd'hui de nombreux serveurs d’IC. Les plus rependus sont : [Jenkins](http://jenkins-ci.org/), [Cruise Control](http://cruisecontrol.sourceforge.net/), [Hudson](http://hudson-ci.org/), [Continuum](http://continuum.apache.org/), [Team Foundation Server](http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx), Codefresh, Codeship, CircleCI, GitLab CI

* **Description de quelques outils d’intégration continue [6]**
* **Jenkins**

[](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://jenkins-ci.org/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhh6guonvOCkvKG0eNzYsp2UDT86Cw)

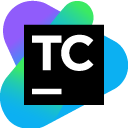
Jenkins est un outil CI open-source écrit en Java. Il est originaire de la [fourche de Hudson](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://jenkins-ci.org/blog/2011/01/11/hudsons-future/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhib5J5KGH8Jjzh0alBOpyZYW1iHbQ) lorsque l'Oracle a acheté le Sun Microsystems. Jenkins est un outil d’IC multiplateforme et il offre une configuration à la fois via l'interface graphique et les commandes de la console.

Ce qui rend Jenkins très flexible est l'extension de fonctionnalité à travers les plugins. [Jenkins plugin liste](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://wiki.jenkins-ci.org/display/JENKINS/Plugins&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhjBRUyHTWpdpBa2Fn1h5SbmXfRZtQ) est très complète et vous pouvez facilement ajouter votre propre. En plus de l'extensibilité, Jenkins se targue de distribuer des builds et des tests sur plusieurs machines. Il est publié sous licence MIT, de sorte qu'il est libre d'utiliser et de distribuer.

Cloudbees propose également une solution hébergée sous la forme du [Jenkins in the Cloud](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.cloudbees.com/products/jenkins-cloud&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhhtI5knPRDWdNSMpIrlgbwMFVcIBw).

Donc, Jenkins est l’une des meilleures solutions, à la fois puissante et flexible. La courbe d'apprentissage peut être un peu raide, mais si vous avez besoin de flexibilité, cela vaut la peine d'apprendre à l'utiliser.

* **TeamCity**

[](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.jetbrains.com/teamcity/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhiWxIP5-l8YWwmL_rVKJANo252AtA)

TeamCity est le serveur mature d’IC, issu des laboratoires de la compagnie JetBrains. JetBrains a établi son autorité dans le monde du développement de logiciels, et leurs outils tels que WebStorm et ReSharper sont utilisés par les développeurs du monde entier.

TeamCity offre toutes les fonctionnalités de sa version gratuite, mais il est limité aux [100 configurations de build et aux 3 build agents](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://code-maze.com/continuous-integration-with-teamcity/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhiNjIsMn54HJ5bv6wkyZtc9bBQoEw#basicconcepts) . Des agents de construction supplémentaires et des configurations de construction doivent être achetés. Récemment, JetBrains a commencé à offrir un essai en nuage de TeamCity où vous pouvez l'essayer pour un projet sans avoir à l'installer sur place. Il dure 60 jours et exporte le projet par la suite.

Prêt à l'emploi, TeamCity fonctionne sur de nombreuses plateformes et prend en [charge une grande variété d'outils et de frameworks](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://confluence.jetbrains.com/display/TCD9/Supported%2BPlatforms%2Band%2BEnvironments&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhiHaoD-f6V4xNf5TJWUnL38QETSfw) . Il existe de nombreux [plugins](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://plugins.jetbrains.com/teamcity&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhj0bQCUC37oFs3rExhnysd82Q_lhA) disponibles publiquement, développés par JetBrains et des tiers.

En dépit d'être la solution basée sur Java, TeamCity offre le meilleur support .NET parmi les outils de cette liste. Il existe également différents packages d'entreprise, qui évoluent en fonction du nombre d'agents requis.

**Verdict**: Grande solution globale, mais en raison de sa complexité et son prix, mieux adapté aux besoins de l'entreprise.

