|  |
| --- |
|  |



|  |
| --- |
| **Mise en place d’un système d’Intégration Continue** |
| Faculté des Technologies de l’information et de la communication Département de Technologies de l’information |
|  |
| RAPPORT DE STAGE |
|  |
| **Microsoft** |
| **[Choisir la date]** |
|  |

**Dédicaces**

Nous dédions ce rapport á nos familles, nos amis et nos frères et sœurs en Christ

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 et motivation qui ont permis de mener notre projet à bout malgré de nombreux obstacles et découragements !

Nous tenons également à remercier MSC NIRAGIRA Ferdinand pour avoir encadré notre projet de fin d’études. Nous sommes reconnaissants envers lui pour nous avoir soutenus et aidés à achever ce projet.

**Liste de figures**

[Fig. I‑1 Comparaison de l’intégration classique et l’intégration continue - 17 -](#_Toc514400503)

[Fig. II‑1 Détail d'un système d'IC - 21 -](#_Toc514400504)

[Fig. II‑2 Système de gestion de version centralisé - 24 -](#_Toc514400505)

[Fig. II‑3 Système de gestion de version distribuée [3] - 25 -](#_Toc514400506)

[Fig. II‑4 processus d’exécution de builds avec Maven - 33 -](#_Toc514400507)

[Fig. III‑1 les opérations dans un système de gestion de version avec GIT - 41 -](#_Toc514400508)

[Fig. III‑2 Début de l'installation de git - 42 -](file:///C:\Users\BEDE\Desktop\rapport%20de%20stage%20correction.docx#_Toc514400509)

[Fig. III‑3 Assistant d'installation de git - 42 -](file:///C:\Users\BEDE\Desktop\rapport%20de%20stage%20correction.docx#_Toc514400510)

[Fig. III‑4 Configuration de l'identité en git - 42 -](#_Toc514400511)

[Fig. III‑5 Liste des configurations - 43 -](#_Toc514400512)

[Fig. III‑6 début de la création d'un compte GitHub - 44 -](#_Toc514400513)

[Fig. III‑7 Configuration du compte GitHub - 44 -](#_Toc514400514)

[Fig. III‑8 Premier pas avec GitHub - 44 -](#_Toc514400515)

[Fig. III‑9 Vérification de l'adresse e-mail - 45 -](#_Toc514400516)

[Fig. III‑10 e-mail de vérification de GitHub - 45 -](#_Toc514400517)

[Fig. III‑11 Cre’ation d'un depot GitHub - 46 -](#_Toc514400518)

[Fig. III‑12 Dépôt créé(1) - 46 -](#_Toc514400519)

[Fig. III‑13 Dépôt créé(2) - 47 -](#_Toc514400520)

[Fig. III‑14 envoi d'une invitation pour une collaboration sur un projet - 47 -](#_Toc514400521)

[Fig. III‑15 Attente d'une réponse sur une invitation de collaboration sur un projet github - 48 -](#_Toc514400522)

[Fig. III‑16 email contenant l'invitation github de collaboration sur un projet - 48 -](#_Toc514400523)

[Fig. III‑17 Invitation de collaboration sur un projet github - 49 -](#_Toc514400524)

[Fig. III‑18 collaboration réussie - 49 -](#_Toc514400525)

[Fig. III‑19 page d'accueil git en interface graphique - 50 -](#_Toc514400526)

[Fig. III‑20 initialisation d'un dépôt git(1) - 50 -](#_Toc514400527)

[Fig. III‑21 indexation des fichiers - 50 -](#_Toc514400528)

[Fig. III‑22 Fichiers tous indexés - 50 -](#_Toc514400529)

[Fig. III‑23 le premier commit - 51 -](#_Toc514400530)

[Fig. III‑24 Commit effectué - 51 -](#_Toc514400531)

[Fig. III‑25 Début d'un push - 51 -](#_Toc514400532)

[Fig. III‑26 Authentification pour effectuer un push(1) - 52 -](#_Toc514400533)

[Fig. III‑27 Authentification pour effectuer un push(2) - 52 -](#_Toc514400534)

[Fig. III‑28 Push encours - 53 -](#_Toc514400535)

[Fig. III‑29 Push réussi - 53 -](#_Toc514400536)

[Fig. III‑30 installation de github desktop - 54 -](#_Toc514400537)

[Fig. III‑31 Démarrage avec github desktop - 54 -](#_Toc514400538)

[Fig. III‑32 Authentification sur github desktop - 55 -](#_Toc514400539)

[Fig. III‑33 Page d'accueil github desktop - 55 -](#_Toc514400540)

[Fig. III‑34 Choix du dépôt à cloner - 56 -](#_Toc514400541)

[Fig. III‑35 Clonage en cours - 56 -](#_Toc514400542)

[Fig. III‑36 Aperçue du dépôt cloné - 57 -](#_Toc514400543)

[Fig. III‑37 Dépôt cloné mais pas encore modifié - 57 -](#_Toc514400544)

[Fig. III‑38 aperçue du dépôt modifié - 58 -](#_Toc514400545)

[Fig. III‑39 push - 58 -](#_Toc514400546)

[Fig. III‑40 git pull - 58 -](#_Toc514400547)

[Fig. III‑41 Assistant d'installation de jdk - 59 -](#_Toc514400548)

[Fig. III‑42 Choix de l'emplacement de jdk - 59 -](#_Toc514400549)

[Fig. III‑43 Choix des modules de xampp - 60 -](#_Toc514400550)

[Fig. III‑44 Démarrage Tomcat dans xampp - 61 -](#_Toc514400551)

[Fig. III‑45 Page d'accueil apache tomcat - 61 -](#_Toc514400552)

[Fig. III‑46 Ajout d'une variable d'environnement pour Maven(1) - 62 -](#_Toc514400553)

[Fig. III‑47 Ajout d'une variable d'environnement pour Maven(2) - 63 -](#_Toc514400554)

[Fig. III‑48 Déploiement de jenkins - 63 -](#_Toc514400555)

[Fig. III‑49 Réussite de déploiement de jenkins dans xampp - 64 -](#_Toc514400556)

[Fig. III‑50 Déblocage de jenkins - 64 -](#_Toc514400557)

[Fig. III‑51 Page pour débloquer Jenkins - 65 -](#_Toc514400558)

[Fig. III‑52 Page d'administration de Jenkins - 66 -](#_Toc514400559)

[Fig. III‑53 Configuration Jenkins de jdk - 66 -](#_Toc514400560)

[Fig. III‑54 configuration Jenkins de git - 67 -](#_Toc514400561)

[Fig. III‑55 Configuration jenkins de maven - 68 -](#_Toc514400562)

[Fig. III‑56 Prêt pour la création d'un nouveau job - 68 -](#_Toc514400563)

[Fig. III‑57 Début de la création d'un nouveau job - 68 -](#_Toc514400564)

[Fig. III‑58 Configuration d'un nouveau job(1) - 69 -](#_Toc514400565)

[Fig. III‑59 Configuration d'un nouveau job(2) - 70 -](#_Toc514400566)

[Fig. III‑60 Configuration d'un nouveau job(3) - 71 -](#_Toc514400567)

[Fig. III‑61Build encours - 72 -](#_Toc514400568)

**LISTE DES ABREVIATIONS ET SIGLES**

**IC**: Intégration Continue ou **CI:** Continuos 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)

Table des matières

[I. Chapitre I INTRODUCTION GENERALE - 10 -](#_Toc514400569)

[I.1 JUSTIFICATION DU CHOIX DU SUJET - 12 -](#_Toc514400570)

[I.1.1 Motivation et intérêt personnel du sujet - 12 -](#_Toc514400571)

[I.1.2 Pertinence scientifique du sujet : intérêt académique - 12 -](#_Toc514400572)

[I.1.3 Pertinence sociale du sujet :Intérêt des bénéficiaires - 13 -](#_Toc514400573)

[I.2 PROBLEMATIQUE DE LA RECHERCHE - 13 -](#_Toc514400574)

[I.3 HYPOTHESES DE LA RECHERCHE - 14 -](#_Toc514400575)

[I.4 OBJECTIFS DE LA RECHERCHE - 14 -](#_Toc514400576)

[I.4.1 Objectif global - 14 -](#_Toc514400577)

[I.4.2 Objectifs spécifiques - 14 -](#_Toc514400578)

[I.5 DELIMITATION DU SUJET - 14 -](#_Toc514400579)

[I.6 CADRE CONCEPTUEL ET THEORIQUE - 15 -](#_Toc514400580)

[I.6.1 Quelques définitions - 15 -](#_Toc514400581)

[I.6.2 Approche théorique - 16 -](#_Toc514400582)

[I.6.2.2 Déroulement du processus d’IC - 17 -](#_Toc514400583)

[I.7 DEMARCHE SUIVIE POUR LA CONDUITE DU PROJET - 18 -](#_Toc514400584)

[I.8 CONCLUSION DU PREMIER CHAPITRE - 19 -](#_Toc514400585)

[II. CHAPITRE II: SPECIFICATIONS FONCTIONNELLES ET CONCEPTUELLES DE LA PLATEFORME IC - 21 -](#_Toc514400586)

[II.1 Détails du processus d’intégration continue [5] - 21 -](#_Toc514400587)

[II.2 Fonctionnalités nécessaires pour l’intégration continue - 22 -](#_Toc514400588)

[II.2.1 La gestion et contrôle de versions - 22 -](#_Toc514400589)

[II.2.2 L’exécution des builds - 23 -](#_Toc514400590)

[II.2.3 Le suivi et qualification des performances - 23 -](#_Toc514400591)

[II.2.4 La gestion des bugs - 24 -](#_Toc514400592)

[II.3 Etude des composants de la plateforme pour la solution - 24 -](#_Toc514400593)

[II.3.1 Outils de contrôle et gestion des versions - 24 -](#_Toc514400594)

[II.3.2 Outils de tests automatisés et d’exécution des builds - 28 -](#_Toc514400595)

[II.3.3 Serveurs d’IC - 34 -](#_Toc514400596)

[II.4 Conclusion du second chapitre - 38 -](#_Toc514400597)

[III. Chapitre III. Mise en place du système d’intégration continue - 40 -](#_Toc514400598)

[III.1 Introduction du troisième chapitre - 40 -](#_Toc514400599)

[III.2 La gestion de version avec GIT - 40 -](#_Toc514400600)

[III.2.1 Résumé du fonctionnement de GIT - 40 -](#_Toc514400601)

[III.2.2 Installation sous Windows et configuration de Git - 42 -](#_Toc514400602)

[III.3 Serveur d’Intégration continue JENKINS - 59 -](#_Toc514400603)

[III.3.1 Installation et configuration des outils - 59 -](#_Toc514400604)

**MOTS CLES**

Build[[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,gitk

# Chapitre I 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**, livraison du logiciel au client.
* **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.

## 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.

### 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.

### 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.

### 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 tâches 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.

## 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.

## 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.

## OBJECTIFS DE LA RECHERCHE

### 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.

### 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

## 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.

## CADRE CONCEPTUEL ET THEORIQUE

### 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
* **Repository**: 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** (pull): 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.
* La livraison continue (« Continuous Delivery”) est une discipline de développement logiciel dans laquelle vous construisez un logiciel de telle manière qu’il puisse être déployé dans l’environnement de production à tout moment.[[3]](#footnote-3)
* **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 :**
* **Une tâche de build** est une manière de compiler, tester, empaqueter, déployer ou d'effectuer des actions sur votre projet [7]
* **gitk** : une interface graphique qui aide à visualiser les logs.
* **Le déploiement** est le terme utilisé pour le processus d'installation d'une application Web (une application tierce WAR ou votre propre application Web personnalisée) sur le serveur Tomcat[[4]](#footnote-4)
* A quelques exceptions près, le but principal d’une tâche de build est généralement de construire quelque chose. Dans Jenkins, nous appelons cette chose un **artefact**. Un artefact pourrait être un exécutable binaire (un fichier JAR ou WAR pour un projet Java, par exemple), ou certains autres livrables liés, comme de la documentation ou du code source[7]

### 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. I‑1 Comparaison de l’intégration classique et l’intégration continue

#### **Déroulement du processus d’IC**[[5]](#footnote-5)

**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 [[6]](#footnote-6)

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.

## DEMARCHE SUIVIE POUR LA CONDUITE DU PROJET

Le projet a été découpée en 3 grandes phases savoir :

1. Une première étape de prise en main pendant laquelle nous avons fait l’état des lieux. Nous avons donc fait des recherches au sujet de notre projet. Nous avons lu plusieurs ouvrages pour avoir une idée sur le thème qui était nouveau pour nous. Cette première étape a donc mené à une phase de documentations, de recherches principalement sur la gestion des versions et l’intégration continue. Et donc tout ce qui permettait de partir sur des directives claires pour mettre en place un système d’IC.
2. Une seconde étape a été la documentation sur les outils nécessaires. En effet, une fois

Les bases posées, nous avons fait une recherche sur les outils utilisés en IC pour pouvoir faire un choix relativement.

1. La troisième étape a été la mise en place des outils choisis pour notre système d’IC et leur configuration. Pendant cette phase nous avons pu, premièrement, faire de test en guise d’exemple sur la gestion des versions, à part. Puis nous avons combiné la gestion de version avec l’intégration continue.
2. La quatrième étape nous a permis de reprendre la pratique dès le début afin de montrer dace document à l’aide des screenshots la plus part de la pratique effectuée au cours du projet
3. Enfin la dernière étape, l’une des plus importantes, a été menée en parallèle aux deux premières étapes et à la quatrième étape : C’est l’étape de documentation.la production de ce rapport était nécessaire, puisqu’il fallait que quelqu’un voulant se renseigner sur le sujet, ou mettre en place le système d’IC ou un étudiant voulant approfondir le sujet puissent trouver un document a ce propos.

## 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

## Détails du processus d’intégration continue [5]

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

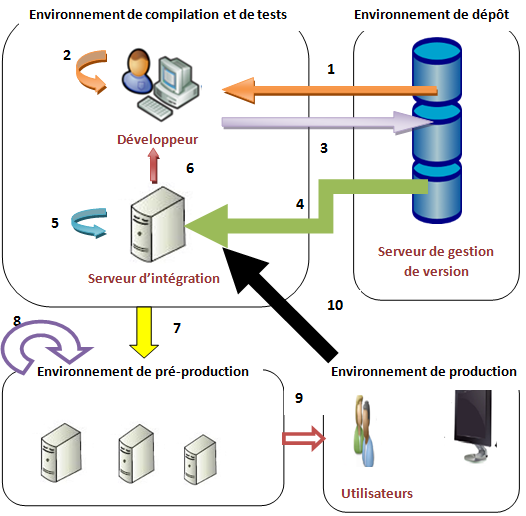


Fig. II‑1 Détail d'un système d'IC

* + - 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.

## 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.[[7]](#footnote-7)

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).[[8]](#footnote-8)

### La gestion des bugs[[9]](#footnote-9)

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.

## Etude des composants de la plateforme pour la solution

### Outils de contrôle et gestion des versions

#### Quelques détails sur la gestion et le control de version

##### **Système de gestion de version centralisé**

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

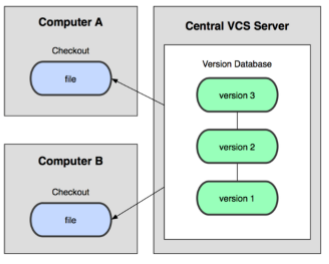


Fig. II‑2 Système de gestion de version centralisé

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 les dépôts central et les membres doivent mettre à jour leur référentiel en dupliquant le dépôt sur serveur central.

