Mise en place d'un système d'intégration Continue avec Jenkins et des tests automatisés : cas de PlexusERP

Rapport de stage

Master Sécurité des Systèmes Numériques

Présenté par

DONTSA TAKOUDJOU Cyprien

Effectué dans la société HADRON S.A

Tuteur entreprise : M. NDOUNA Alex

Tuteur universitaire : M. ABDOU Wahabou

Année universitaire 2019 - 2020

Dédicace

Je dédie ce travail

A la Famille DONTSA

Remerciements

La rédaction et la réalisation de ce rapport n'auraient pu voir le jour sans les faveurs que Dieu par sa miséricorde m'a accordé directement ou à travers les abnégations et le soutien de nombreuses personnes. L'occasion m'est donnée d'exprimer ma reconnaissance et ma profonde gratitude :

- Au Pr DIPANDA Albert, Directeur de l'ESIREM qui a eu l'initiative de créer cette formation et qui s'est toujours assurer de son bon déroulement ;
- A mon tuteur académique Dr ABDOU Wahabou et aussi en tant que responsable académique de ce Master qui n'a fournir aucun effort pour le bon déroulement de cette formation ;
- A mon tuteur professionnel M. NDOUNA Alex pour ses conseils pratiques ;
- A tout le corps enseignant et responsable de ce Master ;
- A mes parents DONTSA René et SONDJOU Georgette, mes frères et sœurs Bertold, Boenish, Ariane Roswita, pour leur soutien multiforme ;
- Un merci particulier à Morelle Nantia pour sa bienveillance et son attention ;
- A tous mes promotionnaires de ce Master ;
- A tous ceux qui de près ou de loin ont contribué à la réalisation de ce rapport et dont le nom ne figure pas dans ce rapport, je vous dis humblement merci.

Sommaire

Liste des Figures

Liste des Tableaux

Sigles

CD: Continuous Development

CRM: Customer Relationship Management

CNRPH : Centre National de Réhabilitation des Personnes Handicapées

ERP: Enterprise Resource Planning

IC: Integration Continue

MINESUP : Ministère des enseignements supérieurs

MINTP : ministère des travaux publics

MIT: Massachusetts Institute of Technology

QA: Quality Assurance

SA : Société Anonyme

SCM : Supply **C**hain **M**anagment

Résumé

Pour clôturer le cycle Master, option Sécurité des Systèmes Numériques de l'université de Bourgogne, les étudiants de master II sont appelés à faire un stage en entreprise de 4 à 6 moins. Ce stage qui s'intègre directement à notre cursus académique nous permet non seulement de parfaire les connaissances théoriques par la pratique mais aussi de familiariser le futur employé au monde professionnel. Mon stage à cet effet c'est effectué à Hadron S.A.

Durant ce stage, j'ai été mené à plusieurs activités tels que le développement des modules web, l'administration système et le déploiement d'application. Ces activités ont été menées en parallèle avec mon thème de stage intitulé : '' ***Mise en place d'un système d'intégration continue avec Jenkins et des tests automatisés : Cas de PlexusERP''***.

Pour atteindre efficacement notre objectif, nous avons fais ressortir le contexte, la problématique et la méthodologie adaptée à notre thème. Le serveur d'intégration continue (IC) Jenkins à été installé et configuré pour assurer le bon fonctionnement et la compilation du projet PlexusERP.

Abstract

To end the Master cycle, Digital Systems Security option at the University of Burgundy, Master II students are required to do an internship of 4 to 6 months. This step, which is directly integrated into our academic curriculum, allows us not only to perfect theoretical knowledge through practice but also to familiarize the future employee with the professional world. My internship has this effect at Hadron S.A.

During this internship, I was led to several activities such as web module development, system administration and application deployment. These activities were carried out in parallel with my internship topic entitled: "Setting up a Continuous Integration System with Jenkins and Automated Testing: Case of PlexusERP".

To effectively achieve our objective, we have highlighted the context, the problematic and the methodology adapted to our theme. The Jenkins Continuous Integration (CI) server has been installed and configured to ensure proper functioning and compilation of PlexusERP projects.

INTRODUCTION

Les nouvelles technologies de l'information et de la communication ont été la révolution la plus importante qui ont marquées ses dernières décennies. Loin d'être un phénomène éphémère, ces technologies nous ont apportés du confort dans notre vie quotidienne par leurs capacités à traiter l'information dans les délais raisonnables. Plusieurs entreprises se sont rythmées à ces technologies en créant des solutions viables et efficaces pour des besoins utilisateurs qui ne cessent de croitre.

Hadron S.A ayant pris conscience des besoins utilisateurs a décidé d'orienter ses activités dans la mise en place d'un ERP, nommé PlexusERP qui s'adapte aux besoins des établissements publics et parapublics. Fondé par trois jeunes brillants ingénieurs des prestigieuses écoles américaines, Hadron S.A propose des solutions uniques et révolutionnaires notamment la gestion du patrimoine (MINTP et MINESUP) et la gestion des ressources humaines et de la solde (CNRPH). C'est dans cette lancée qu'HADRON S.A à bien voulu nous accueillir au sein de son organisation et m'a chargé de mettre en place un système d'intégration continue avec Jenkins et des tests automatisés.

Le présent rapport de stage est organisé en trois parties. Dans un premier temps nous donnerons un aperçu de la structure d'accueil, ensuite je vais ressortir le contexte, la problématique et la méthodologie adaptée a notre thème, et enfin je vais vous présenter l'outil d'intégration continue choisis et son implémentation.

CHAPITRE 1 : PRESENTATION DE LA STRUCTURE D'ACCUEIL

1.1 Présentation d'HADRON SA

Fondée en 2015 par des jeunes sortis de la prestigieuse école du MIT, Hadron S.A est une société d'ingénierie logicielle et de technologie de l'information qui se concentre sur la transformation numérique des organisations publiques et privées à travers le monde. S'il est indéniable que la technologie est et continuera de perturber et de transformer énormément chaque secteur d'activité, Hadron S.A propose de soulager les clients de la frustration de l'exclusion numérique ceci à travers leur détermination à concrétiser leurs rêves numériques.

Sa mission est de combler le fossé numérique de chaque entreprise et organisation qui se présente à nous. Face aux enjeux de l'inclusion numérique, Hadron S.A aide les entreprises à adopter une approche pérenne et efficace ainsi qu'à intégrer la technologie dans leurs objectifs stratégiques.

1.2 Organisation d'HADRON SA

Hadron S.A dispose de plusieurs locaux dont son Siège est à Boston (Massachusetts, États-Unis), une branche à Paris (France), une autre branche à Yaoundé (Cameroun) et à Abidjan (Côte Ivoire) avec un personnel compris entre 11 et 50 employés. La figure suivante présente l'organigramme d'Hadron S.A (Yaoundé).