* **Travis CI**

[https://code-maze.com/wp-content/uploads/2016/02/TravisCI-logo-gray.png](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://travis-ci.org/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhicWBDl4UQogsNbtJDJoUNQCgkRpA)

Travis CI est l'une des solutions hébergées les plus anciennes et elle a gagné la confiance de nombreuses personnes. Bien qu'il soit principalement connu pour la solution hébergée, il offre également la version sur site sous forme de [package d'entreprise](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://enterprise.travis-ci.com/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhipXNC4Pt-vc2SrrwyNS2tbY2hegA) .

Travis CI est gratuit pour tous les projets open source hébergés sur le GitHub et pour les 100 premières builds sinon. Vous pouvez choisir parmi plusieurs [plans de tarification](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://travis-ci.com/plans&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhidpuWFLZZOo9of_ksQzLr_zWRB4w) , la principale différence étant le nombre de builds simultanés que vous pouvez exécuter.

Les constructions sont configurées en utilisant le fichier .travis.yml qui contient les tâches de construction qui seront exécutées lors de l'exécution de la construction. Il prend en charge une variété de langues différentes et une bonne documentation pour les sauvegarder.

On en conclut que c’est une solution mature qui offre à la fois des variantes hébergées et sur site, aimées et utilisées par de nombreuses équipes, très bien documentée.

* **Go CD**

[](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.go.cd/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhhkwNPD8EXpb2ay4eOrUZzzRkd87w)

Go est la [nouvelle incarnation](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=http://build-doctor.com/2010/06/25/cruise-go/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhjU72gnpUNxAg9Xu9SIa4bozyHYlQ) de [Cruise Control](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=http://build-doctor.com/2010/06/25/cruise-go/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhjU72gnpUNxAg9Xu9SIa4bozyHYlQ) de la société ThoughtWorks. En excluant le support commercial offert par ThoughtWorks, Go est gratuit. Il est disponible pour Windows, Mac et diverses distributions Linux.

Ce qui distingue Go de la foule, c'est le concept de pipelines qui facilite la modélisation du workflow. Sur le concept du pipeline, comment il peut aider avec la livraison continue et comment il se compare aux pipelines Jenkins, vous pouvez lire [ici](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://highops.com/insights/continuous-delivery-pipelines-gocd-vs-jenkins&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhg2cwrKKSilezNV4X5uuP33dQrlHA) . Il est conçu à partir du zéro pour supporter les pipelines et éliminer les goulets d'étranglement du processus de construction avec l'exécution parallèle des tâches.

**Bambou**

[logo en bambou](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.atlassian.com/software/bamboo&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhjbNnioKHKCDv2jUaJ-sBhREHiQ1w)

Atlassian est la société qui se concentre sur la fourniture d'outils pour les équipes de développement de logiciels et vous pouvez les connaître grâce à leurs outils tels que [JIRA](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.atlassian.com/software/jira&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhjKuT96XPkuSnK17K9QoaoKZnixoQ) et [Bitbucket](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.atlassian.com/software/bitbucket&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhibBsd_xdrBvESh3wPsM50LcIYs7A) . À l'origine, Bamboo proposait à la fois des solutions cloud et On-premises, mais en mai 2016, la version cloud a été abandonnée en faveur des pipelines Bitbucket (accessibles via le panneau de gauche de votre compte Bitbucket).

Étant alimenté par Docker, Bitbucket Pipelines est une solution très efficace et rapide qui grandit rapidement et devient un digne successeur du Cloud Bamboo.

Le bambou est libre d'essayer pendant 30 jours, et après cela, il y a deux plans pour les petites équipes en croissance. Étant l'outil Atlassian, il a le support natif pour JIRA et BitBucket et vous pouvez même importer facilement vos configurations Jenkins dans le Bamboo.