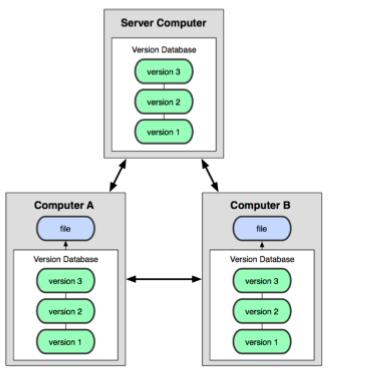


Fig. II‑3 Système de gestion de version distribuée [3]

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))

#### Description de quelques-uns des outils de gestion de version[[10]](#footnote-10)

##### **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[[11]](#footnote-11)

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.**

### Outils de tests automatisés et d’exécution des builds

#### 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.

#### 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 :

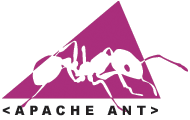
* Les TestCase sont des classes Java héritant de la classe junit.framework. Ils contiennent un certain nombre de méthodes de tests et servent généralement à tester le bon fonctionnement d'une classe.
* Une TestSuite permet d'exécuter un certain nombre de TestCase déjà définis.

#### Outils d’exécution des builds[[12]](#footnote-12)

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.

#### 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**
* **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[[13]](#footnote-13)**

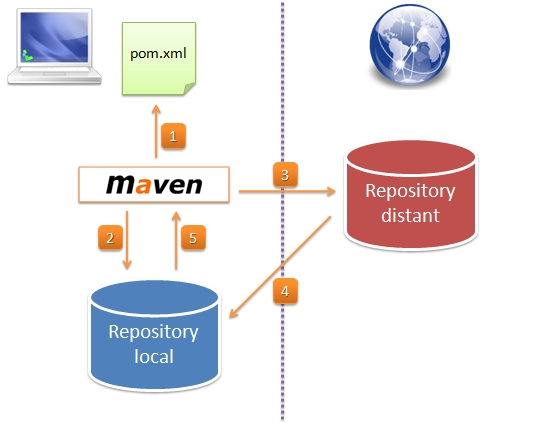
[](http://linsolas.developpez.com/articles/java/outils/builds/images/maven-dependencies.jpg)

Fig. II‑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.

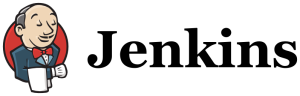
* **Avantages**
* Utilisation des conventions **Maven**, permettant de raccourcir le pom.xml.
* La gestion des dépendances, en particulier les dépendances transitives.
* L'héritage de projets.
* De nombreux plugins sont aujourd'hui disponibles.
* Possibilité de réaliser de nombreuses tâches avec un pom.xml 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).

### 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)

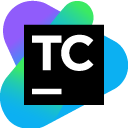
Jenkins est un outil CI open-source écrit en Java. Il est originaire de la [fourche de Hudson](https://translate.googleusercontent.com/translate_c?depth=1&hl=fr&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://jenkins-ci.org/blog/2011/01/11/hudsons-future/&xid=17259,15700021,15700105,15700124,15700149,15700168,15700173,15700201&usg=ALkJrhib5J5KGH8Jjzh0alBOpyZYW1iHbQ) 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 savons maintenant que la fonction principale de tout outil d'Intégration Continue est de surveiller du code source dans un outil de gestion de configuration, d'en extraire la dernière version et de la construire dès que des modifications sont apportées, de faire des tests et de déployer l’application. Donc, il Nous faudra :

* **Git** pour la gestion de versions
* **Jenkins** comme serveur d’intégration continue.
* **Maven** pour effectuer des builds
* **Tomcat** pour déployer des applications java (nous avons utilisé celui qui est de **xampp**)
* Un navigateur web : **Google** **chrome**
* **Netbeans** pour coder en java
* **GithubDesktop** qui facilite les opérations entre git et github.com
* **JDK :** un environnement de développement java

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

## Introduction du troisième chapitre

Le système d’IC est constitué en grande partie par deux grands outils à savoir un gestionnaire source et un serveur d’intégration continue. Pour notre projet il s’agit respectivement de git et de Jenkins. Nous allons montrer dans ce chapitre des screenshots de quelques-uns des écrans d’installation, de configuration, d’utilisation des outils et des écrans montrant des résultats.

Au cours de la pratique, nous avons d’abord travaillé sur la gestion des versions pour nous familiariser avec le concept qui était nouveau pour nous. C’est par après que nous avons combiné le versionnig avec l’IC

Nous avons utilisé seulement deux machines sur lesquelles etaient installer git.l’une de deux contenait jenkins en plus.

## La gestion de version avec GIT

### Résumé du fonctionnement de GIT

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 donc trois états dans lequel les fichiers peuvent résider [3]:

* **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 ainsi que la section du projet distants.

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

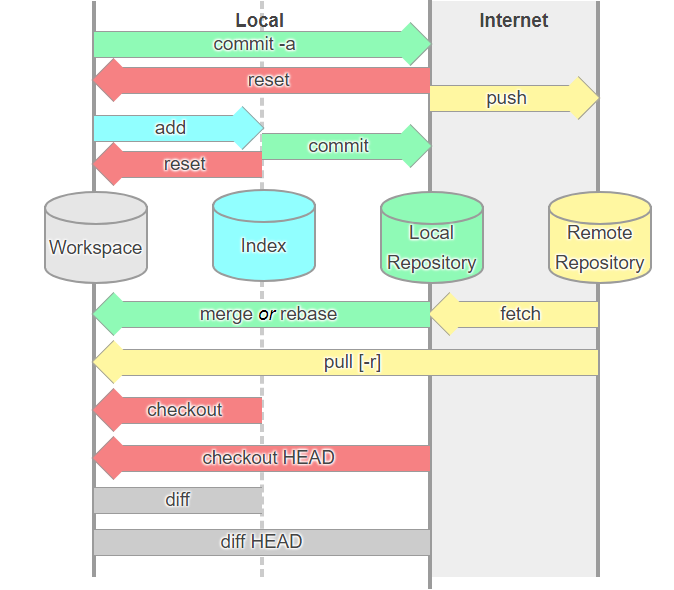


Fig. III‑1 les opérations dans un système de gestion de version avec GIT

* 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]
* Dépôt distant (Remote repository) :c’est le dépôt serveur du projet. C’est celui-ci qu’on clone pour avoir une copie en local

### Installation sous Windows et configuration de Git

Pour installer Git, il nous a suffi de télécharger le fichier d’installation sur  <https://git-scm.com/downloads> puis de l’exécuter une fois le téléchargement terminé.

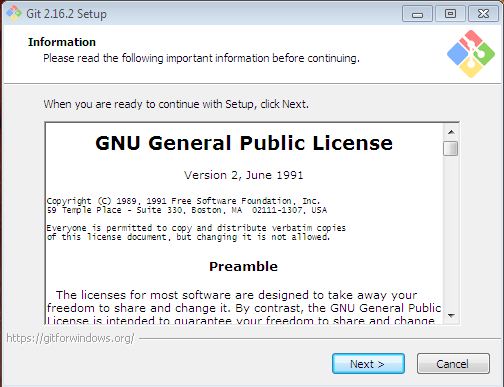


Fig. III‑2 Début de l'installation de git

Fig. III‑3 Assistant d'installation de git

La réussite de l’installation nous a permis d‘avoir à la fois la version en ligne de commande et l’interface graphique de git. Puis nous avons passé a la configuration de l’outil pour la première utilisation. Après avoir ouvert git en lignes de commandes git Bash nous avons entré quelques commandes de:

* configurations  de l’identité de l’utilisateur

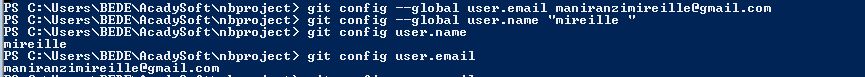


Fig. III‑4 Configuration de l'identité en git

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 la liste des configurations

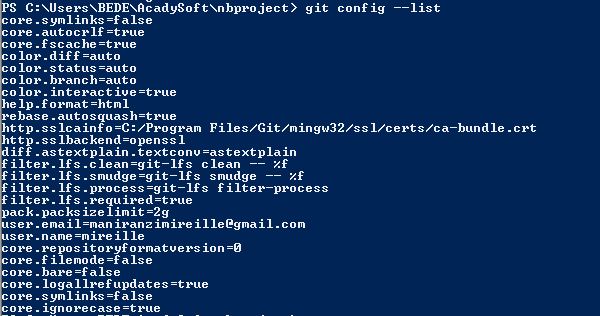


Fig. III‑5 Liste des configurations

* Obtenir de l’aide sur la configuration

***$ git help config***: cette commande ouvre une fenêtre du navigateur web contenant une description complète sur la commande précisé (***config*** pour ces cas)

* Pour obtenir juste de l’aide à propos de git

***$ git help***

#### Cas Pratique sur la gestion de Version avec git

Après l’installation et la configuration de git décrit ci-dessus nous avons passé à la gestion des versions proprement dit.

Nous signalons que, puisque notre sujet de travail été centre su l’intégration continue, 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 d’abord des comptes github.com. ce dernier est un serveur hébergeant des dépôts contenant projets.la création d’un compte est gratuit.