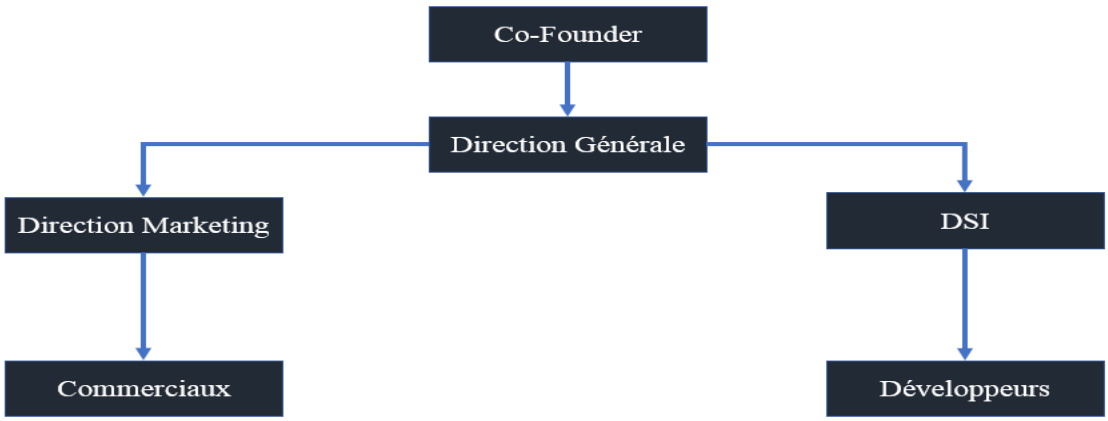


Figure 1: Organigramme Hadron S.A (Yaoundé)

1.3 Projet PlexusERP

PlexusERP est un ERP Flexible et moderne adapté pour les établissements publics et parapublics. Il contient plusieurs modules à savoir :

- La gestion des finances, Budgets et Dépenses
- La gestion des Ressources humaines
- La gestion des approvisionnements, des marchés publics et des CRM
- La gestion du patrimoine
- La gestion des dossiers, courriers et processus (BPM)

Techniquement, PlexusERP est caractérisé par :

- ✓ Une recherche avancée
- ✓ Qualité (Données valide)
- ✓ Systèmes de paiements améliorés
- ✓ Technologie intégrée
- ✓ Données sécurisées
- ✓ Disponible sur le cloud, intranet et accès mobile

PlexusERP propose également plusieurs services à savoir la consultation en TIC, les applications ERP (mobile et web), l'analyse Big Data, la résolution des problèmes liés aux matériels et aux réseaux informatiques, la gestion des projets et les formations professionnelles par des experts des Etats unis.

CHAPITRE 2 : CONTEXTE, PROBLÉMATIQUE, OBJECTIFS DU STAGE

2.1 Contexte

Le cycle de vie d'un logiciel informatique est constitué de plusieurs phases à savoir le recueil des besoins, la conception générale et détaillée, le codage, les tests et la mise en production du produit. C'est donc une succession de phases tout en précisant que la phase précédente soit terminée avant de passer à la suivante.

Cette méthode de travail dite classique est stricte et ne favorise pas la flexibilité de l'évolution du projet, donc il est nécessaire de tout faire correctement dès le début du projet. La phase cruciale qui consiste à mettre en production un produit est le déploiement. C'est ainsi que plusieurs contraintes naissent, les exigences changes sans cesse [2], le développement doit être de plus en plus rapide, on doit pouvoir ajouter sans difficultés des nouvelles fonctionnalités tout en corrigeant les bugs. Ceci conduit donc à un modèle de développement agile [3].

Cependant, l'équipe de développement n'est pas la seule à intervenir sur le projet ; les opérationnels doivent déployer et surveiller les nouvelles applications. Une approche dite DevOps [4] est donc mise en place pour améliorer la communication entre ces deux équipes. La bonne implémentation de la culture DevOps consiste à mettre en place des outils de développement continue, d'intégration continue (IC), de déploiement continue, de test continue et de supervision continue.

Afin d'améliorer la qualité des futurs projets il est donc capital de faire usage à des bonnes pratiques de développement informatique, d'où l'entrée en jeu de l'IC, constituée principalement de 5 étapes :

- **Build** : étape de compilation
- **Test**: étape de test execution
- **QA** : étape de test d'assurance qualité
- **Deploy** : étape de déploiement
- **Monitor** : étape de supervision

Dans le contexte de ce stage un pipeline d'IC a été mise en place contenant un certain nombre de fonctionnalités opérationnelles. Il est question d'apporter des modifications en incluant des étapes de tests tout en tenant compte de l'aspect sécurité jusqu'ici non implémenté.

2.2 Problématique

Afin de maitriser le processus de déploiement et de corriger les éventuels bugs, nous allons implémenter un environnement d'intégration continue. Toutefois, comment réduire le temps de mise sur le marché du logiciel ? comment repérer et corriger les erreurs plus rapidement afin d'augmenter la fiabilité du logiciel ? Comment mettre en place un processus de rétroaction afin d'avoir un retour sur expérience sur chaque étape du processus de déploiement ?

2.3 Objectifs

L'objectif principal de ce stage de fin d'étude est de mettre en place un système d'intégration continue et des tests pour l'ERP PlexusERP. Pour atteindre cet objectif, il sera question pour moi de :

- Mettre en place l'outil d'IC choisis
- D'implémenter un pipeline
- Ecrire les scripts de déploiements automatisés
- Ressortir un rapport des étapes d'exécutions du pipeline

CHAPITRE 3 : METHODOLOGIE

3.1 Définition de l'intégration continue

Depuis des années, durant le processus de développement de la plupart des projets informatiques, le logiciel ne se trouve pas dans un état de fonctionnement. Le logiciel est donc inutilisable durant la majorité du temps de développement. Ainsi les développeurs apportent des modifications sur le code, exécutent des tests unitaires sans toutes fois évaluer le comportement de l'application dans un environnement semblable à celui de production.

Ceci est doublement vrai dans des projets qui reportent les tests d'acceptations jusqu'à la fin. Beaucoup de ces projets programment les phases d'intégrations à la fin du développement pour permettre à l'équipe de développement de fusionner leurs modifications de telle sorte que ces dernières passent les tests d'acceptations. Etant donné que certains projets se rendent compte que quand ils arrivent à la phase d'intégration, le logiciel produit ne fonctionne pas comme prévu. Ce qui implique des phases d'intégrations relativement longues et le pire c'est que le temps passé à réaliser cette phase n'est pas quantifiable [8]. Cependant certains projets informatiques échappent à cette situation en restant dans un état de non-fonctionnement durant une courte période due à des modifications du code source puis passent à un état de fonctionnement. Ceci est due à de l'intégration continue.

Selon **martin Fowler** [6] l'IC est vue comme « … *une pratique de développement logiciel où les membres d'une équipe intègrent leur travail fréquemment, habituellement chacun au moins une fois par jour – ce qui entraine plusieurs intégrations par jour. Chaque intégration est validée par un '**build'** automatique (ce qui inclut les tests) pour détecter les erreurs d'intégrations aussi vite que possible. Cette approche permet de réduire significativement les problèmes d'intégrations et permet à une équipe de développer des logiciels de qualités plus rapidement* … ». L'IC est une méthode de développement logiciel dans laquelle le logiciel est reconstruit et testé à chacune modification apportée par un programmeur. La série de figure suivante compare les projets avant et après l'IC.