On constate que Bamboo est un excellent outil CI sur site qui offrait à l'origine une solution Cloud. La solution de cloud a été remplacée par des pipelines Bitbucket, un outil CI moderne et rapide intégré dans Bitbucket. A un essai gratuit pendant 30 jours, et les plans payés après cela.

**Conclusion du second chapitre**

Nous clôturons ce chapitre en faisant un choix des outils que nous allons adopter lors de la mise en place de la solution exemple de notre projet. Nous allons donc utiliser *GIT* pour la gestion de version, *Jenkins* pour serveur d’intégration continue

**Chapitre III. Mise en place du système d’intégration continue**

1. **Gestion de version avec GIT et mise place de l’outil**
   * + 1. **Fonctionnement**

GIT copie l’intégralité du dépôt distant en local. Donc la plus part des commandes exécutent des modifications du dépôt local. Ce qui fait que Git soit très rapide.

Git gère trois états dans lequel les fichiers peuvent résider :

* **Commité**: quand les données sont stockées en sécurité dans votre base de données locale.
* **Modifié** : quand vous avez modifié le fichier mais qu’il n’a pas encore été commité en base.
* **Indexé** : quand vous avez marqué un fichier modifié dans sa version actuelle pour qu’il fasse partie du prochain instantané du projet.

Ceci nous mène aux trois sections principales d’un projet Git local: le répertoire Git, le répertoire de travail et la zone d’index [3].

Voici un schéma récapitulant l’ensemble des opérations local et distant dans un système de gestion de version avec GIT

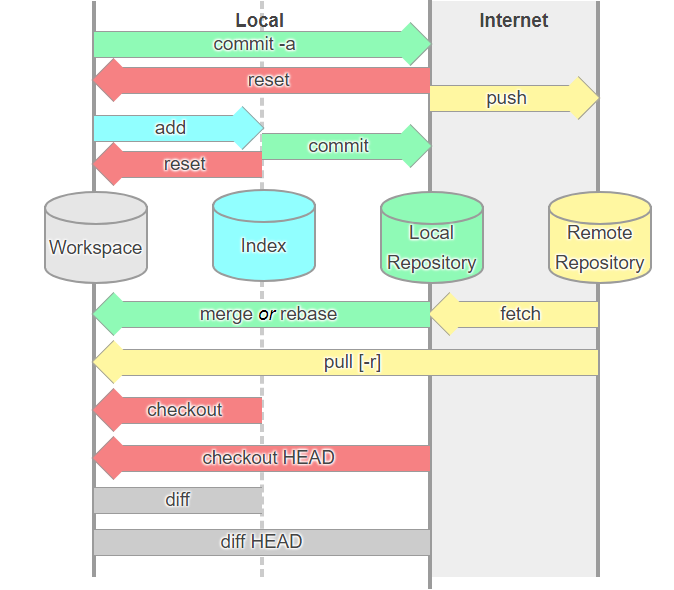


Fig2.1 les opérations dans un système de gestion de version avec GIT[[12]](#footnote-12)

* Local repository (*répertoire Git)* est l’endroit où Git stocke les méta-données et la base de données des objets de votre projet. C’est la partie la plus importante de Git et c’est ce qui est copié lorsque vous clonez un dépôt depuis un autre ordinateur. [3]
* Workspace *(répertoire de travail)* est une extraction unique d’une version du projet. Ces fichiers sont extraits depuis la base de données compressée dans le répertoire Git et placés sur le disque pour pouvoir être utilisés ou modifiés. [3]
* Index *(zone d’index)* est un simple fichier, généralement situé dans le répertoire Git, qui stocke les informations concernant ce qui fera partie du prochain instantané. [3]
  + - 1. **Installation sous Windows et configuration**

Pour installer Git, il nous a suffit de telecharger le fichier d’installation avec l’extension exe sur  <https://git-scm.com/downloads>.