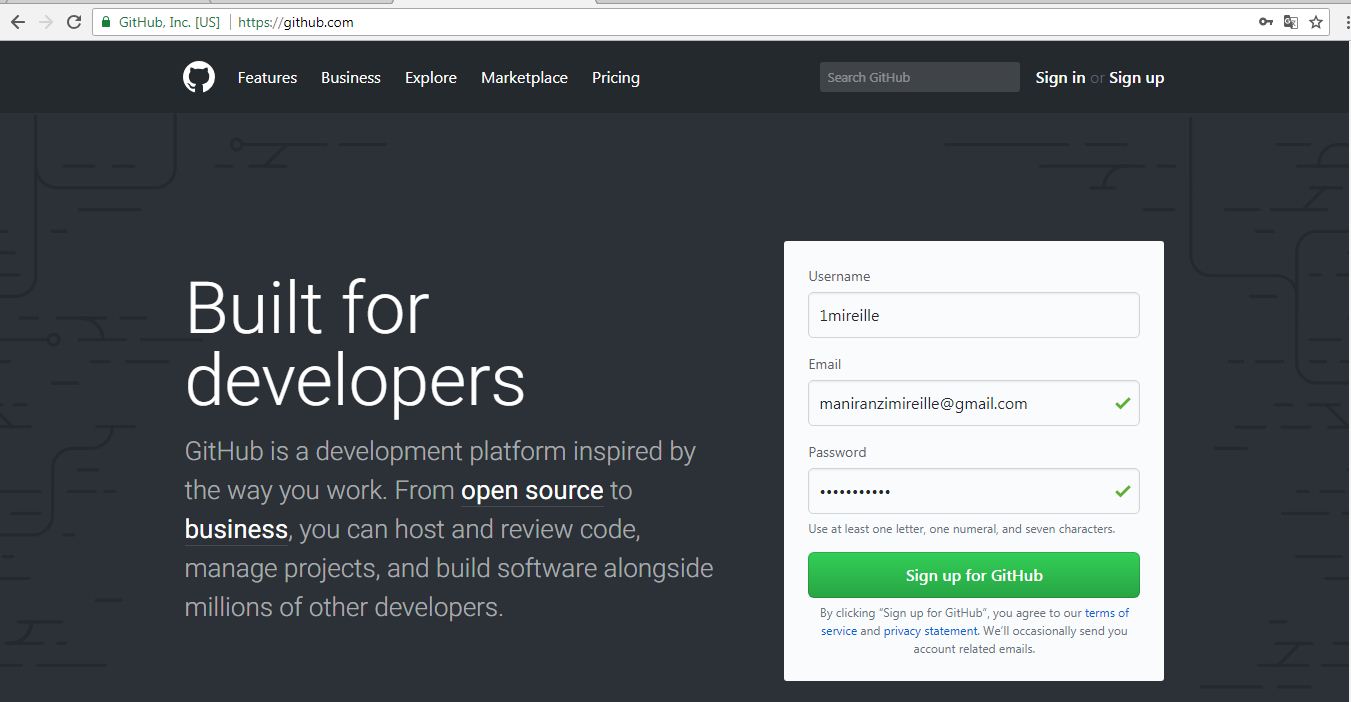


Fig. III‑6 début de la création d'un compte GitHub

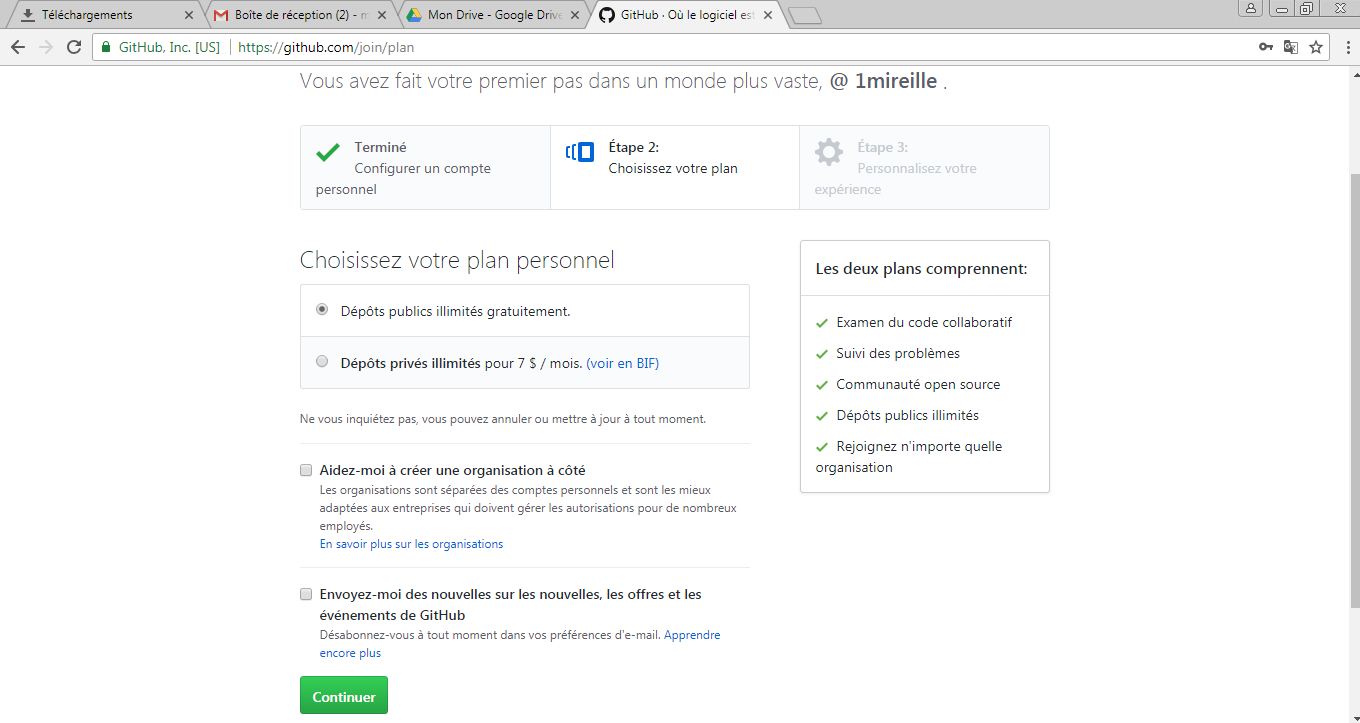


Fig. III‑7 Configuration du compte GitHub



Fig. III‑8 Premier pas avec GitHub

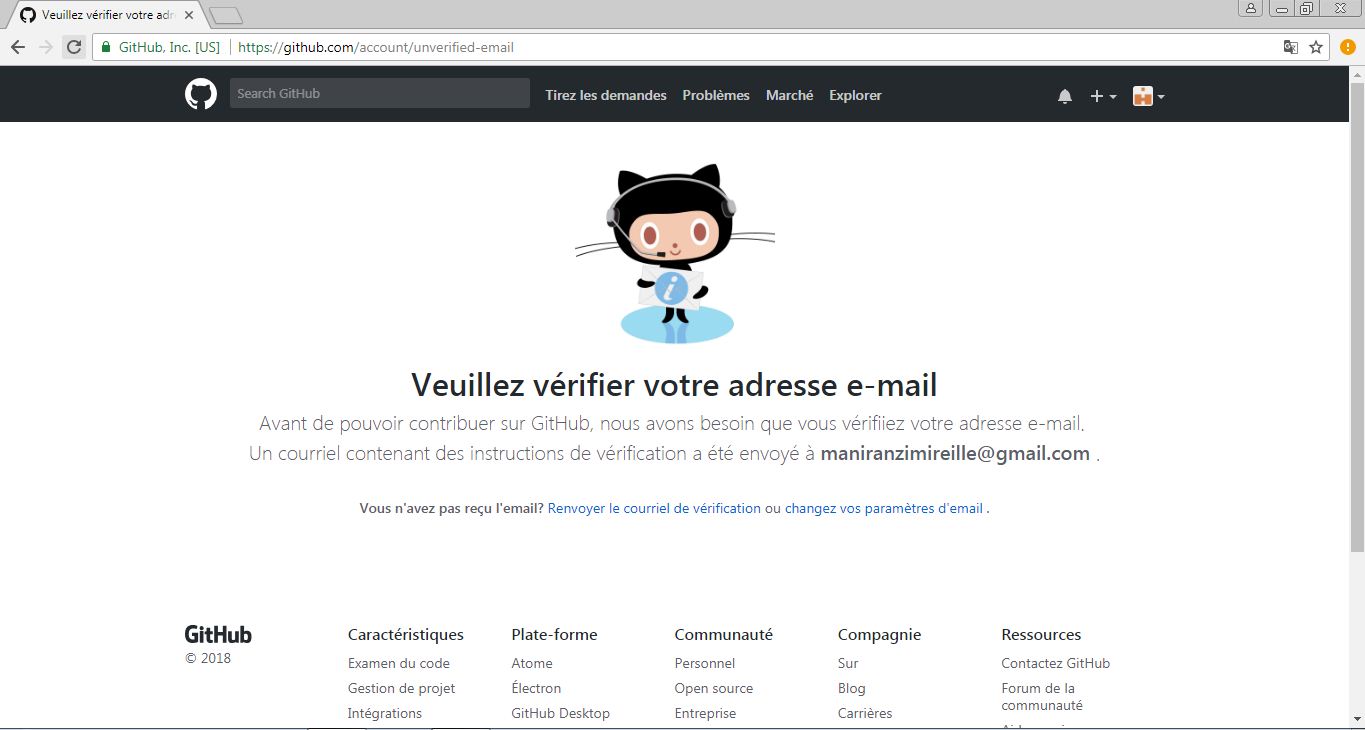


Fig. III‑9 Vérification de l'adresse e-mail

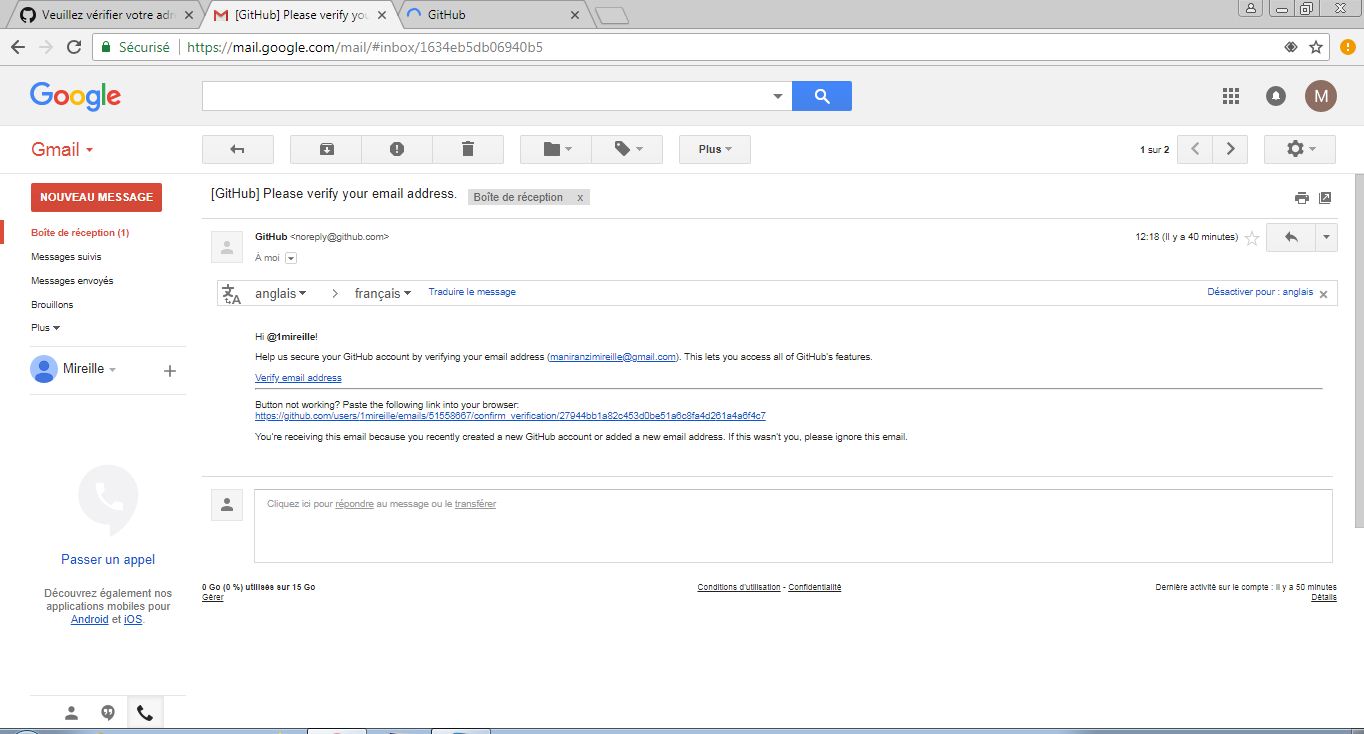


Fig. III‑10 e-mail de vérification de GitHub

1. Nous avons ensuite créé un dépôt sur GitHub

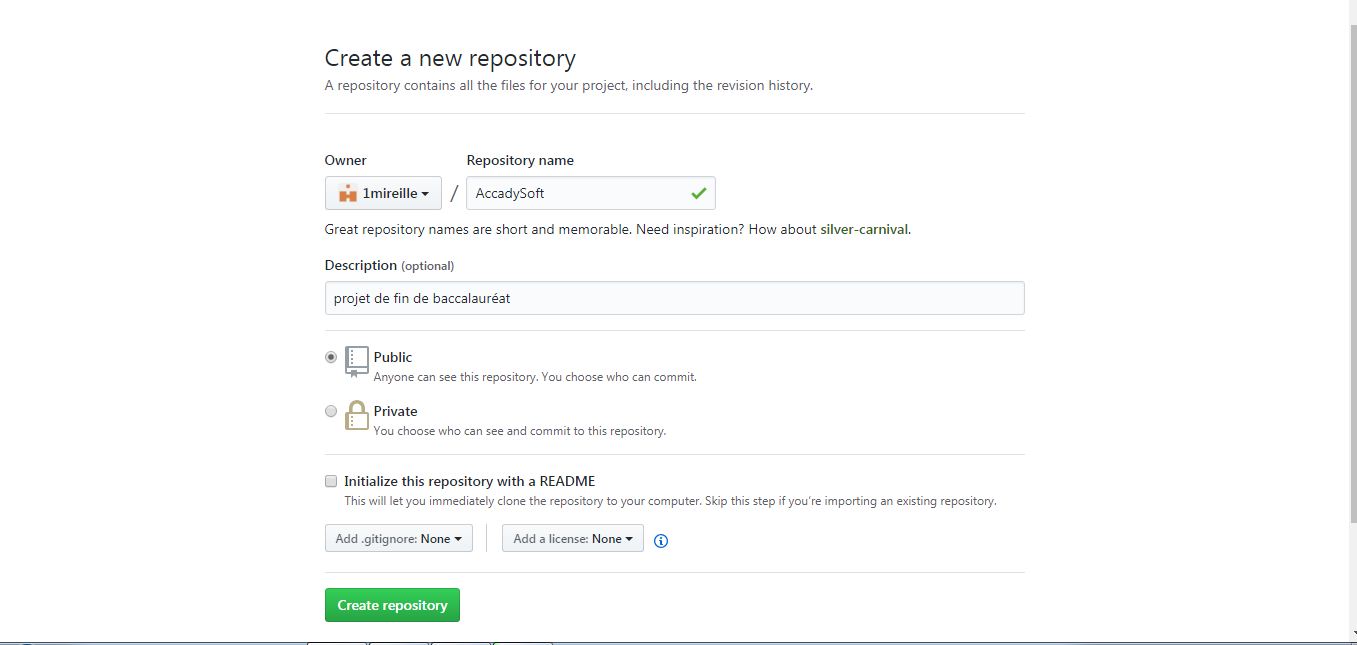


Fig. III‑11 Cre’ation d'un depot GitHub

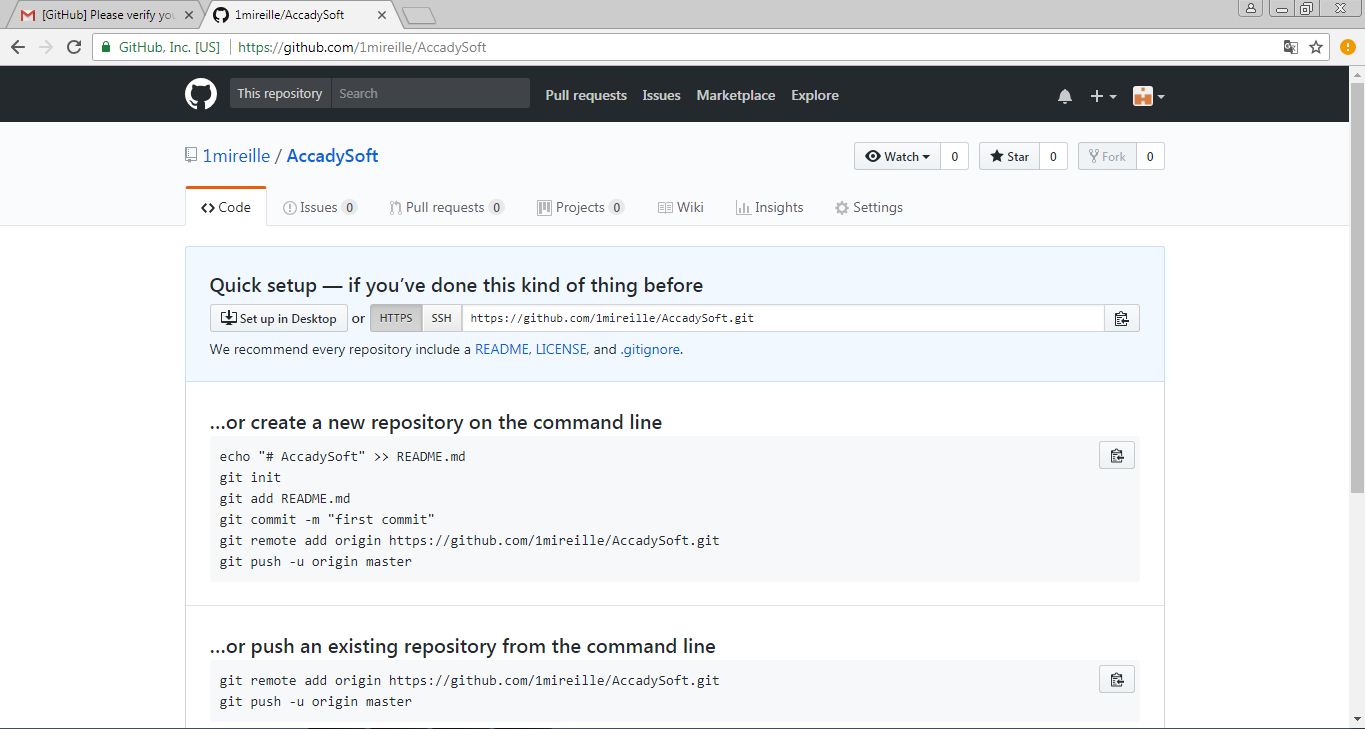


Fig. III‑12 Dépôt créé(1)



Fig. III‑13 Dépôt créé(2)

1. Nous avons ensuite ajoute un collaborateur au projet. Le processus d’ajout d’un collaborateur au projet GitHub est simple. Nous possédions maintenant deux comptes sur GitHub. La demande d’ajout d’un collaborateur s’effectue à partir du compte du propriétaire du dépôt. En cliquant sur le bouton *create repository* puis sur settings c’est possible d’ajouter un collaborateur au projet. Il suffit d’entrer son nom d’utilisateur dans la zone de texte puis de valider. Une fois l’invitation émise, site github.com envoi un e-mail à son adresse mail contenant un lien d’une invitation qu’il faut d’abord confirmer avant d’être reconnu comme collaborateur.