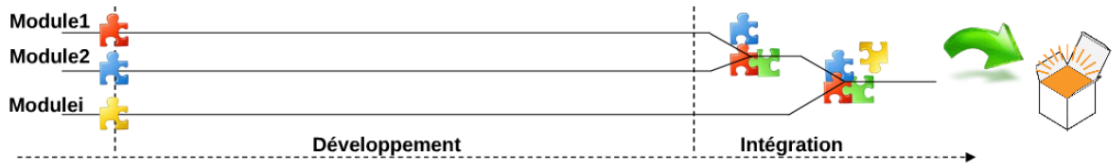


Figure 2: Développement logiciel avant l'IC

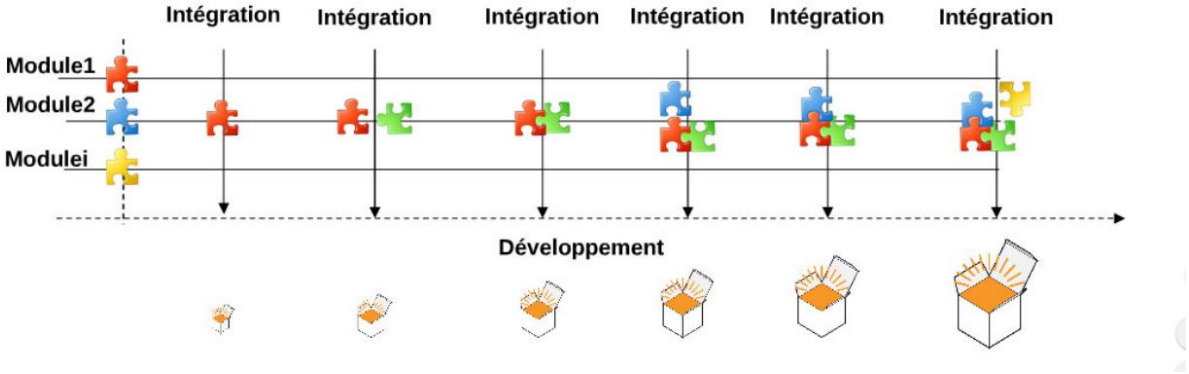


Figure 3: Développement logiciel après l'IC

Les deux diagrammes suivants en image reprennent le principe des précédents en abordant le nombre de bugs dans la comparaison.

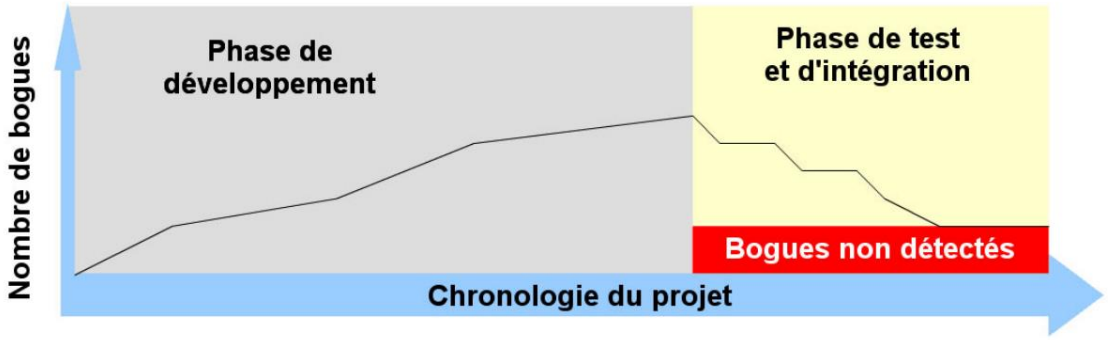


Figure 4: Nombres de bugs avant l'IC

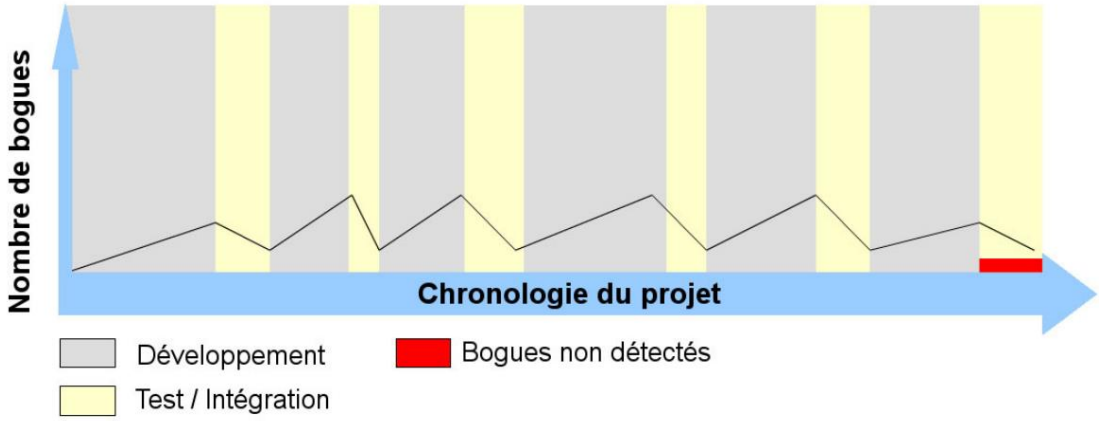


Figure 5: Nombres de bugs après l'IC.

L'objectif de l'IC est de s'assurer que le logiciel puisse fonctionner pendant tout le temps du processus de développement. Ceci nécessite de mettre en place des bonnes pratiques dont feront usages les équipes de développement.

3.2 Architecture et Fonctionnement de l'intégration continue

3.2.1 Architecture de l'IC

L'architecture comporte différents éléments n'intervenant pas uniquement dans l'intégration continue mais plutôt dans des équipes de développements en général. Le build est un terme anglophone qui désigne l'activité principale qui conduit à la construction de l'application. Il comprend plusieurs tâches comme :

- Un gestionnaire de code source
- Un logiciel d'intégration continue
- Une équipe de développeurs
- Un outil de reporting ou/et d'un serveur de suivi de bug

Le schéma suivant présentant les étapes (numérotées de 1 à 9) et les acteurs de l'intégration continue.