Cela nous permis d‘avoir à la fois la version en ligne de commande et l’interface graphique. Puis nous avons configuré l’outil à la première utilisation.il suffit d’ouvrir git pour les lignes de commandes (git Bash ou git CMD) et d’y entrer les commandes de configurations suivantes :

* Définir l’identité de l’utilisateur

***$ git config --global user.name "Mireille"***

***$ git config --global user.email*** [***iranzimoe@gmail.com***](mailto:iranzimoe@gmail.com)

Il faut noter qu’Il est important de configurer git avec le nom et l’e-mail car tous les commits Git utilisent cette information et elle est indélébile dans tous les commits que vous pourrez manipuler.

* Voir les configurations

***$ git config –list***

***user.name=Mireille***

***user.email=iranzimoe@gmail.com***

* Vérification d’un paramètre

***$ git config user.name***

***Mireille***

* Obtenir de l’aide sur la configuration

***$ git help config***

**Déroulement de la pratique**

Pour commencer on a commencé par l’installation et la configuration de git comme décrit ci-dessus. Nous signalons que, puisque notre sujet de travail été centre su l’intégration continue et moyennant aussi les contraintes temps, nous avons choisi de ne pas développer notre propre application au cours du projet mais d’utiliser un logiciel déjà en cours développement.

1. Nous d’abords avons créé un dépôt sur GitHub avec un compte puis ajouter un utilisateur d’un autre compte en collaborateurs. Le site github.com envoi un e-mail à l’adresse du collaborateur qu’il va d’abord confirmer avant d’être reconnu comme collateur.