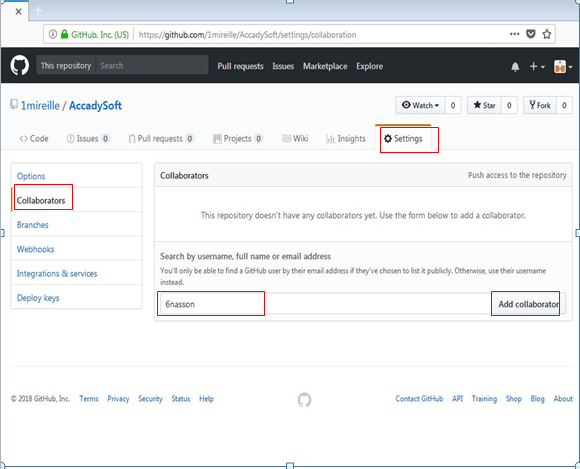


Fig. III‑14 envoi d'une invitation pour une collaboration sur un projet

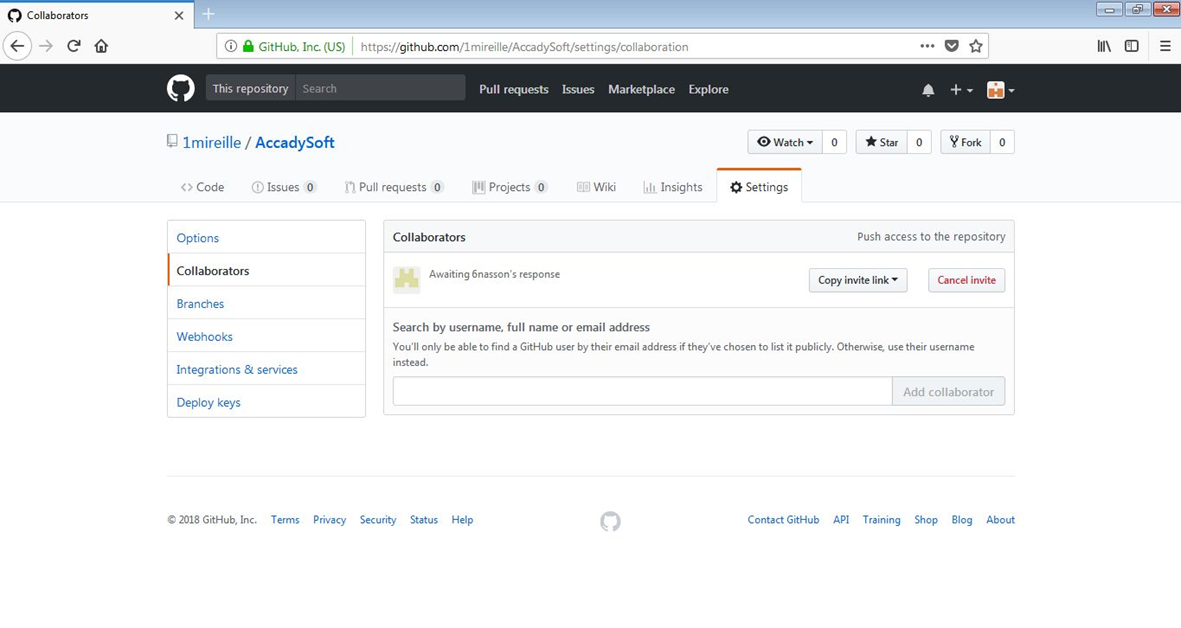


Fig. III‑15 Attente d'une réponse sur une invitation de collaboration sur un projet github

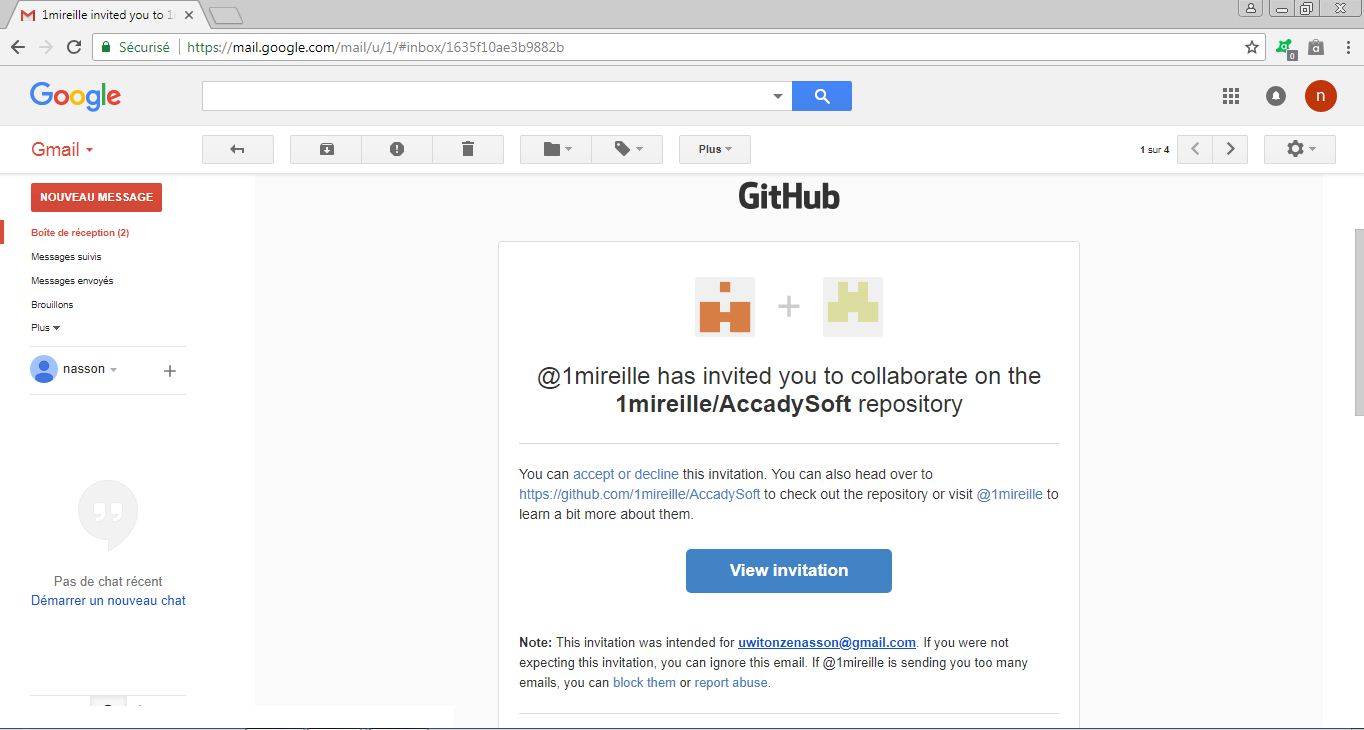


Fig. III‑16 email contenant l'invitation github de collaboration sur un projet

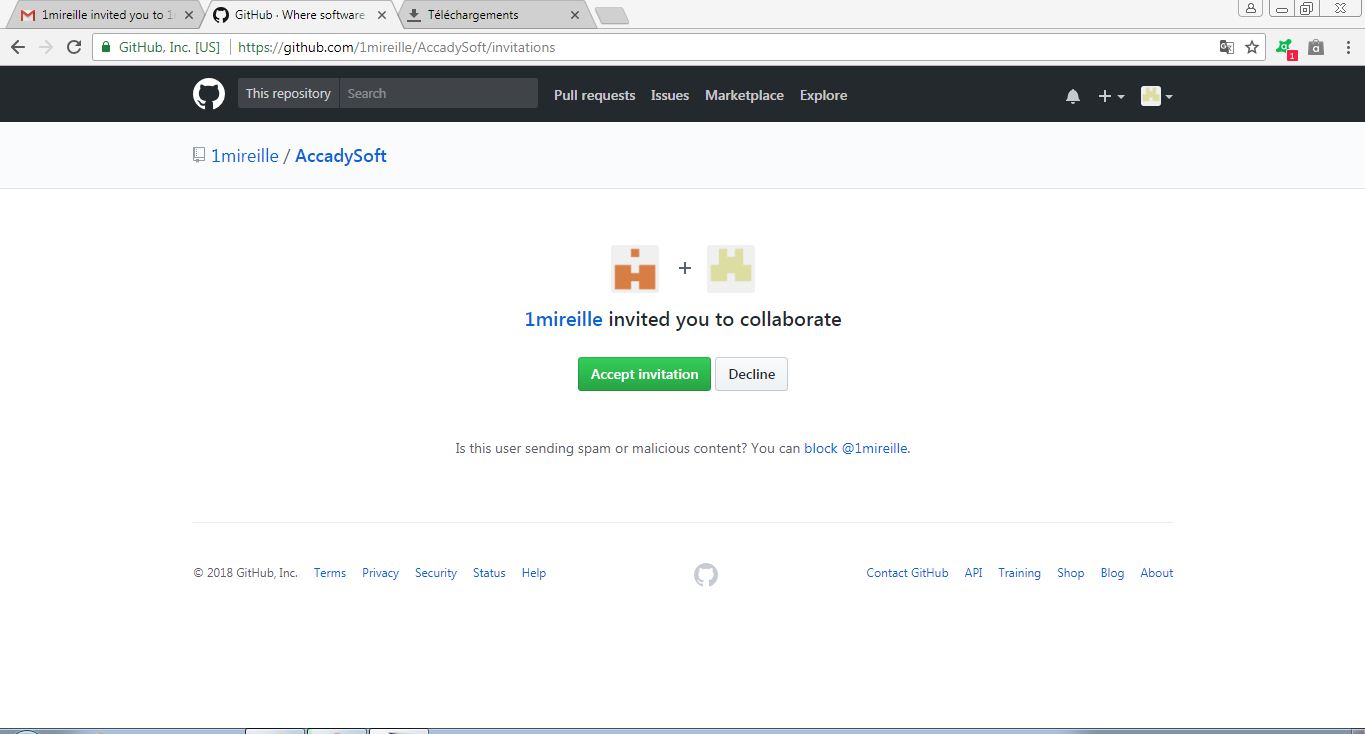


Fig. III‑17 Invitation de collaboration sur un projet github

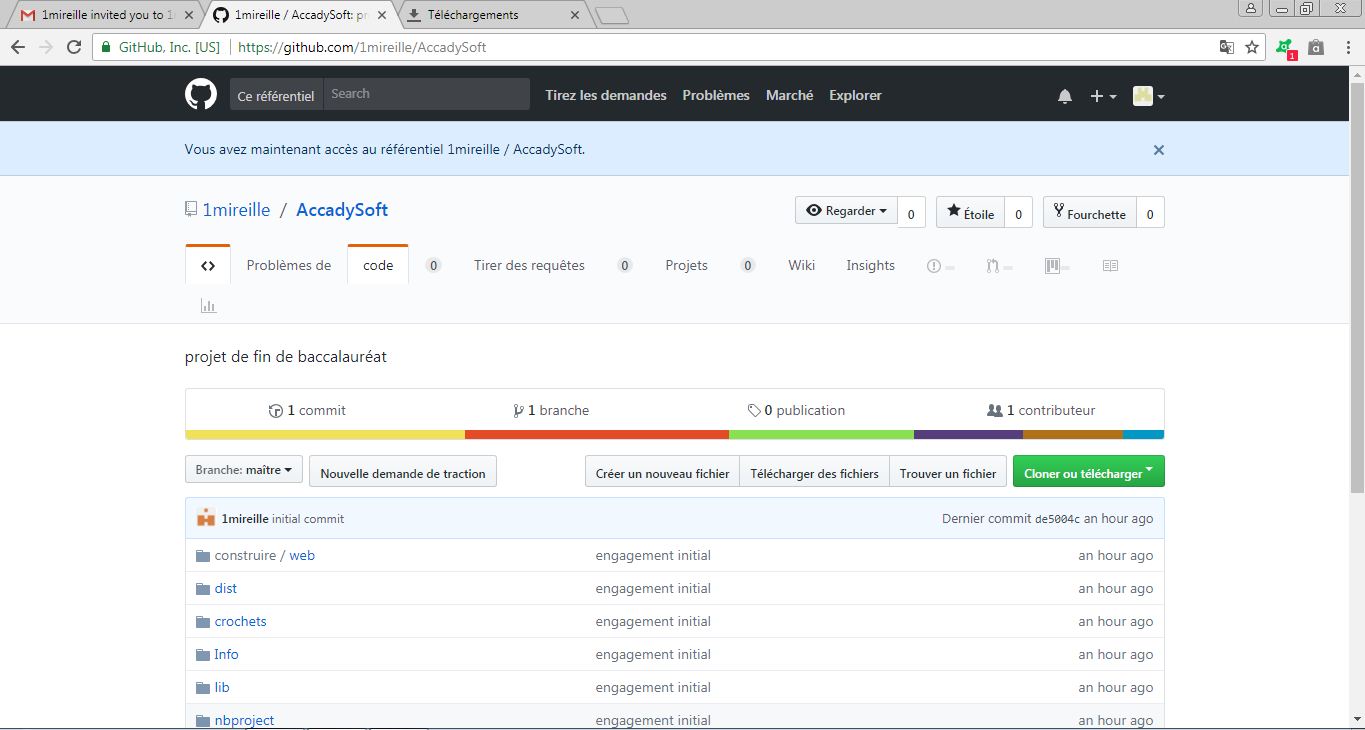


Fig. III‑18 collaboration réussie

1. L’étape suivant a été d’initialisé un dépôt git dans le répertoire contenant l’application.

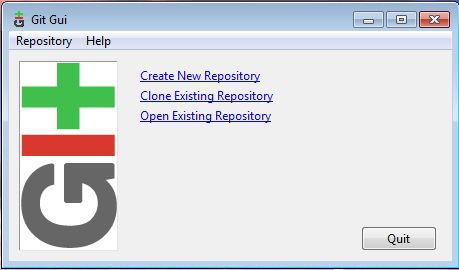


Fig. III‑19 page d'accueil git en interface graphique

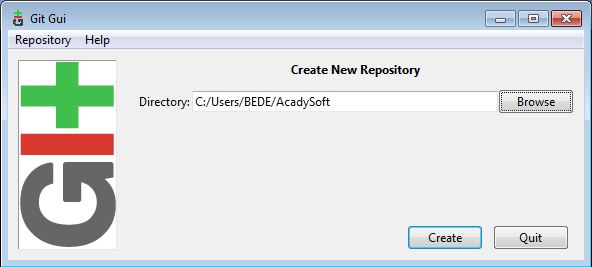


Fig. III‑20 initialisation d'un dépôt git(1)

1. Pour pouvoir effectuer le premier commit, on doit d’abord indexé les fichiers modifiés en cliquant sur stage *changed* puis sur *oui*