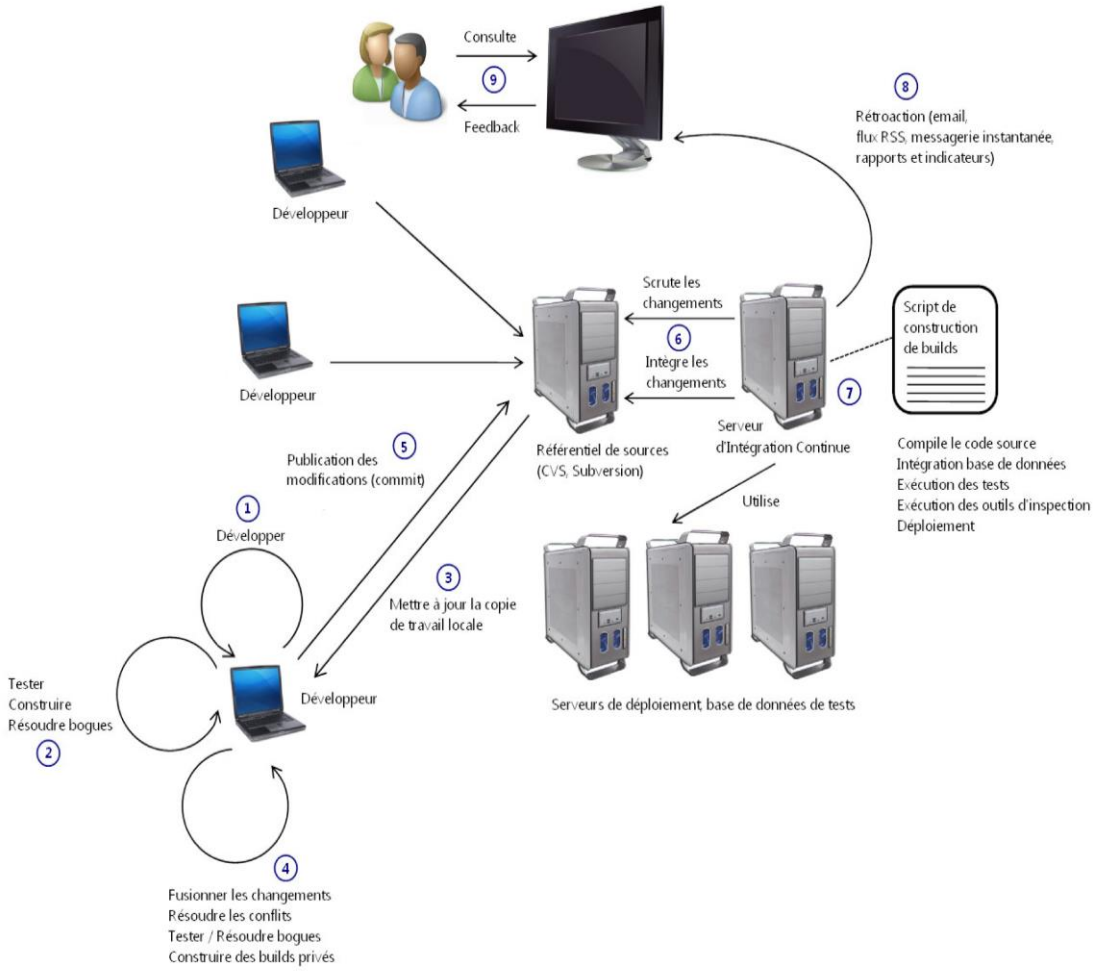


Figure 6: Architecture de l'IC [6]

3.2.2 Fonctionnement de l'IC

L'intégration continue se résume en trois étapes comme suit :

Etape 1 : les développeurs publient (commit) des modifications effectuées sur le code source sur le serveur de gestionnaire de version. Avant de le faire il se rassures d'avoir fait des tests et des builds localement sur leurs postes de travail. Le processus complet de cette etape est décrit comme suit :

a) Le développeur récupère la dernière version datée à partir du gestionnaire de version

b) Il effectue des modifications qui lui sont propre en fonction des taches qui lui sont affectées
c) Une fois terminée il lance un build localement
d) Si le build réussit il peut envisager faire un commit
e) Il peut donc arriver que d'autres développeurs ont fait des commits avant lui
f) Il doit donc mettre à jour sa copie locale afin de récupérer les dernières modifications
g) Ensuite faire un nouveau build local. En cas d'échec le développeur doit résoudre le problème
h) Une fois le build réussit le développeur pourra enfin commiter ces changements
i) Une fois le code commité, le serveur d'intégration continu lance un build d'intégration
j) Le développeur reçoit une notification lui indiquant le statut du build

Etape 2 : Le serveur d'intégration scrute les changements sur le gestionnaire de version. A chaque fois qu'il analyse un changement, il réalise un build d'intégration. C'est une méthode d'automatisation "on stimulus". Il est important de noter que le serveur d'intégration continue utilise des scripts et des environnements de build déjà existant.

Parce que le serveur d'intégration continue utilise un système de build déjà existant, il est important pour l'équipe de développement de consacrer du temps et des efforts à :

- Diminuer, optimiser au maximum le temps de création d'un build ;
- Ecrire des tests ;
- Inclure ces tests dans la construction du build.

Prendre du temps sur ces éléments, même si cela peut demander des modifications en profondeur sur le projet, est indispensable pour être en accord avec l'intégration continue. Non seulement le processus de build sera amélioré, mais c'est la qualité globale du projet qui le sera également. Cette méthode d'automatisation [8] "on the clock" permet d'avoir une charge moins importante sur le serveur, mais peut se révéler moins efficace, dans la résolution des erreurs, qu'une construction de build à chaque commit (notamment si les développeurs ont commité beaucoup de codes en l'espace de 30 minutes). Plus il y a de builds construits, plus il sera facile de diagnostiquer l'origine d'un problème de compilation ou de comprendre pourquoi certains tests sont en échec. En effet, il y aura moins de changements entre les deux builds.

Etape 3 : A chaque fois qu'un build est réalisé, un courriel de notification est envoyé à tous les membres de l'équipe. Ce courriel contient notamment le statut du dernier build. En cas d'échec du build, les développeurs pourront travailler directement sur sa correction. En plus des courriels automatiques, les développeurs consultent régulièrement le portail du serveur d'intégration depuis leurs postes ou sur un écran dédié. Il contient des indicateurs parlants et des statistiques. En cas d'échec d'intégration du projet, on peut combiner les méthodes, par exemple, envoyer un courriel aux développeurs et publier un message d'erreur sur une page web de synthèse. Les moyens disponibles varient selon les outils d'intégrations continues.

3.3 Avantages de l'intégration continue

Le processus le plus aboutis d'intégration continue automatise les tests en font un élément prioritaire du projet de développement informatique. Il a donc plusieurs avantages à savoir :

Améliorer la communication dans l'équipe de développement : l'IC permet de créer un cadre automatisé à la communication entre les membres de l'équipe, qui doivent communiquer au reste de l'équipe ceux sur quoi ils ont travaillé de manière régulière et synthétique. Cela permet d'établir un environnement transparent, où les membres de l'équipe peuvent se voir et évaluer le travail de chacun.

Une meilleure cohérence entre chaque partie du code : l'un des grands avantages de l'IC est la capacité du processus de garantir que le travail de développement effectué par les différents membres de l'équipe sur un projet commun est cohérent et conforme aux objectifs généraux et aux résultats escomptés du projet.