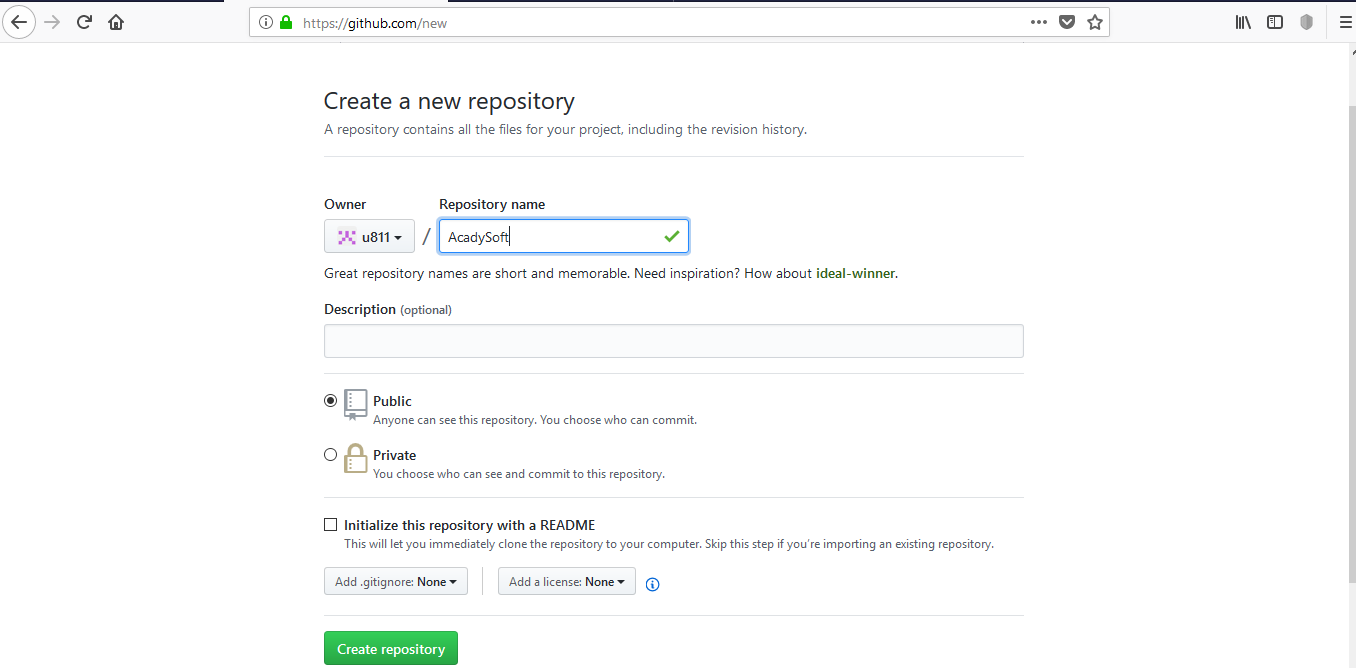


Fig. 2.2 création d’un dépôt sur GitHub

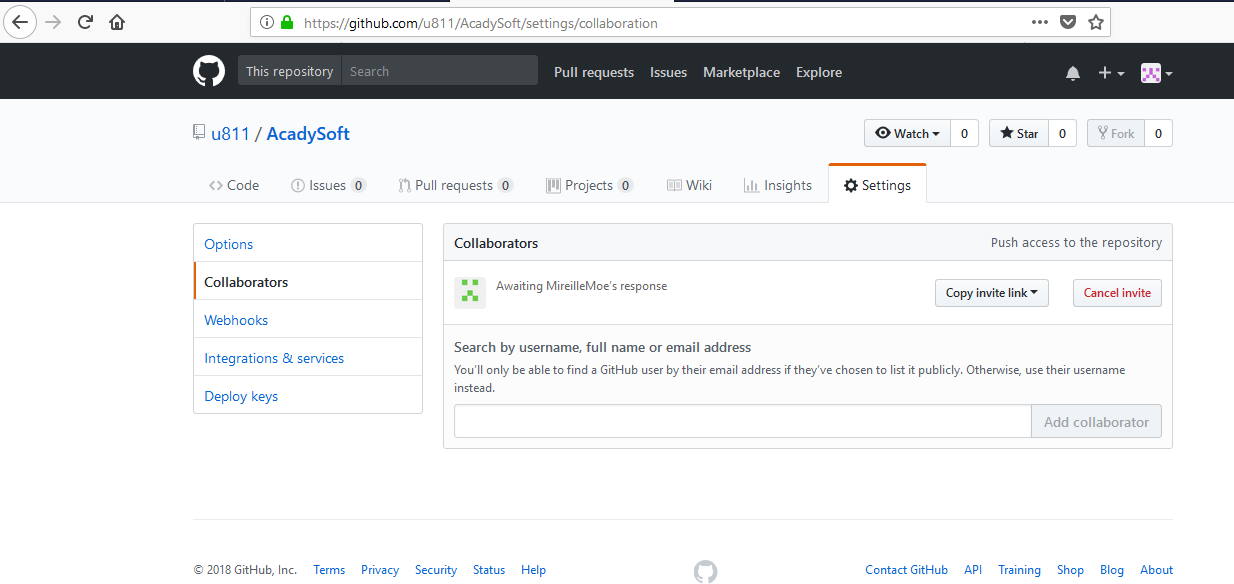


Fig.2.3 Ajout d’un collaborateur

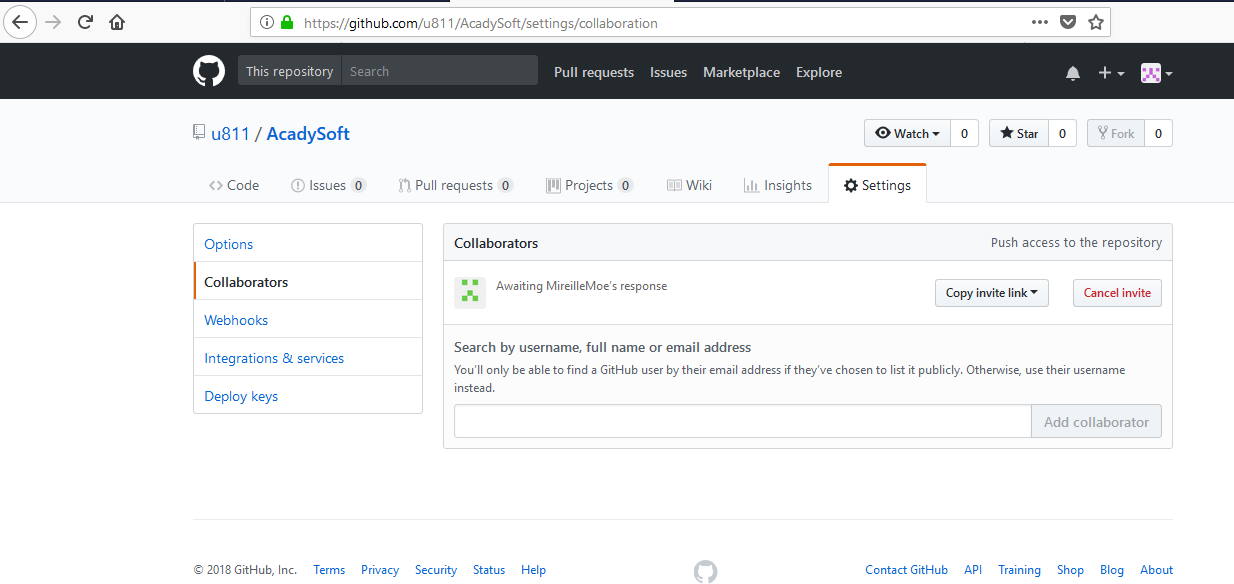


Fig.2.4 Réponse a une invitation de collaboration sur un dépôt GitHub

1. Nous avons ensuite initialisé un dépôt git dans le répertoire contenant l’application sur une deux machines. Puis nous avons envoyé le dépôt