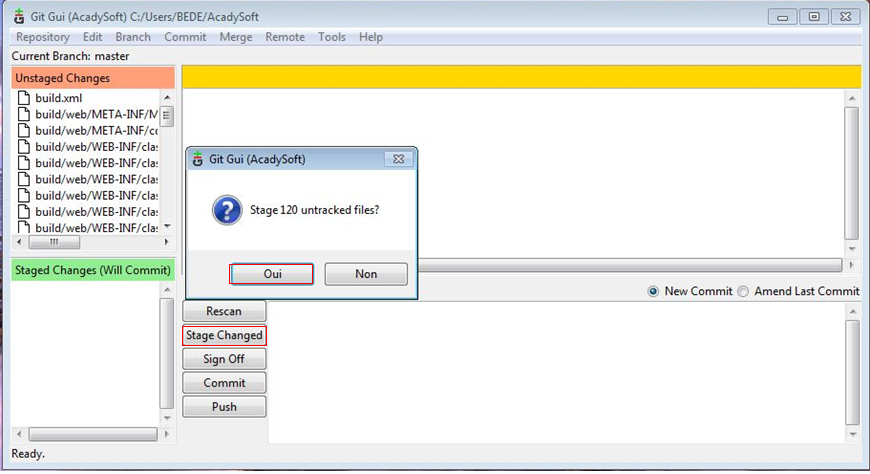


Fig. III‑21 indexation des fichiers

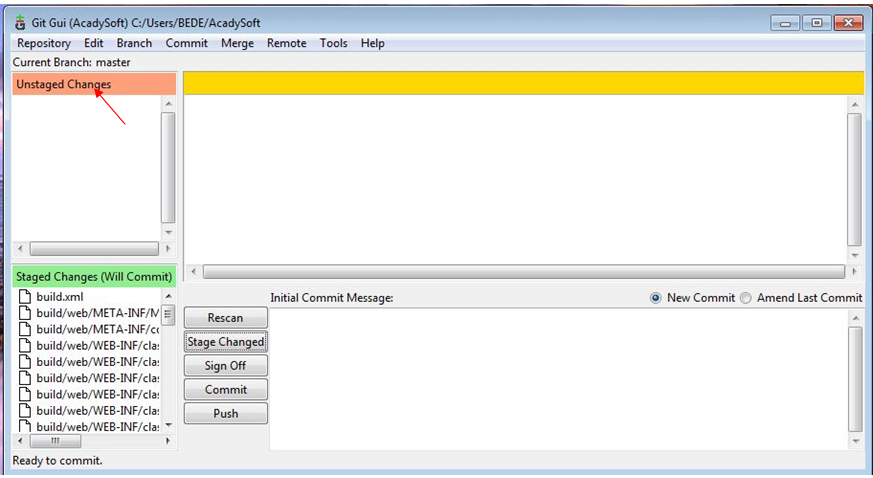


Fig. III‑22 Fichiers tous indexés

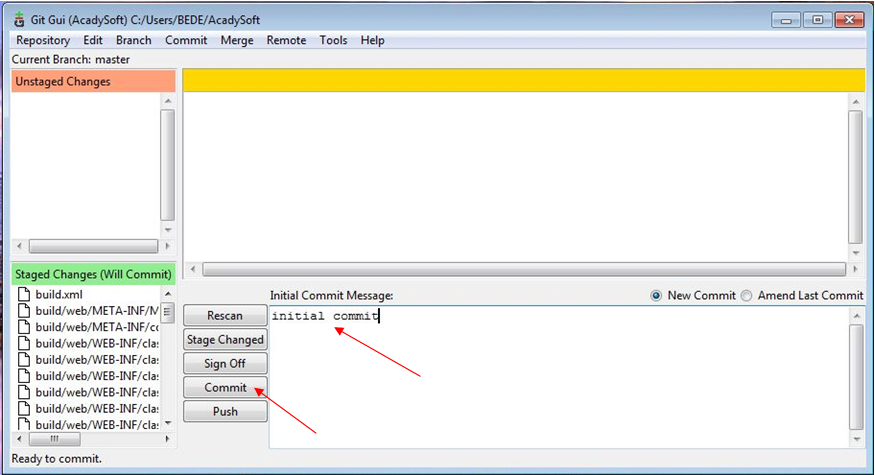


Fig. III‑23 le premier commit

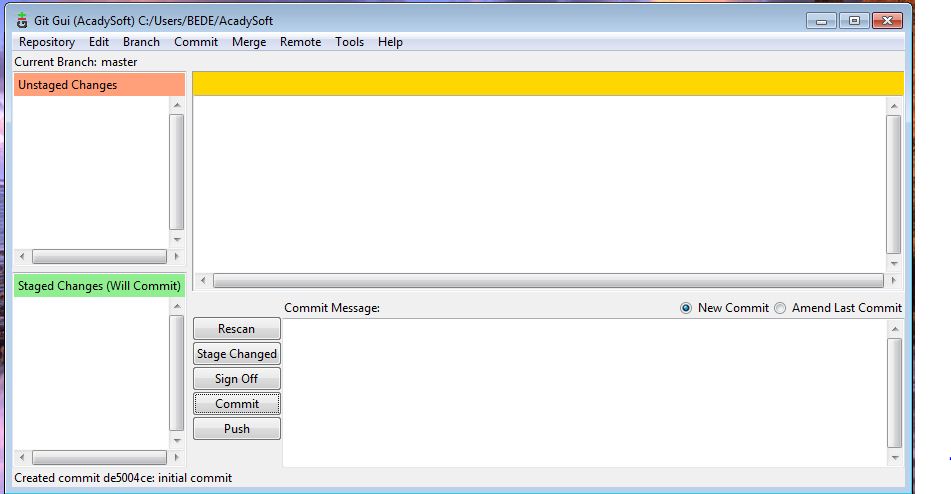


Fig. III‑24 Commit effectué

1. Envoi du dépôt contenant l’application sur GitHub pour pouvoir tous y accéder.il a fallut cliquer sur le bouton *push*

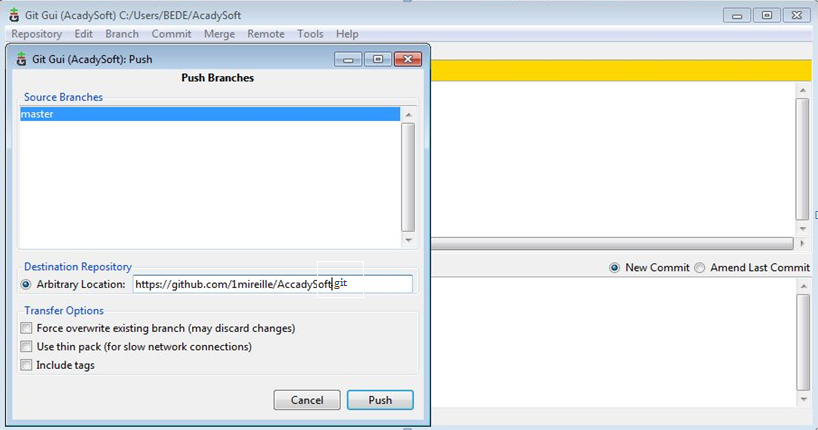


Fig. III‑25 Début d'un push

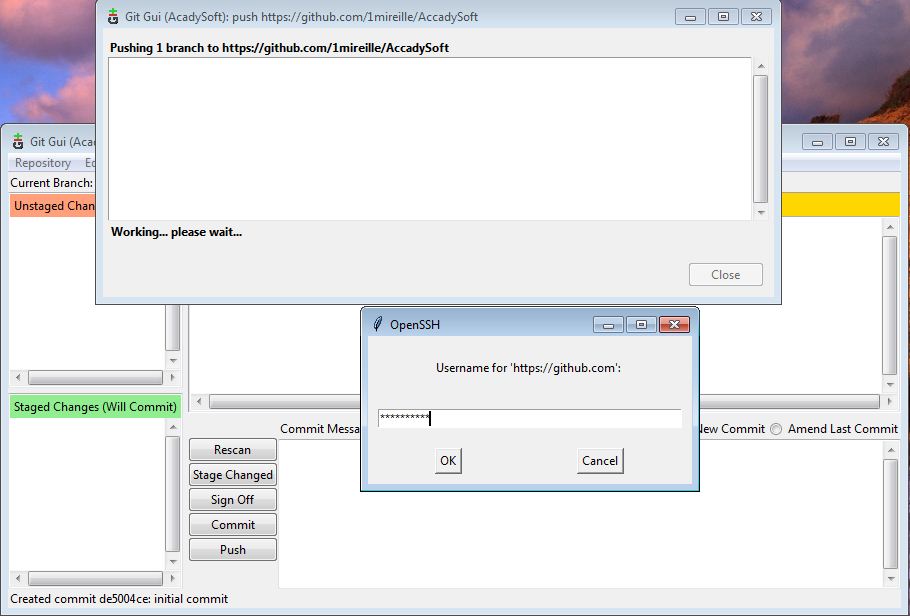


Fig. III‑26 Authentification pour effectuer un push(1)

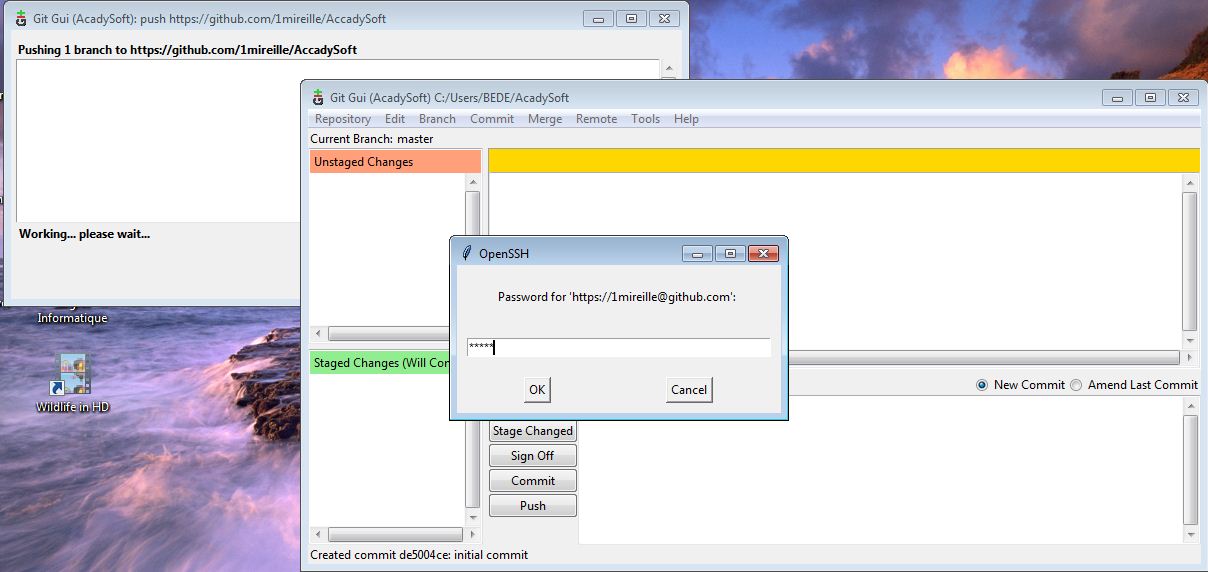


Fig. III‑27 Authentification pour effectuer un push(2)

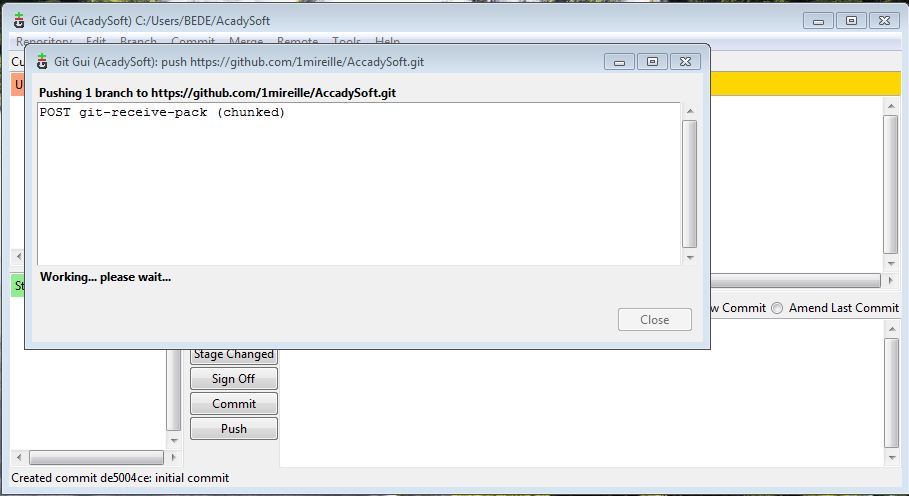


Fig. III‑28 Push encours

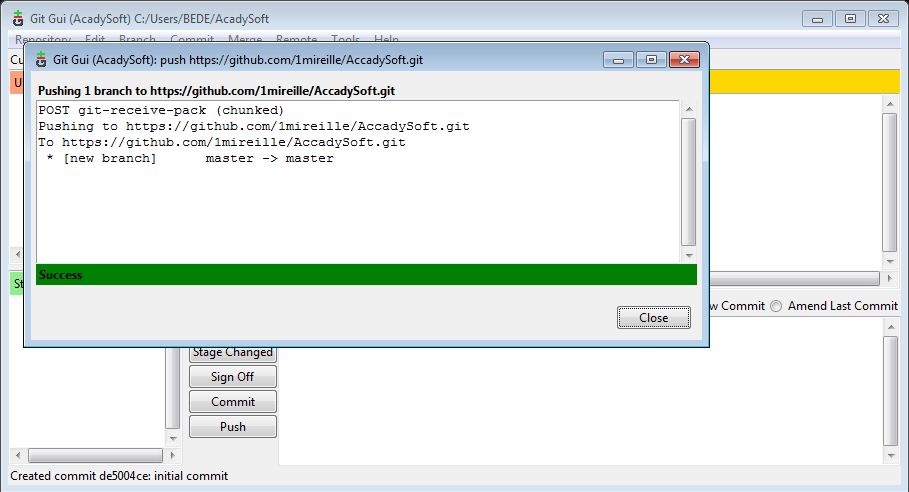


Fig. III‑29 Push réussi

1. Sur l’autre machine nous avons fait le clonage du dépôt avec GitHub Desktop. Cette dernière est une application simplifiant l’interactivité entre les dépôts locaux et le serveur Gitub. Il est en quelque sorte intermédiaire entre github.com et git.