La diminution des régressions et bugs : en effectuant des tests automatisés et des examens d'assurances qualités (QA), le risque que des bugs se glissent dans le produit final est considérablement réduit.

Rapport coût efficacité très intéressant : l'IC réduit la courbe d'apprentissage des développeurs, qui peuvent rentrer facilement dans un projet sans avoir à se familiariser avec tout son architecture technique déjà construit.

Un gage de confiance pour l'équipe et le client : la confiance tant pour les développeurs que pour le client qu'un projet livré a été testé de manière robuste et continue, et a bénéficié d'un système conçu pour maximiser l'efficacité, la qualité, et la réduction des risques.

Selon la plateforme d'hébergement IONOS by 1&1, les avantages et inconvénients de l'IC sont résumés dans le tableau suivant :

Tableau 1: Avantages et inconvénients de l'IC : source (https ://www.ionos.fr/digitalguide/sites-internet/developpement-web/integration-continue/)

Avantages	Inconvénients
Possibilité de recherche précoce des erreurs	Conversion de processus habituels
Feedback permanent	Nécessite des serveurs et des environnements supplémentaires
Pas de surcharge contrairement à une seule grande intégration finale	Nécessité de mettre au point des processus de test adaptés
Enregistrement précis des modifications	Si plusieurs développeurs souhaitent intégrer leur code approximativement au même moment, des délais d'attente peuvent survenir
Disponibilité continue d'une version actuelle opérationnelle	
Nécessité d'un travail progressif	

CHAPITRE 4 : ETAT DE L'ART

4.1 Etude Comparative des outils d'intégration continue

De nombreux outils d'intégrations continues diffèrent sur Internet. Ils ont tous vocation à aider les développeurs dans la mise en œuvre de l'intégration continue et y parviennent de différentes façons avec des fonctionnalités bien spécifiques. Mais les outils d'IC ne se distinguent pas uniquement par l'étendue de leurs fonctionnalités, on constate également de grandes différences en termes de prix et licences. Alors que bons nombres d'entre eux sont des logiciels open source disponibles gratuitement, certains fabricants proposent également des outils payants. Nous présentons ici les outils d'IC les plus appréciés, leurs caractéristiques et leurs fonctionnalités.

4.1.1 Jenkins

Dédié aux DevOps, Jenkins [10] est un outil d'intégration continue open source (sous licence MIT) développé en Java. A chaque modification de code d'une application dans le gestionnaire de configuration, Jenkins se charge automatiquement de la recompiler, et de la tester. Pour cette seconde étape, Jenkins intègre le framework de test open source unit. En cas d'erreur détectée, Jenkins alerte le développeur afin qu'il résolve le problème. Un process évidemment des plus avantageux dans le cadre d'un projet de développement. Le logo de Jenkins est le suivant :

Figure 7: Logo Jenkins

Depuis 2005 (à l'époque sous le nom de Hudson), le logiciel a été constamment amélioré. Aujourd'hui, ce logiciel programmé sous Java offre de nombreuses fonctionnalités et interfaces contribuant à faciliter non seulement l'intégration continue, mais aussi la livraison et le déploiement continus.

4.1.2 Travis CI

Travis CI [12] est un logiciel d'intégration continue open source disponible sous licence MIT. Travis CI permet non seulement de tester des applications ou logiciels en développement, mais aussi d'effectuer automatiquement leur déploiement. Travis CI est un outil gratuit pour les projets open source.

Figure 8: Logo Travis CI

4.1.3 Bamboo

Bamboo [11] est un outil d'intégration continue qui va permettre de construire du code, le packager et le livrer sur des serveurs internes. Bamboo va donner aux développeurs la possibilité de rassembler un ensemble de builds automatisés, de tests et livraisons dans un workflow unique.

Figure 9: Logo Bamboo

4.1.4 GitLab CI

GitLab CI/CD [13] permet d'automatiser les builds, les tests, les déploiements, etc. des applications. L'ensemble des tâches peuvent être divisé en étapes et l'ensemble des tâches et étapes constituent un pipeline. GitLab dispose d'un pipeline de CI-CD qui permet à un développeur de configurer un script automatisé pour les tests en continu, les demandes de pull, les builds, les différents environnements de test et bien plus encore.

Figure 10: Logo Gitlab

4.1.5 TeamCity

TeamCity [14] est une solution de CI/CD à usage général qui permet la plus grande flexibilité pour toutes sortes de workflows et de pratiques de développement. TeamCity teste les modifications apportées au code avant même qu'elles ne soient insérées dans la mainline. Le code source est uniquement intégré au code base pour toute l'équipe lorsqu'il est exempt d'erreurs. TeamCity effectue les tests de façon autonome en arrière-plan de telle sorte que les développeurs peuvent poursuivre leur travail dans l'intervalle.

Figure 11: Logo Teamcity

À l'aide du tableau récapitulatif ci-dessous, nous pouvons identifier en un coup d'œil les programmes susceptibles de nous convenir. Nous pouvons aussi voire directement si le service supporte également la livraison continue ou propose un hébergement sur le cloud.

Tableau 2: Tableau récapitulatif de quelques outils d'intégration continue source (https ://www.ionos.fr/digitalguide/sites-internet/developpement-web/outils-dintegration-continue/)

	Déploiement continu supporté	Hébergement sur le Cloud	Licence	Prix pour l'offre payante	Version gratuite	Particularité
Jenkins	✓	✓	MIT	-	✓	Nombreux plugins
Travis CI	✗	✓	MIT	69–489 $ par mois	✓	Connexion directe à GitHub
Bamboo	✓	✓	Propriétaire	Coût unique de 10-110 000 $	✓	-
GitLab CI	✓	✓	MIT/EE	4-99 $ par mois	✓	Connexion directe avec d'autres produits Atlassian

Circle CI	✓	✓	Propriétaire	50- 3 150 $ par mois	✓	Utilisation simple
CruiseControl	✗	✗	BSD	-	✓	Entièrement gratuit
Codeship	✓	✓	Propriétaire	75- 1 500 $ par mois	✓	Version de base et pro
TeamCity	✓	✗	Propriétaire	Coût unique de 299- 21 999 €	✓	Gated Commits

L'image suivante présente l'usage des serveurs d'intégrations continues.