contenant l’application sur GitHub pour pouvoir tous y accéder



Fig.2.5 Initiation du dépôt git

1. Sur l’autre machine nous avons fait le clonage du dépôt avec GitHub Desktop. Cette dernière est une application simplifiant les activités du sites Gitub.com.

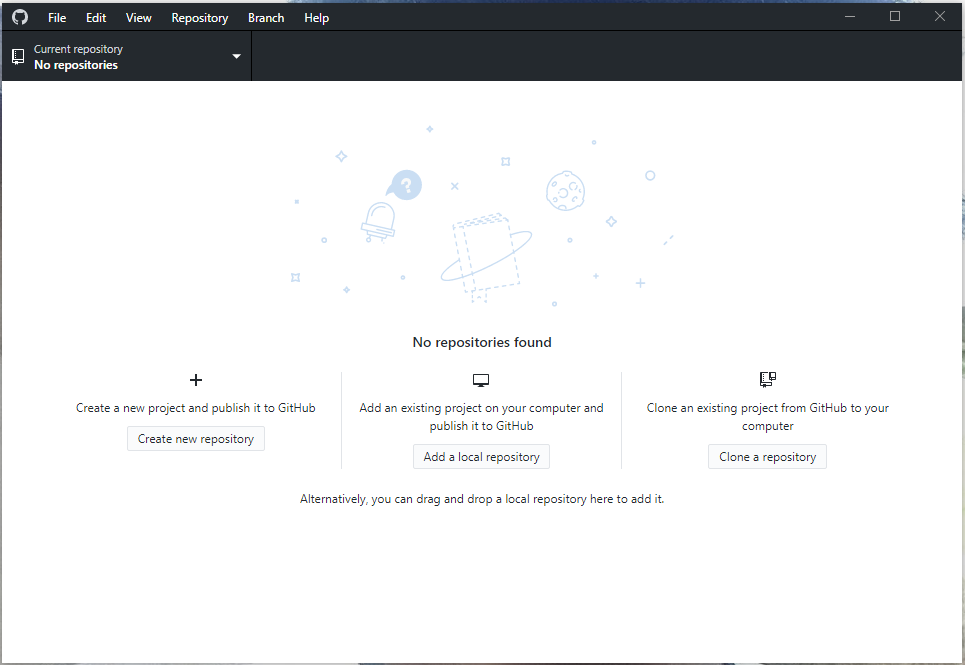


Fig 2.6 Page d’accueil de GitHub Desktop

En choisissant l’option Clone a repository, on obtient la page suivante.