Fig. III‑30 installation de github desktop

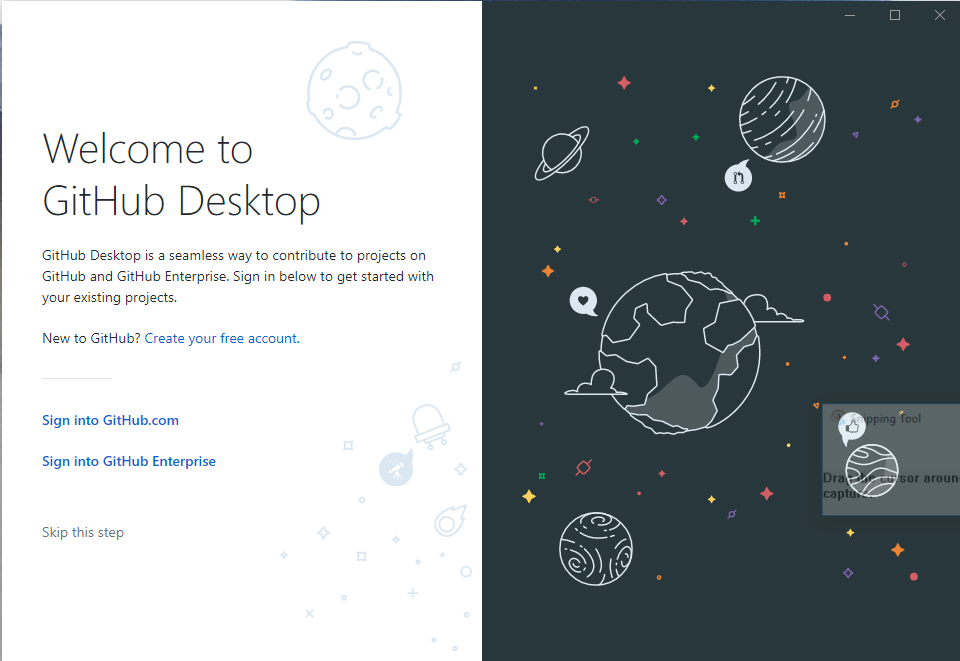


Fig. III‑31 Démarrage avec github desktop

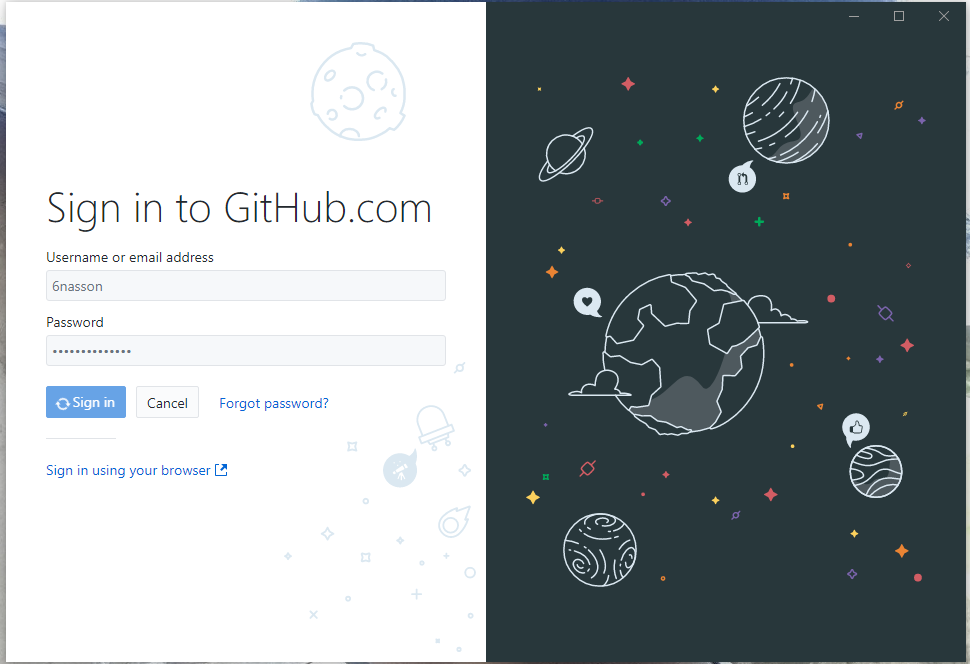


Fig. III‑32 Authentification sur github desktop

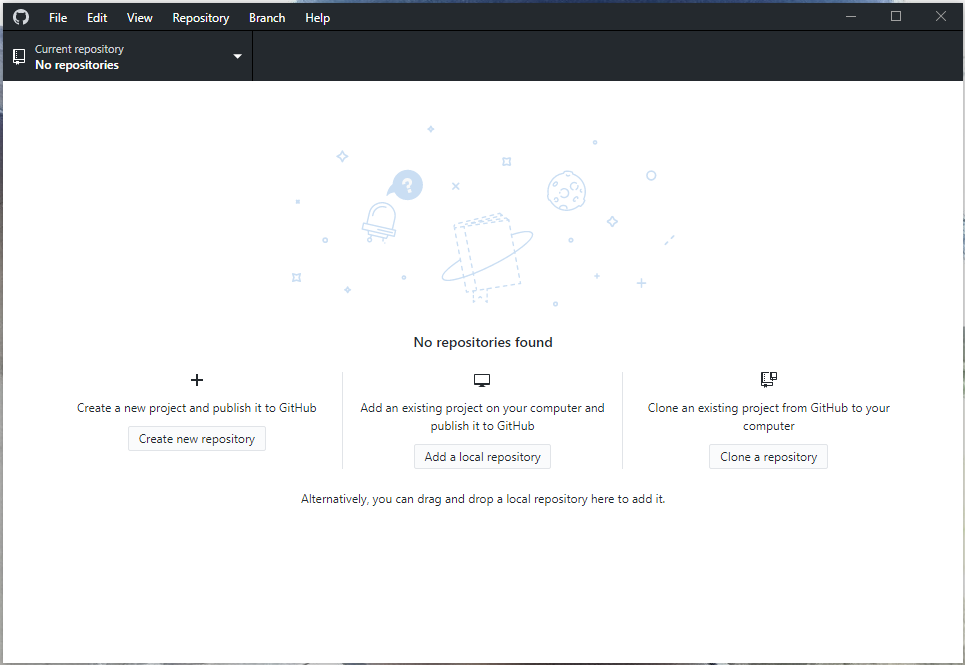


Fig. III‑33 Page d'accueil github desktop

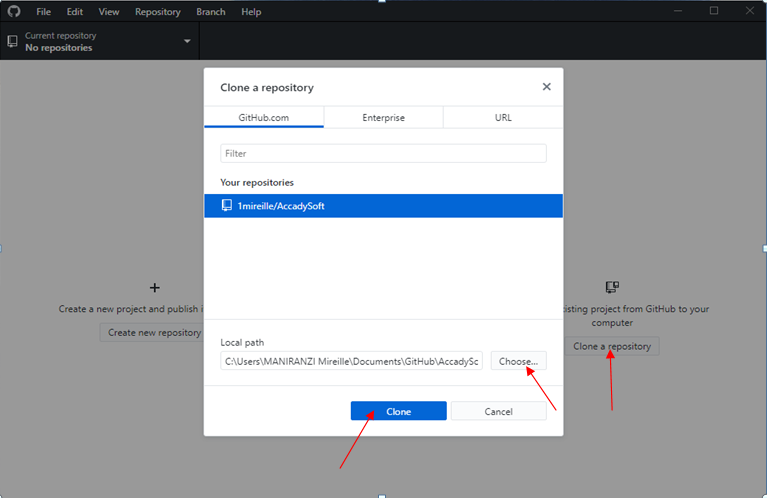


Fig. III‑34 Choix du dépôt à cloner

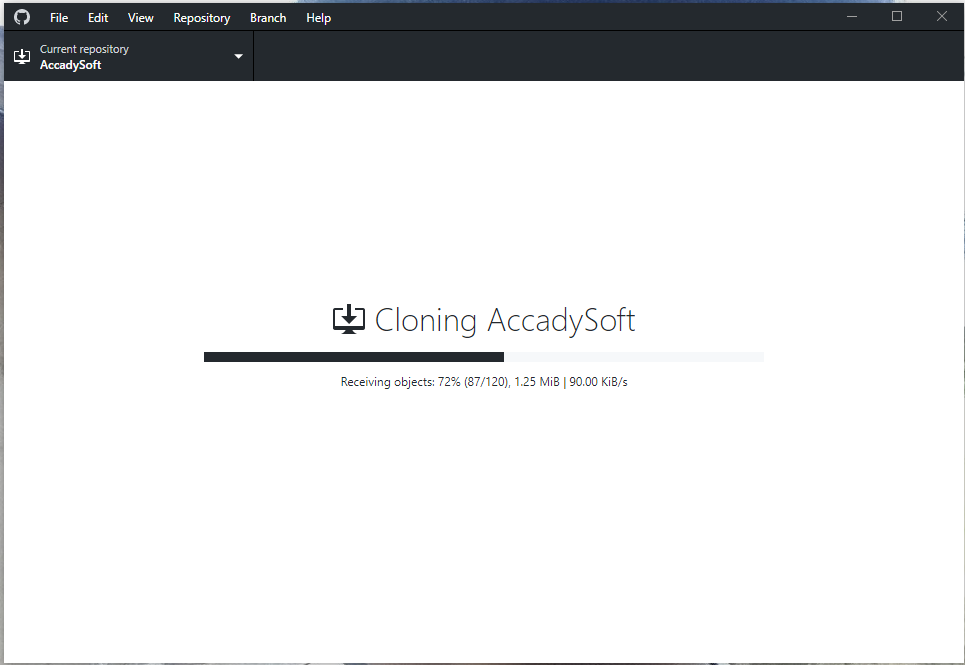


Fig. III‑35 Clonage en cours

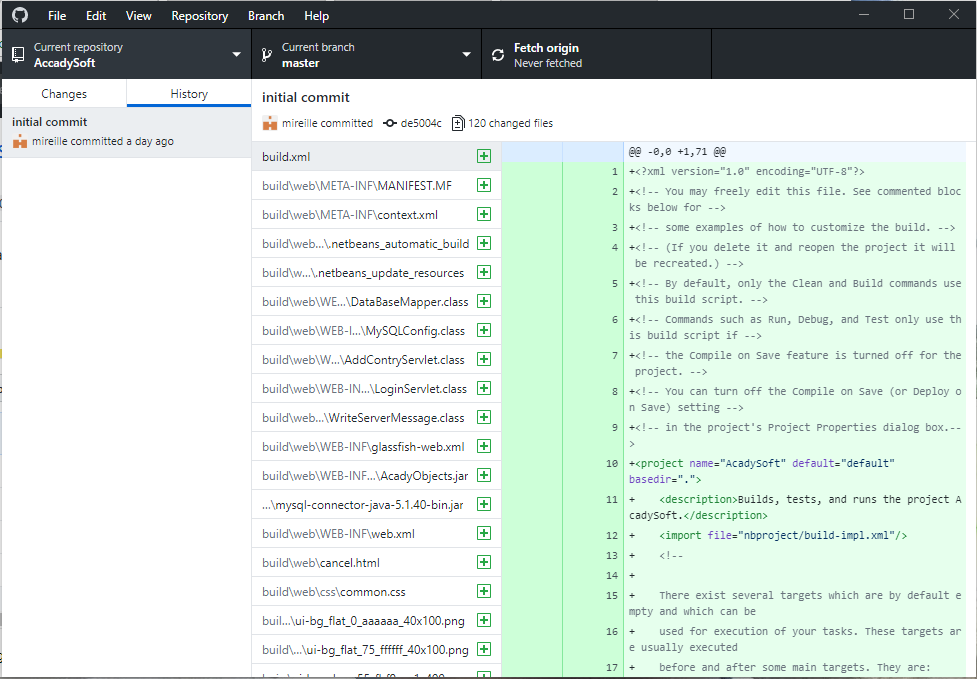


Fig. III‑36 Aperçue du dépôt cloné

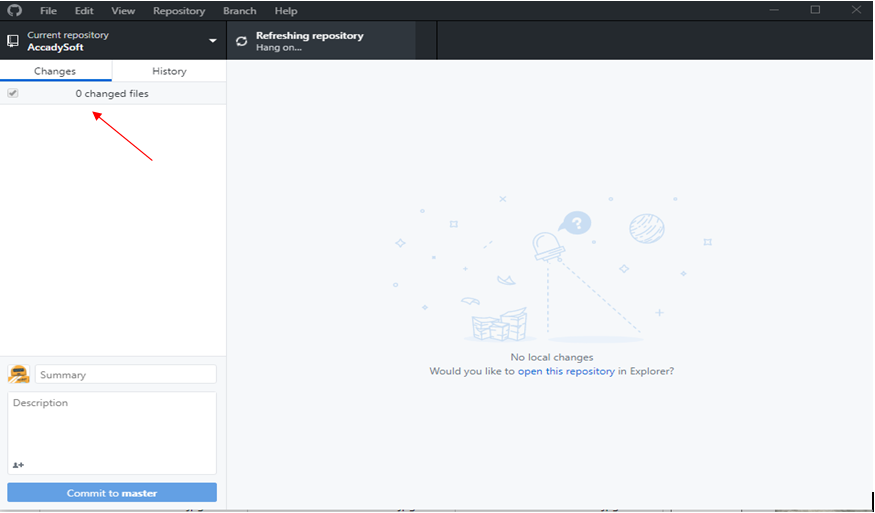


Fig. III‑37 Dépôt cloné mais pas encore modifié

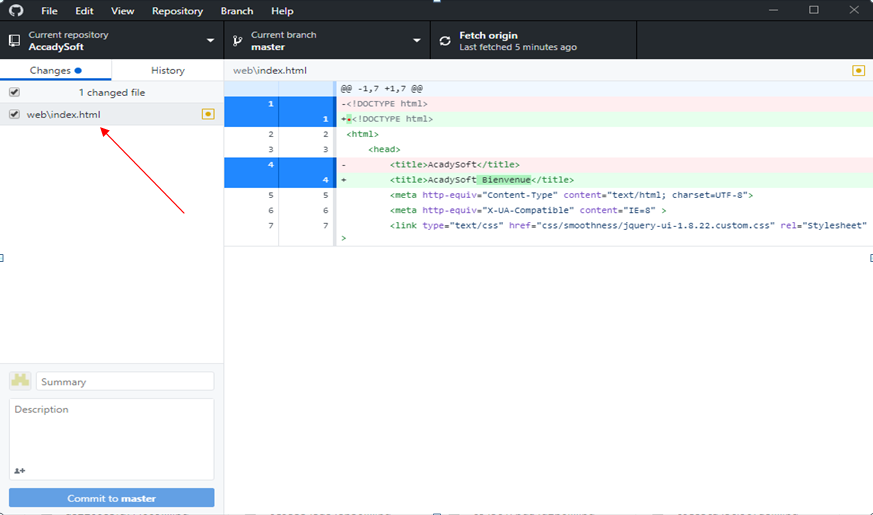


Fig. III‑38 aperçue du dépôt modifié

1. Nous avons ensuite envoyer les modifications sur le dépôt central.

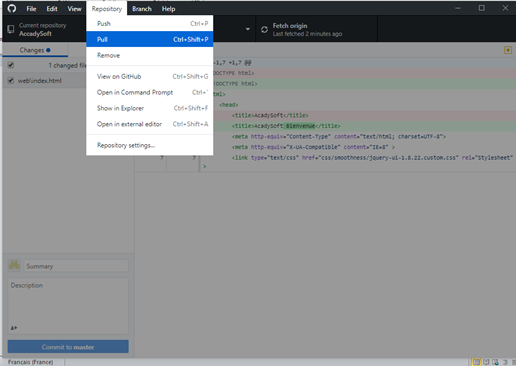


Fig. III‑39 push

1. pour actualiser le dépôt local il faire un pull



Fig. III‑40 git pull

## Serveur d’Intégration continue JENKINS

Dans cette partie nous avons combiné la gestion de version avec l’intégration continue proprement dit. Nous avons mis en place tous les outils requis, le serveur et sa configuration.