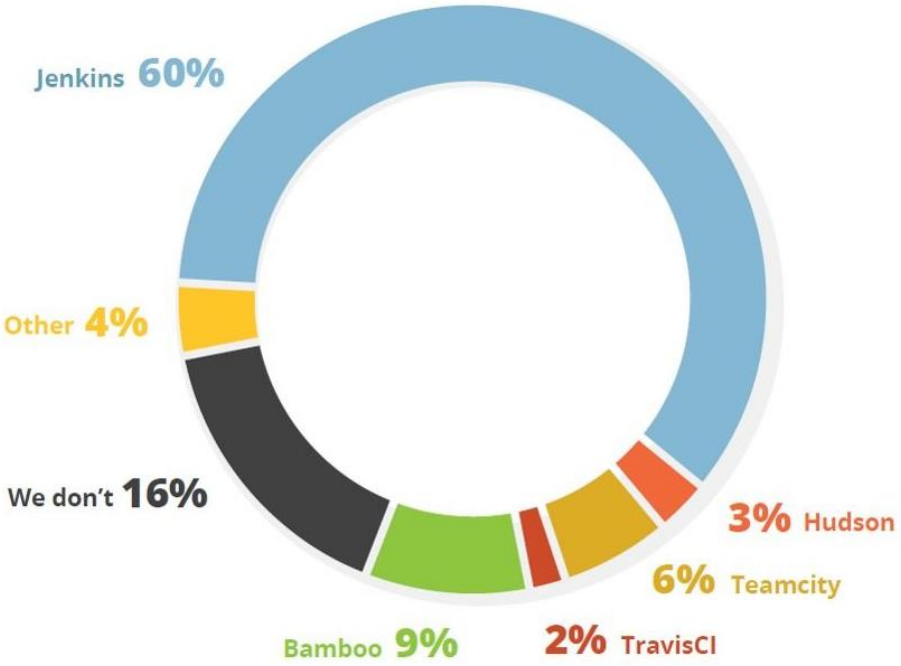


Figure 12: Usage de serveur d'intégration continue, 2018 [17].

Le tableau 2 si dessus et la figure précédente nous oriente clairement à choisir l'outil Jenkins, car en plus de posséder une multitude de plugins, elle est open source et existent en version essentiellement gratuite.

4.2 Présentation de l'outil choisit

4.2.1 Historique

Jenkins, qui s'appelais à l'origine Hadson est un outil d'intégration continue open source écrit en java. En 2009 Oracle Sun acheta et hérita de la base de code de Hadson. En 2011, les tensions existantes entre Oracle et la communauté open source ont crée un point de rupture et le projet se scinda en deux : d'une part Jenkins conduit par la plupart des développeurs Hadson qui étaient favorable à l'open source et d'autre part, Hadson, qui reste sous le contrôle d'Oracle.

Avec Jenkins, les entreprises peuvent accélérer le processus de développement logiciel grâce à l'automatisation. Jenkins intègre le processus de cycle de vie de développement de toutes sortes, y compris la construction, le document, les tests, le package, l'étape, le déploiement, l'analyse statique et bien d'autre.

4.2.2 Fonctionnalités

Jenkins réalise l'intégration continue à l'aide des plugins permettant d'intégrer les différentes étapes DevOps. Possédant plus d'un million d'utilisateurs dans le monde avec environs 147 000 installations actives, Jenkins est interconnecté avec plus de 1000 plugins qui lui permettent de s'intégrer dans la plupart des outils de développements, de tests et de déploiements. L'image suivante montre l'intégration de Jenkins avec différents outils du DevOps.

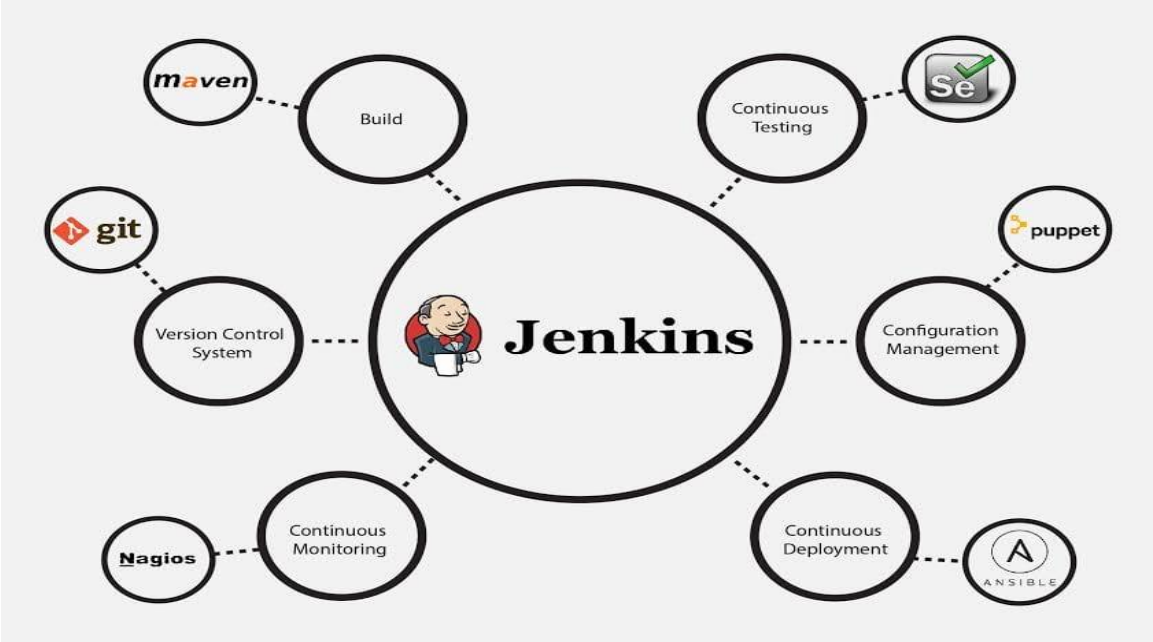


Figure 13: Intégration de Jenkins avec différents outils du DevOps (source : https ://www.edureka.co/blog/what-is-jenkins/)

4.2.3 Architecture distribuée de Jenkins

Jenkins utilise une architecture maitre-esclave pour gérer les versions distribuées. Dans cette architecture, maitre et esclave communiquent via le protocole TCP/IP [16].

Maître Jenkins : Le travail du Master est de gérer :

- Planification des travaux de construction.
- Distribution des builds aux esclaves pour l'exécution réelle.
- Surveillez les esclaves (éventuellement en les mettant en ligne et hors ligne selon les besoins).
- Enregistrement et présentation des résultats de construction.
- Une instance maître de Jenkins peut également exécuter des tâches de génération directement.

Esclave Jenkins : Un esclave est un exécutable Java qui s'exécute sur une machine distante. Voici les caractéristiques des esclaves Jenkins :

- ✓ Il entend les demandes de l'instance Jenkins Master.
- ✓ Les esclaves peuvent fonctionner sur une variété de systèmes d'exploitation.
- ✓ Le travail d'un esclave est de faire ce qu'on lui dit, ce qui implique d'exécuter des tâches de construction envoyées par le maître.
- ✓ Vous pouvez configurer un projet pour qu'il s'exécute toujours sur une machine esclave particulière, ou un type particulier de machine esclave, ou simplement laisser Jenkins choisir le prochain esclave disponible.

Le diagramme ci-dessous est explicite. Il se compose d'un maître Jenkins qui gère trois esclaves Jenkins.