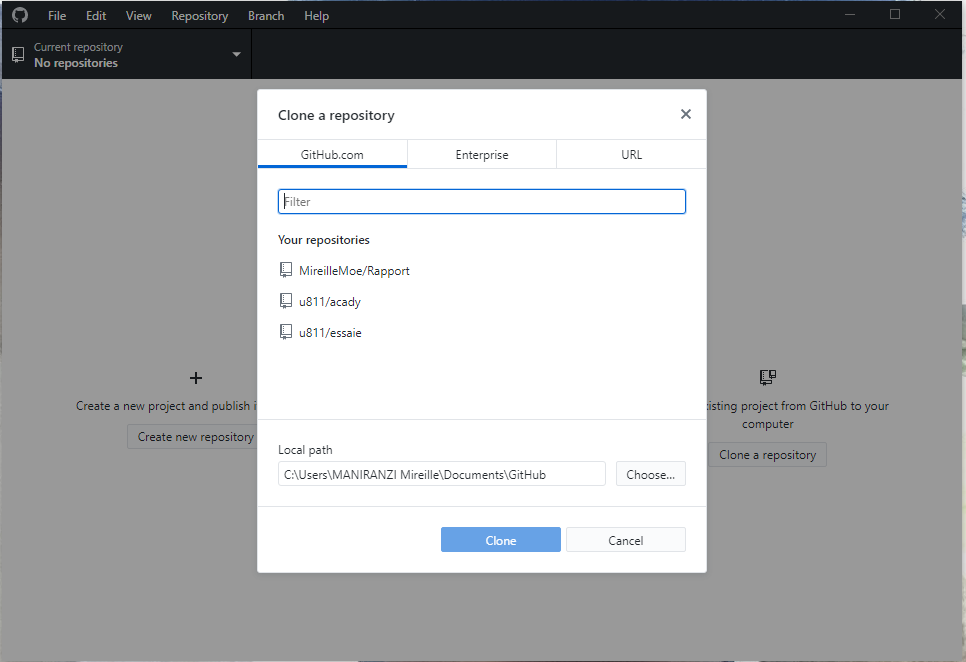


Fig.2.7 Choix du dépôt a cloner

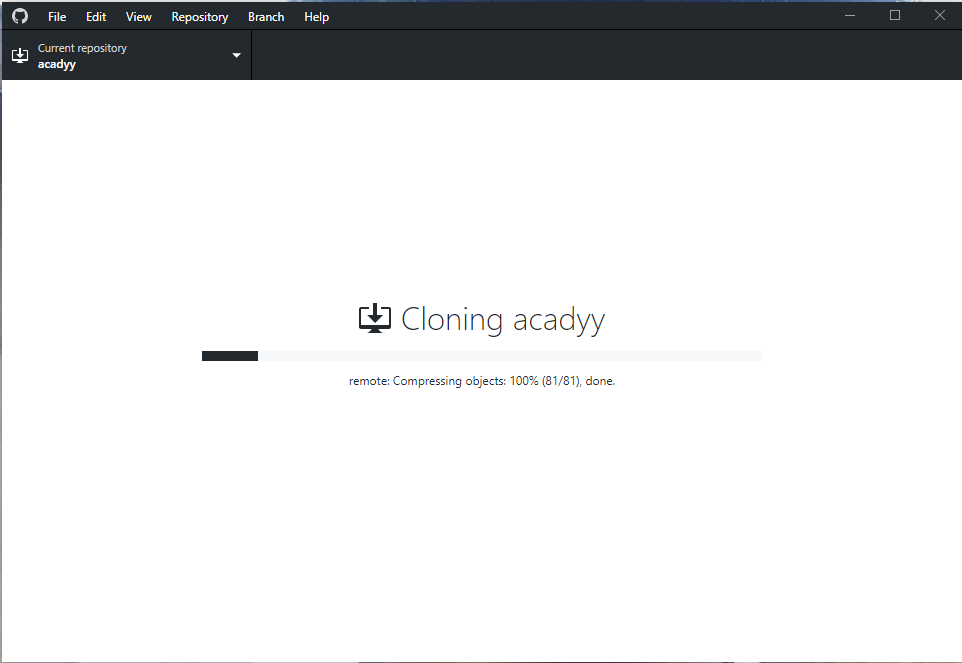


Fig. 2.8 Processus de clonage encours

**REFERENCES BIBLIOGRAPHIQUES**

1. Anne, C. (DI5 2013 - 2014). “ Mise en place d’un Serveur d’Intégration Continue et Qualité de Code, disponible sur : [anne.castier@etu.univ-tours.fr](mailto:anne.castier@etu.univ-tours.fr)”
2. Martin, F. (2006). “ Continuous Integration, disponible sur : https://www.martinfowler.com/articles/continuousIntegration.html [visité dernièrement : 26/03/2018] ”, 2006.
3. Scott, C. (2011-07-13). “ Pro Git, disponible sur :http://tinyurl.com/amazonprogit”
4. Gauthier, P. “ Tests unitaires Développement dirigé par les tests Utilisation de JUnit : École Nationale Supérieure des Mines de Saint-Étienne ”
5. Mise en place d’une plateforme d’intégration continue au sein de la direction générale des impôts
6. [Vladimir, P](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://code-maze.com/author/codemaze_blog/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhiksWzCoNG09gmuO4RUT9Yqz4m6Vw) . (20 février 2016).  “Top 8 des outils d'intégration continue”
7. John , F, S.(2010). “continuous-integration-with-hudson”

1. Nous n’avons pas traduit le terme build car nous n’avons pas trouvé de traduction pertinente et parce que

   Ce terme reste majoritairement utilisé dans les métiers des TI [↑](#footnote-ref-1)