### Installation et configuration des outils

#### L’environnement de développement java(JDK)



Fig. III‑41 Assistant d'installation de jdk

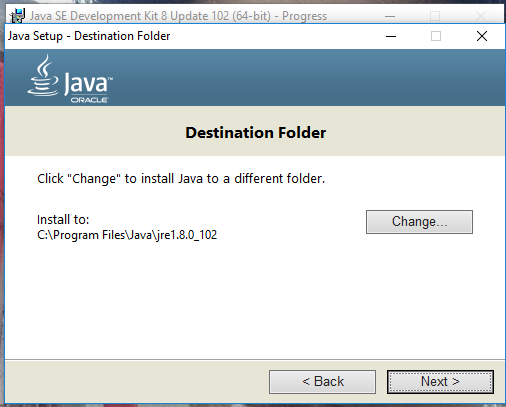


Fig. III‑42 Choix de l'emplacement de jdk

#### Le serveur xampp

Dans ce projet, nous avons utilisé le serveur apache Tomcat, qui est un conteneur d’application java. Nous avons alors téléchargé et puis installé xampp pour utiliser Tomcat comme module

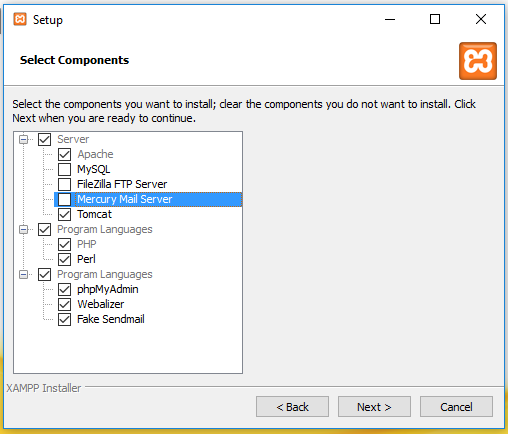


Fig. III‑43 Choix des modules de xampp

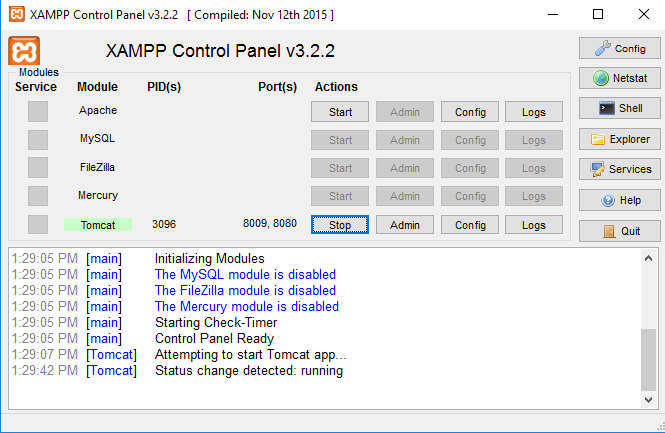


Fig. III‑44 Démarrage Tomcat dans xampp

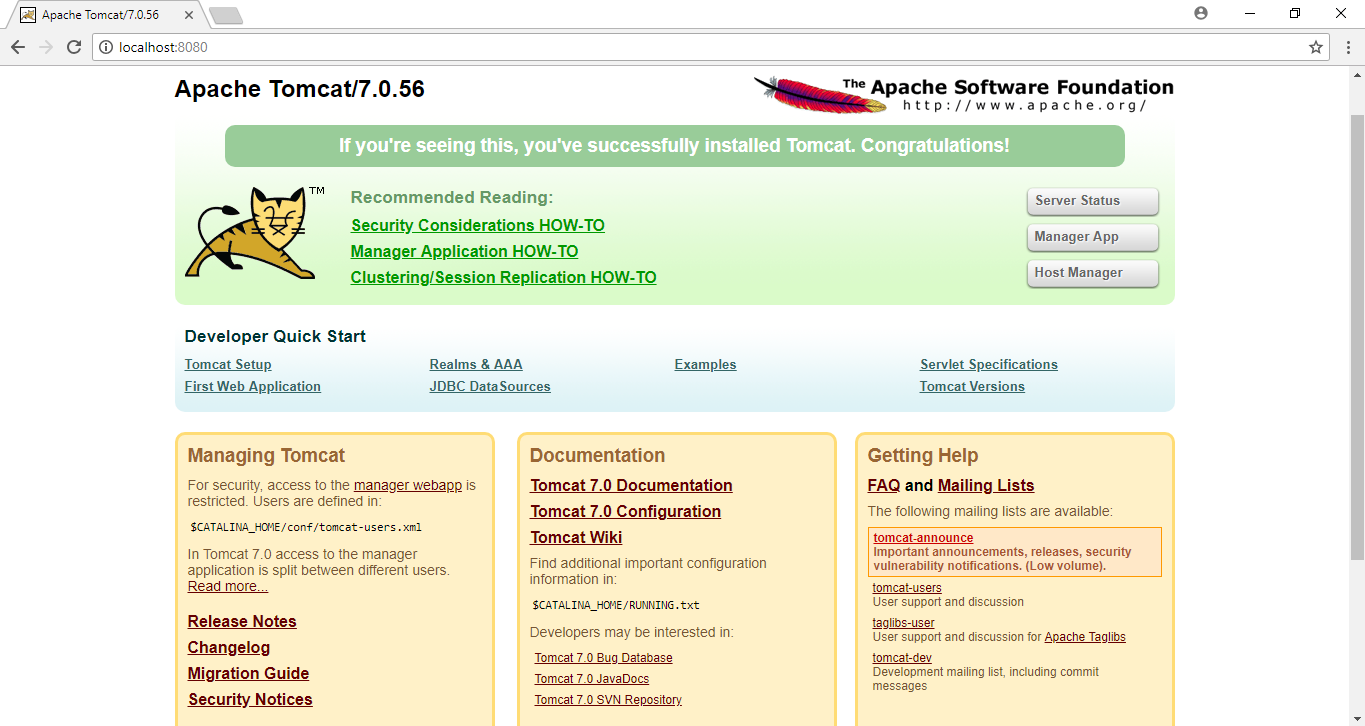
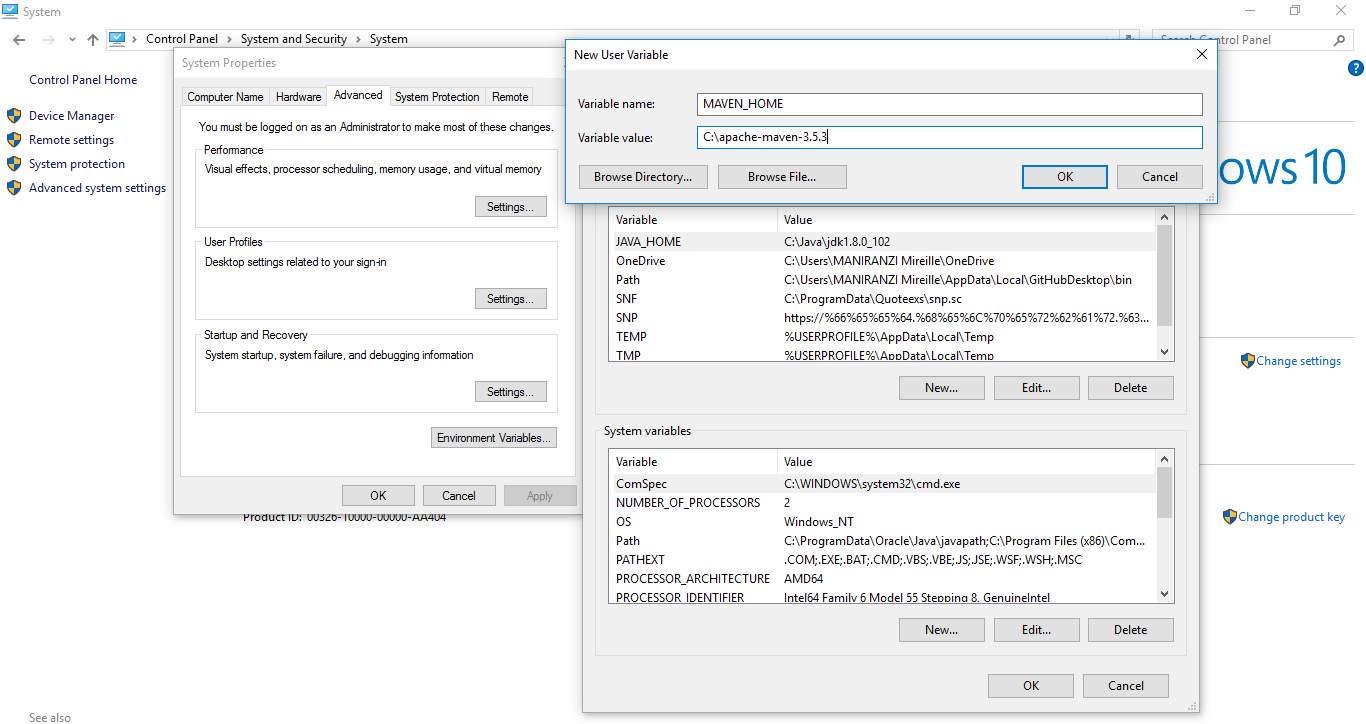


Fig. III‑45 Page d'accueil apache tomcat

#### Maven

Nous avons téléchargé le fichier compressé contenant apache Maven sur [https://maven.apache.org/download.cgi, nous](https://maven.apache.org/download.cgi,%20nous) l’avons décompressé puis ajouter aux variables d’environnement. Nous avons ouverts Control Panel\System and Security\System



**12**

**10**

**9**

**5**

**5**

**4**

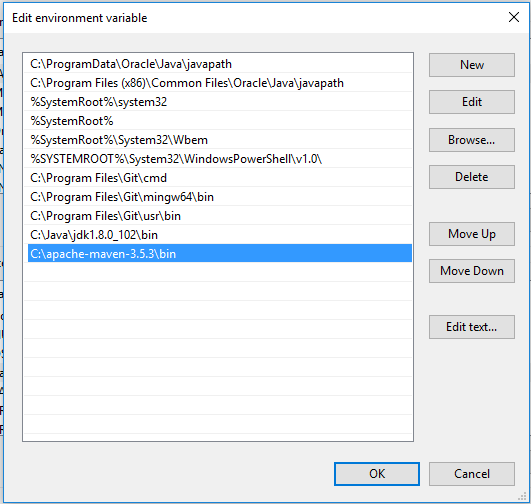
**3**

**2**

**1**

Fig. III‑46 Ajout d'une variable d'environnement pour Maven(1)

**3**



**8**

**7**

**6**

Fig. III‑47 Ajout d'une variable d'environnement pour Maven(2)

#### JENKINS

##### Téléchargement et deploiement

##### Apres avoir téléchargé le fichier jenkins.war, nous l’avons déployé sur Tomcat en plaçant le fichier jenkins.war dans ce répertoire*: C:\xampp\tomcat\webapps*

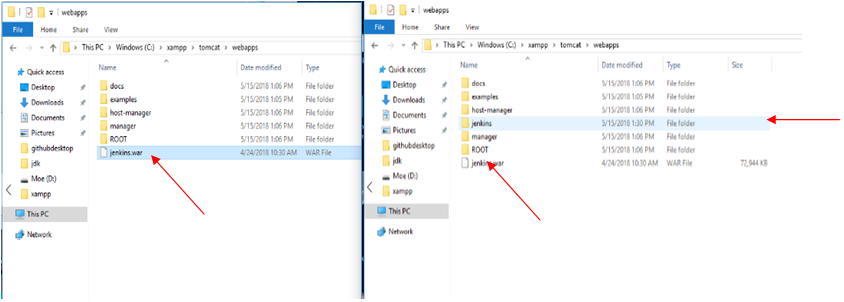


Fig. III‑48 Déploiement de jenkins

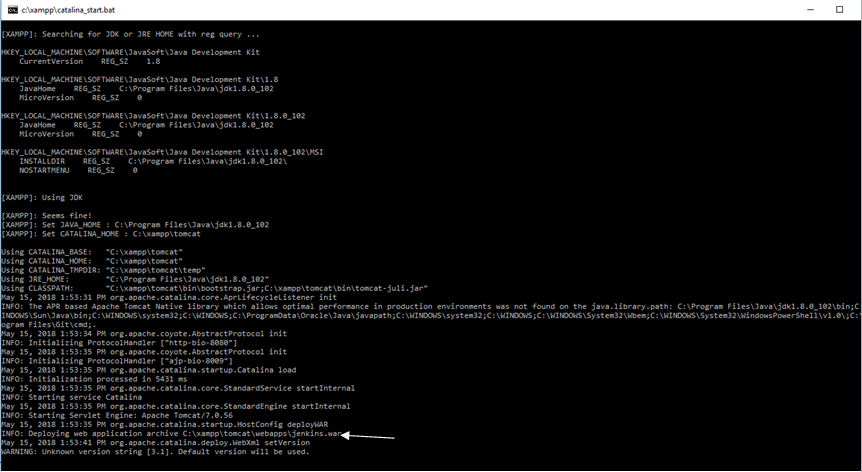


Fig. III‑49 Réussite de déploiement de jenkins dans xampp

##### Installation de Jenkins

En ouvrant le navigateur, nous avons puis accéder à la page d‘installation de Jenkins grâce à ce lien : [*http://localhost:8080/jenkins*](http://localhost:8080/jenkins)

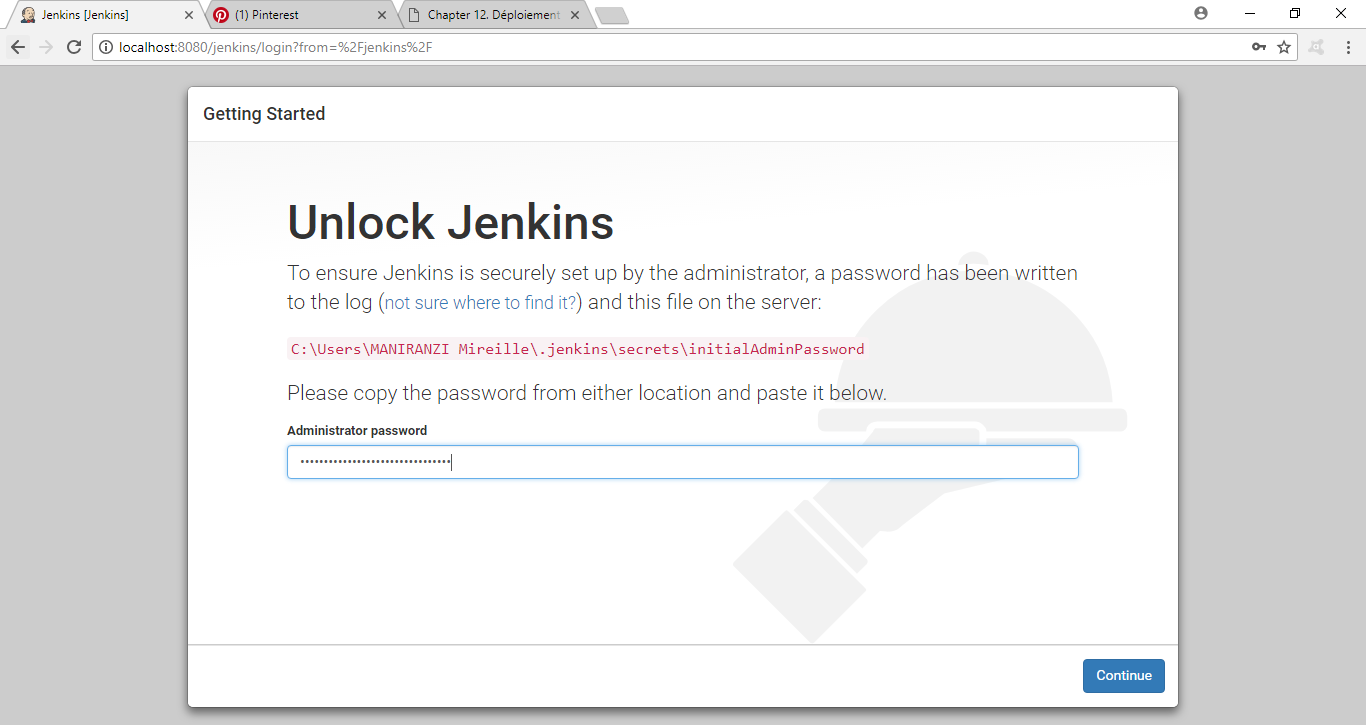


Fig. III‑50 Déblocage de jenkins

Fig. III‑51 Page pour débloquer Jenkins

##### Plugins choisis

* Jenkins ne propose pas moins de 600 plugins pour étendre ses capacités et les adapter aux besoins de chacun [1]. Il convient de choisir ceux qui correspondent le plus aux besoins du projet "Mise en place d’un système d’intégration continue”. Nous nous sommes contentés d’une rapide étude et d’indiquer quels plugins étaient réellement nécessaires.
* maven Project plugin; Ce plugin permet d'exécuter des builds Maven
* git plugin; Ce plugin permet l'utilisation de Git en tant que SCM de build.[[14]](#footnote-14)
* GitHub plugin ; Ce plugin intègre Jenkins avec des projets Github.
* Deploy container plugin
* Copy Artifact plugin