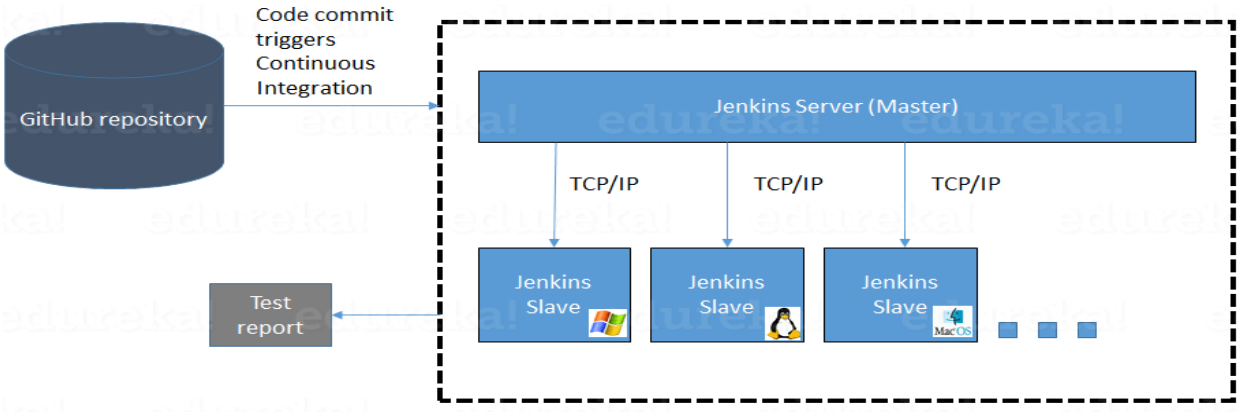


Figure 14: Exemple d'utilisation de Jenkins dans un environnement distribué.

4.2.4 Pipeline Jenkins

Nous ne pouvons terminer sans parler du pipeline Jenkins qui est utilisé pour savoir qu'elle tache Jenkins est en cours d'exécution.

Souvent, plusieurs modifications différentes sont apportées par plusieurs développeurs à la fois, il est donc utile de savoir quelle modification est testée ou quelle modification est dans la file d'attente ou quelle version est interrompue. C'est là que le pipeline entre en scène. Le pipeline Jenkins vous donne un aperçu de la progression des tests. Dans le pipeline de construction, la construction dans son ensemble est divisée en sections, telles que les phases de test unitaire, de test d'acceptation, de packaging, de reporting et de déploiement. Les phases du pipeline peuvent être exécutées en série ou en parallèle, et si une phase réussit, elle passe automatiquement à la

phase suivante (d'où la pertinence du nom « pipeline »). L'image ci-dessous montre la structure d'un pipeline de construction.

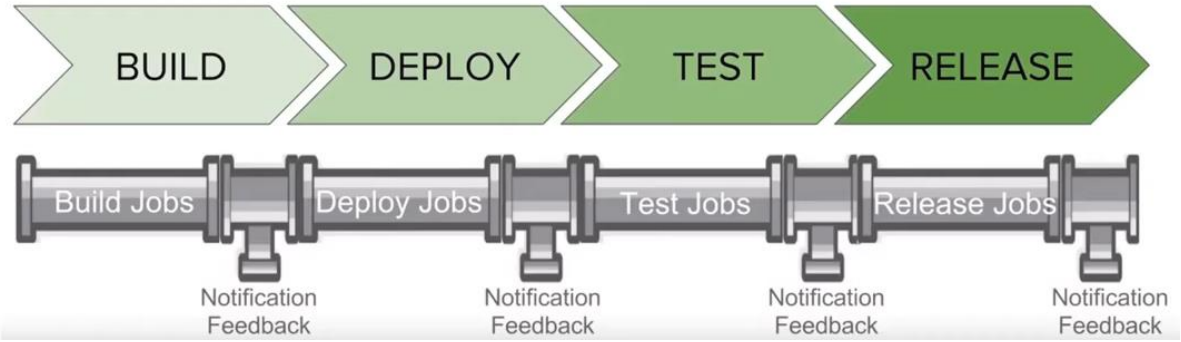
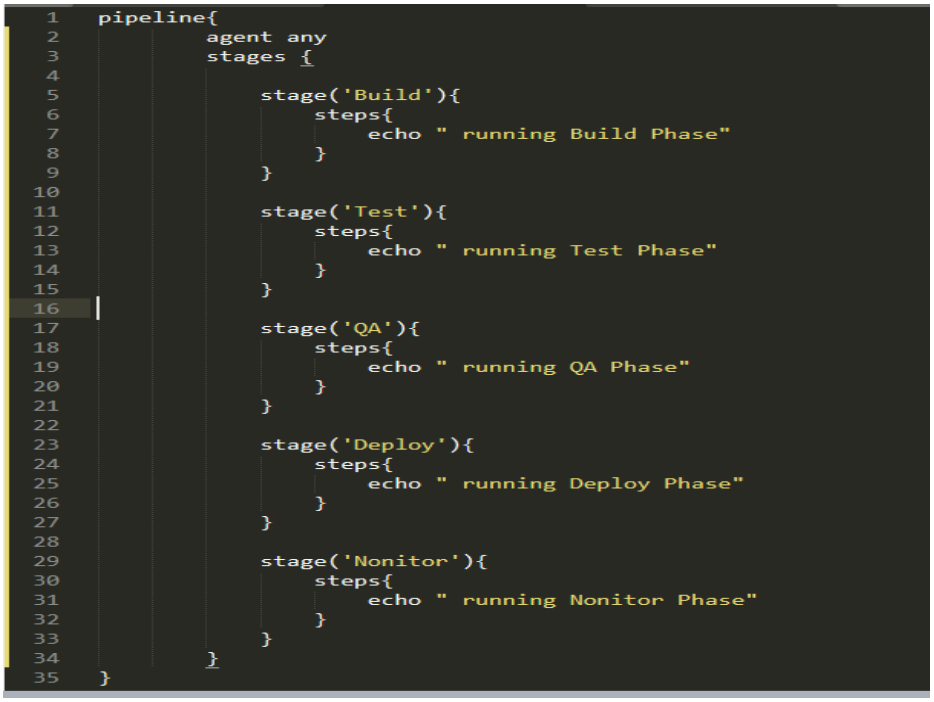


Figure 15: Structure d'un pipeline

Le pipeline repose sur le langage de script Groovy écrit dans des fichiers appelées **Jenkinsfile**. Jenkinsfile est un fichier qui stocke le pipeline sous forme de code. Les scripts pipelines apportent les notions suivantes :

- Un pipeline est l'ensemble du processus à exécuter ;
- Un nœud (node) représente un environnement pouvant exécuter un pipeline (une machine esclave) ;
- Un niveau (stage) représente un ensemble d'étapes de votre processus (par exemple, la récupération des sources, la compilation…) ;
- Une étape (step) représente ce qu'il y a à faire à un moment donné (l'action à proprement parler, comme make).