2. http://www.commentcamarche.com/contents/473-cycle-de-vie-d-un-logiciel [↑](#footnote-ref-2)
3. http://igm.univ-mlv.fr/~dr/XPOSE2012/Integration%20Continue/concept.html [↑](#footnote-ref-3)
4. https://fr.wikipedia.org/wiki/Int%C3%A9gration\_continue [↑](#footnote-ref-4)
5. [https://www.pulsar-informatique.com/creation-site-internet/comment-creer-un-site-internet/comment-gerer- son-projet-de-site-web/outils-de-gestion-de-projet-web/environnements-de-dev/controle-de-version](https://www.pulsar-informatique.com/creation-site-internet/comment-creer-un-site-internet/comment-gerer-%20%20%20%20son-projet-de-site-web/outils-de-gestion-de-projet-web/environnements-de-dev/controle-de-version) [↑](#footnote-ref-5)
6. Mise en place d’une plateforme d’intégration continue au sein de la direction générale des impôts.pdf [↑](#footnote-ref-6)
7. <http://www.aubryconseil.com/post/La-gestion-des-bugs-dans-le-projet-IceScrum> [↑](#footnote-ref-7)
8. <https://www.codebuilder.fr/blog/developpement-collaboratif-logiciels-gestion-versions/> [↑](#footnote-ref-8)
9. <https://www.codebuilder.fr/blog/developpement-collaboratif-logiciels-gestion-versions/> [↑](#footnote-ref-9)
10. <http://linsolas.developpez.com/articles/java/outils/builds/> [↑](#footnote-ref-10)
11. <http://linsolas.developpez.com/articles/java/outils/builds/> [↑](#footnote-ref-11)
12. https://blog.lesieur.name/comprendre-et-utiliser-git-avec-vos-projets/ [↑](#footnote-ref-12)