##### **Le tableau de bord de configuration**

L'écran Administrer Jenkins, permet de gérer pratiquement tous les aspects de la configuration du système dans l'écran Administrer Jenkins. Vous pouvez aussi atteindre cet écran directement depuis n'importe où dans l'application en tapant “manage” dans la boîte de recherche Jenkins. Cet écran change en fonction des plugins que vous installés [7]

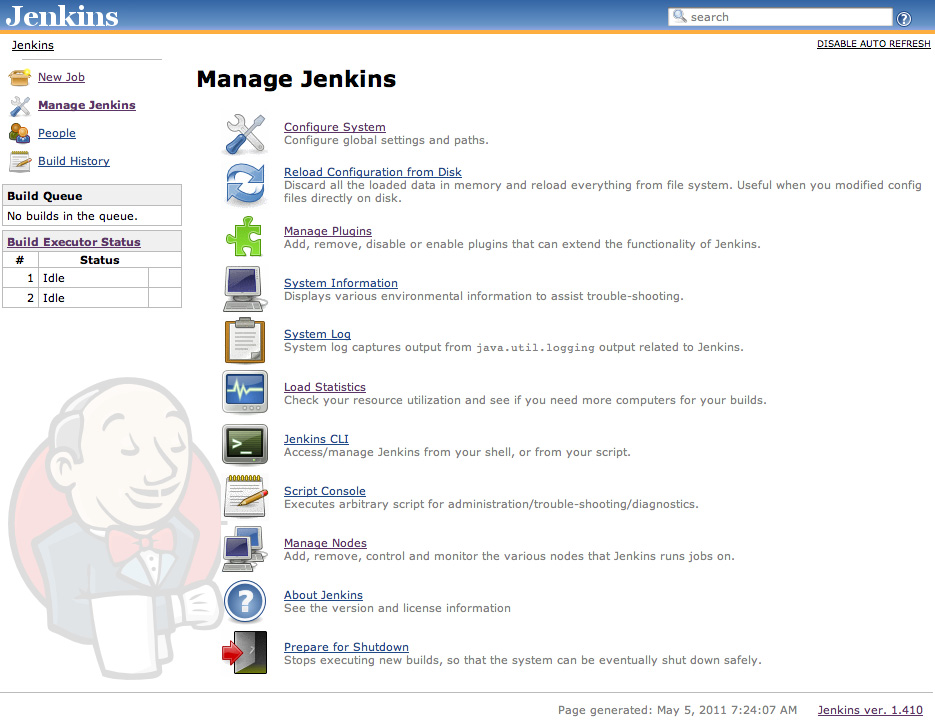


Fig. III‑52 Page d'administration de Jenkins

1. **Configuration du système dans Jenkins**
2. **jdk**

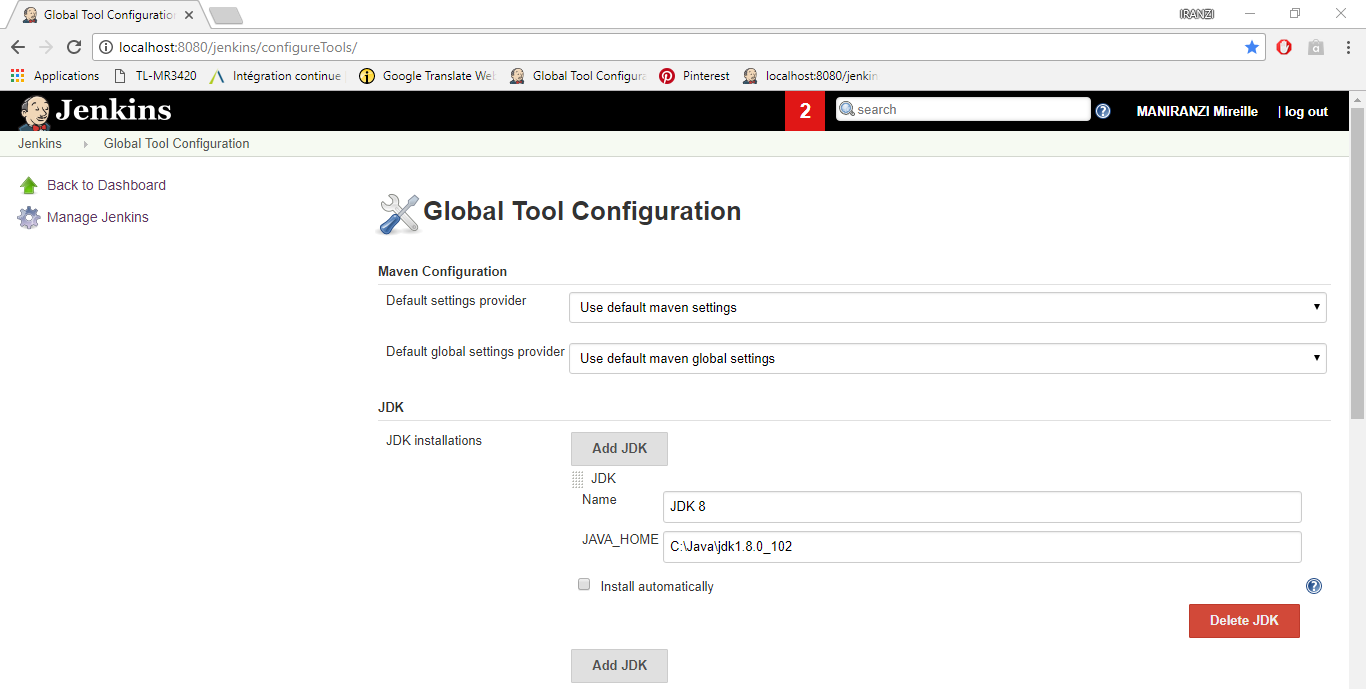


Fig. III‑53 Configuration Jenkins de jdk

1. **Configure d’un outil de gestion de version**

Jenkins arrive de base préinstallé avec des plugins pour CVS et Subversion. Les autres systèmes de gestion de version sont supportés par des plugins que vous pouvez téléchargé depuis l'écran Gérer les plugins. [7] Nous avons donc recherché le plugin pour git, nous l’avons sélectionné puis télécharger.

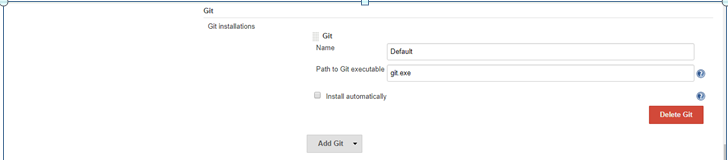


Fig. III‑54 configuration Jenkins de git

1. **Maven**

Maven est un frameworks de Scripting de haut-niveau pour Java qui utilise des notions telles qu'une structure normalisée d'arborescence et des cycles de vie normalisés, "Convention over Configuration”, et une gestion déclarative des dépendances pour simplifier le Scripting de bas-niveau que l'on trouve souvent dans un script de build Ant typique. Avec Maven, votre projet utilise un cycle de vie normalisé et bien défini — compile, test, package, deploy, etc. Chaque phase de cycle est associée avec un plugin Maven. Les différents plugins Maven utilisent la structure d'arborescence normalisée pour traiter ces tâches avec un minimum d'intervention de votre part. Vous pouvez aussi étendre Maven en redéfinissant les configurations de plugin par défaut ou en invoquant des plugins supplémentaires. Jenkins fournit un excellent support pour Maven, et comprend parfaitement les structures de projet et les dépendances Maven. Vous pouvez soit demander à Jenkins d'installer automatiquement une version de Maven, ou fournir un chemin vers une installation locale de Maven. Vous pouvez configurer autant de versions de Maven pour vos projets de build que vous le voulez, et utiliser différentes versions de Maven pour différents projets. [7]

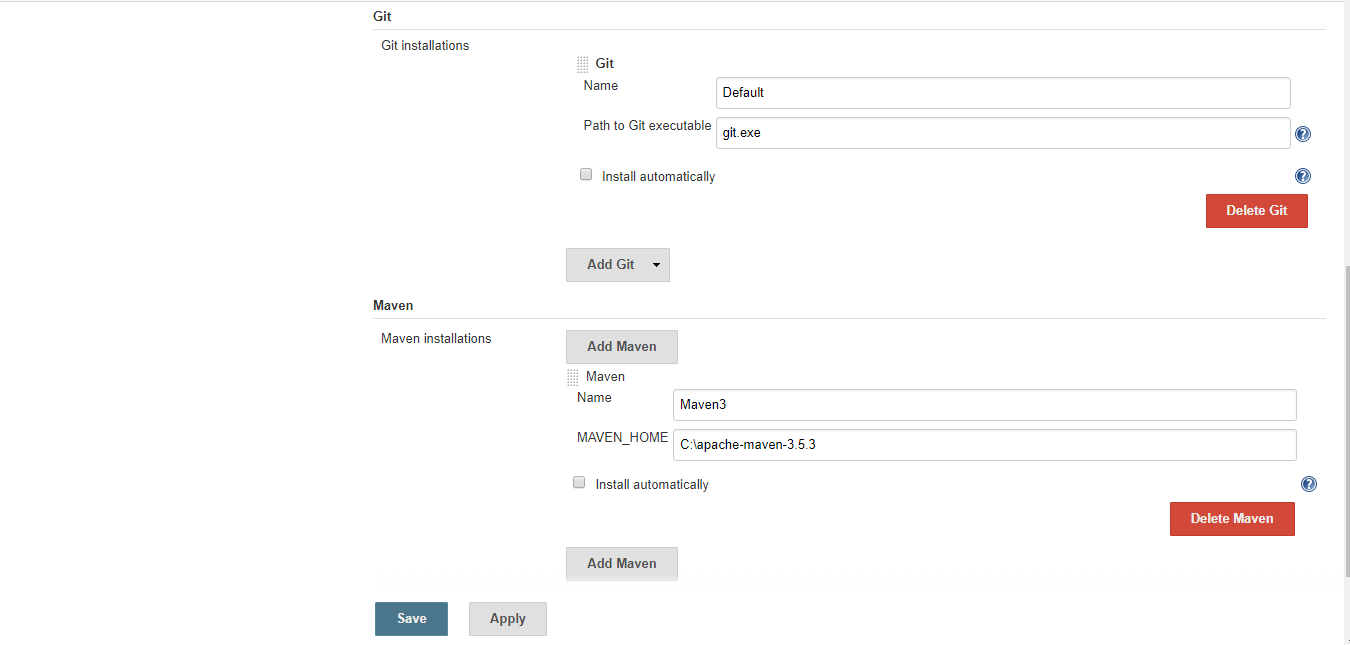


Fig. III‑55 Configuration Jenkins de maven

### Tâches de Build Jenkins free-style

Jenkins supporte quatre principaux types de tâches de build mais la tâche de build free-style est l'option la plus flexible et la plus configurable et peut être utilisée pour n'importe quel type de projet. Elle est assez rapide à mettre en place et la plupart des options de configuration vues ici sont également disponible dans les autres types de tâches de build.Nous avons pu créer deux jobs pour pouvoir déployer notre application java.les plugins dont on aura besoin sont :git plugin, github plugin, artefact copy plugin, deployer plugin et leurs dépendances.

#### Notre premier job

#### Il exécute deux grandes taches :

* Faire le build d’un projet situé sur github
* archiver l’artefact du build sous extension .war

#### Second job

Il exécute deux taches :

* Copier l’archive du build précèdent
* Déploie l’archive sur le serveur Tomcat.

**J’ai déjà réalisé ces deux builds.il me reste simplement d’ajouter des captures d’écran**

# Conclusion générale

Un Système d’intégration continue peut considérablement contribuer à l’optimisation de la qualité logicielle d’une application en rendant ses fonctionnalités plus fiables et performantes. Grâce à un processus automatique, elle peut également assurer un déploiement rapide et conforme aux délais des applications aussi complexes qu’elles soient. [5]

Notre projet de fin d’étude s’inscrit dans cette optique. Au terme de notre travail, nous avons pu définir et mettre en place une plateforme d’intégration continue. Cette plateforme assure des fonctionnalités d’automatisation de builds et déploiement, de gestion et contrôle de versions. Afin de mettre en place cette plateforme et après avoir établis une étude conceptuelle conforme au mécanisme standard d’intégration continue, nous avons été amenés d’abord à chercher et étudier un ensemble d’outils d’intégration continue existant sur le marché afin d’en choisir ceux adaptés aux besoins et contraintes de notre projet. Notre choix a porté sur les outils suivants : git, jenkins, Maven, github desktop, Tomcat, jdk. Cette opération a été conclue par l’installation de l’ensemble de ces outils mais aussi l’établissement de paramétrages et configurations nécessaires pour assurer la communication entre ces différents outils. Par ailleurs, en vue de valider la performance de la plateforme proposée et mise en place, nous avons également procédé à la réalisation de certains tests. Ces tests ont ciblé un projet de développement d’une application gérant les opérations académiques de l’UPG. Cependant, vu les délais accordés à notre projet, nous n’avons pas eu la possibilité de poussés loin notre travail.

En conclusion, nous aimerions noter l’intérêt de ce projet qui d’une part, nous était d’une grande importance puisqu’il nous a permis de travailler sur des aspects aussi bien théoriques que techniques de l’assurance qualité et d’approfondir nos connaissances en gestion de projet et de conception de projets réels. D’autre part, nous pensons que ce projet apporte une solution qui contribuerait à l’amélioration de la qualité et la livraison rapide des applications développées par quelqu’un qui pourra adopter cette pratique.

Nous suggérons aux étudiants qui trouveront ce sujet assez intéressant de l’approfondir.

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”
8. Aurelazy. (09 February 2017). “Petit tutoriel sur Git Documentation Version 1.0.0”

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://www.itilfrance.com/index.php?pc=pages/docs/pratique-05/171-04.inc&pb=haut\_entete\_pratique.inc [↑](#footnote-ref-3)
4. http://localhost:8080/docs/deployer-howto.html [↑](#footnote-ref-4)
5. http://igm.univ-mlv.fr/~dr/XPOSE2012/Integration%20Continue/concept.html [↑](#footnote-ref-5)
6. https://fr.wikipedia.org/wiki/Int%C3%A9gration\_continue [↑](#footnote-ref-6)
7. [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-7)
8. Mise en place d’une plateforme d’intégration continue au sein de la direction générale des impôts.pdf [↑](#footnote-ref-8)
9. <http://www.aubryconseil.com/post/La-gestion-des-bugs-dans-le-projet-IceScrum> [↑](#footnote-ref-9)
10. <https://www.codebuilder.fr/blog/developpement-collaboratif-logiciels-gestion-versions/> [↑](#footnote-ref-10)
11. <https://www.codebuilder.fr/blog/developpement-collaboratif-logiciels-gestion-versions/> [↑](#footnote-ref-11)
12. [http://linsolas.developpez.com/Chapitres/java/outils/builds/](http://linsolas.developpez.com/articles/java/outils/builds/) [↑](#footnote-ref-12)
13. <http://linsolas.developpez.com/articles/java/outils/builds/> [↑](#footnote-ref-13)
14. http://www.vogella.com/tutorials/Jenkins/article.html [↑](#footnote-ref-14)