L'image suivante présente la structure d'un Jenkinsfile.



```
pipeline{
        agent any
        stages {

            stage('Build'){
                steps{
                    echo " running Build Phase"
                }
            }

            stage('Test'){
                steps{
                    echo " running Test Phase"
                }
            }

            stage('QA'){
                steps{
                    echo " running QA Phase"
                }
            }

            stage('Deploy'){
                steps{
                    echo " running Deploy Phase"
                }
            }

            stage('Nonitor'){
                steps{
                    echo " running Nonitor Phase"
                }
            }
        }
}
```

Figure 16: Structure d'un Jenkinsfile

CHAPITRE 5 : IMPLEMENTATION

5.1 Test

PlexusERP tourne sur un environnement linux avec les configurations minimales suivante :

- Processeur core I7
- Ram 12Go minimale (16 Go Recommandée)

L'installation de Jenkins [18] est très simple. Il suffit d'ajouter la clé du référentiel au système avec la commande suivante :

```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

Ensuite on ajoute la source du référentiel dans la source.list de Debian.

```
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

Puis on fait une mise à jour et on installe les dépendances de Jenkins comme suit :

```
$ apt update
```

```
$ sudo apt install jenkins
```

. Activer par la suite les services de Jenkins et entrée l'adresse suivante pour lancer Jenkins http ://127.0.0.1 :8080/. L'interface d'accueil après avoir entré toutes les configurations est présentée comme suit :

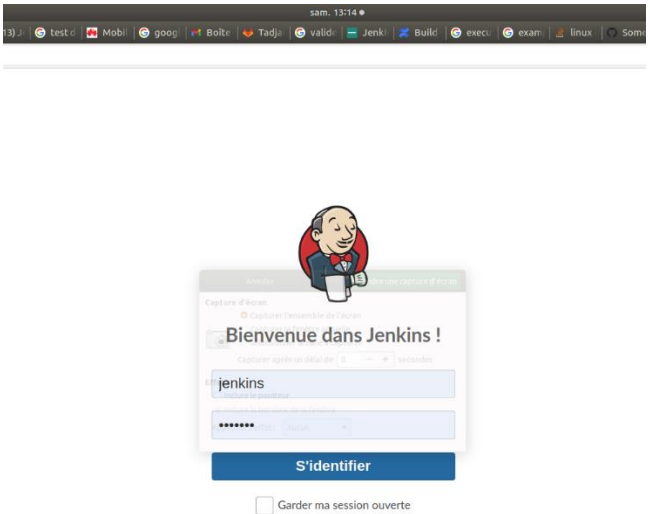


Figure 17: Interface de connexion de Jenkins

Une fois connecté, nous accédons à la page d'accueil de Jenkins ou on peut configurer nos premiers jobs.

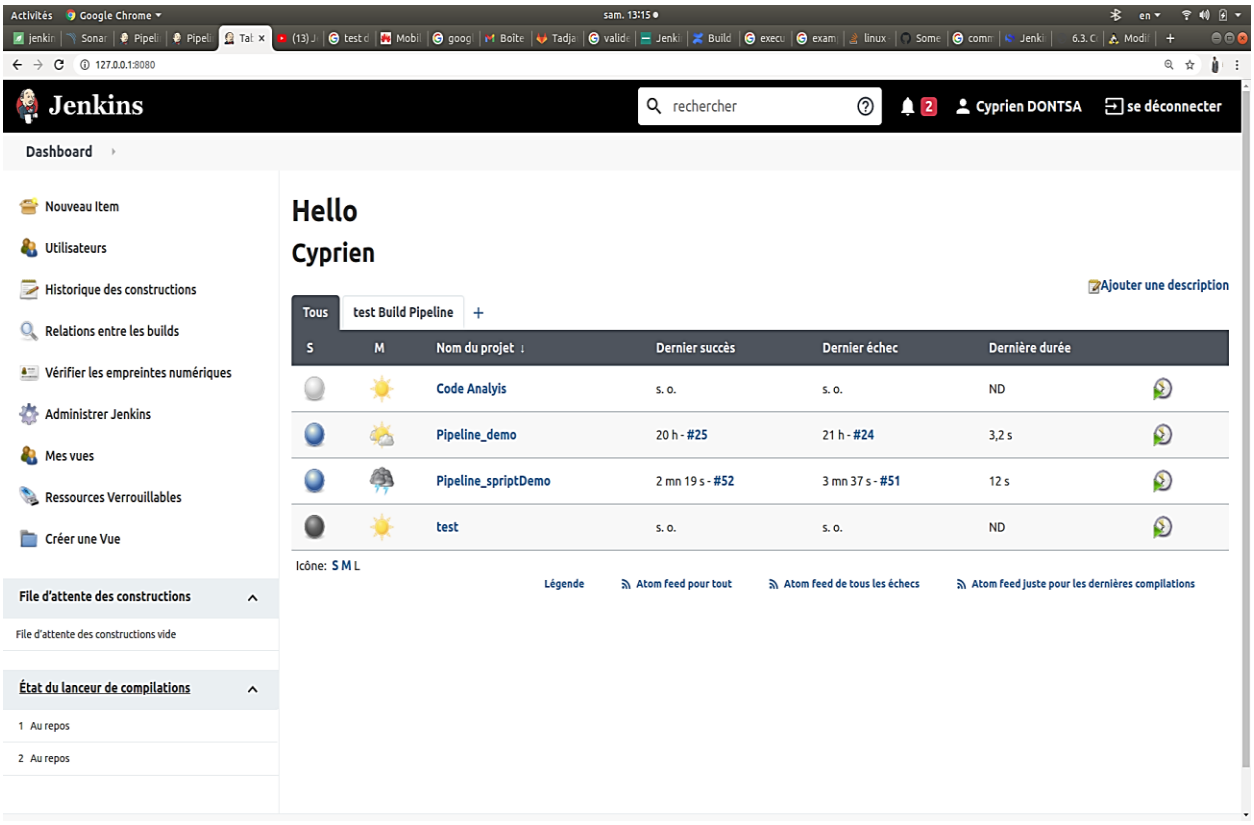


Figure 18: Page d'accueil Jenkins

Nous avons écrit notre propre pipeline pour la compilation du projet plexusERP. Pour le faire nous avons écrit un script shell contenant toutes les commandes que nous voulions automatiser pour l'exécution de build avant de l'insérer dans notre Pipeline.

Le contenu de ce script est présenté comme l'illustre la figure suivante :

```
cd /home/hadron/asset-manager/ast_mgr_env/ast_mgr
pwd
echo "Activation de la variable denvironnement "
source ../venv/bin/activate
pwd
echo "Starting building "

pwd
pip install -U pip
echo " je suis ici"
echo " **********************************************************************"
echo " **********************************************************************"
echo " **********************************************************************"
pwd
yarn install
pip install -r requirements_py3.txt
source ../venv/bin/activate
fab install-services
fab create-postgres-db
fab prepare-db
fab restart-app
fab start-nginx
fab fd:features=common
fab status


exit 0
```

Figure 19: Script d'exécution de compilation

5.2 Résultats et commentaires

Comme résultat obtenu nous pouvons voir sur les images suivantes l'état d'exécution, les tendances des différentes exécutions et le temps d'exécution nécessaire pour chaque étape du pipeline.

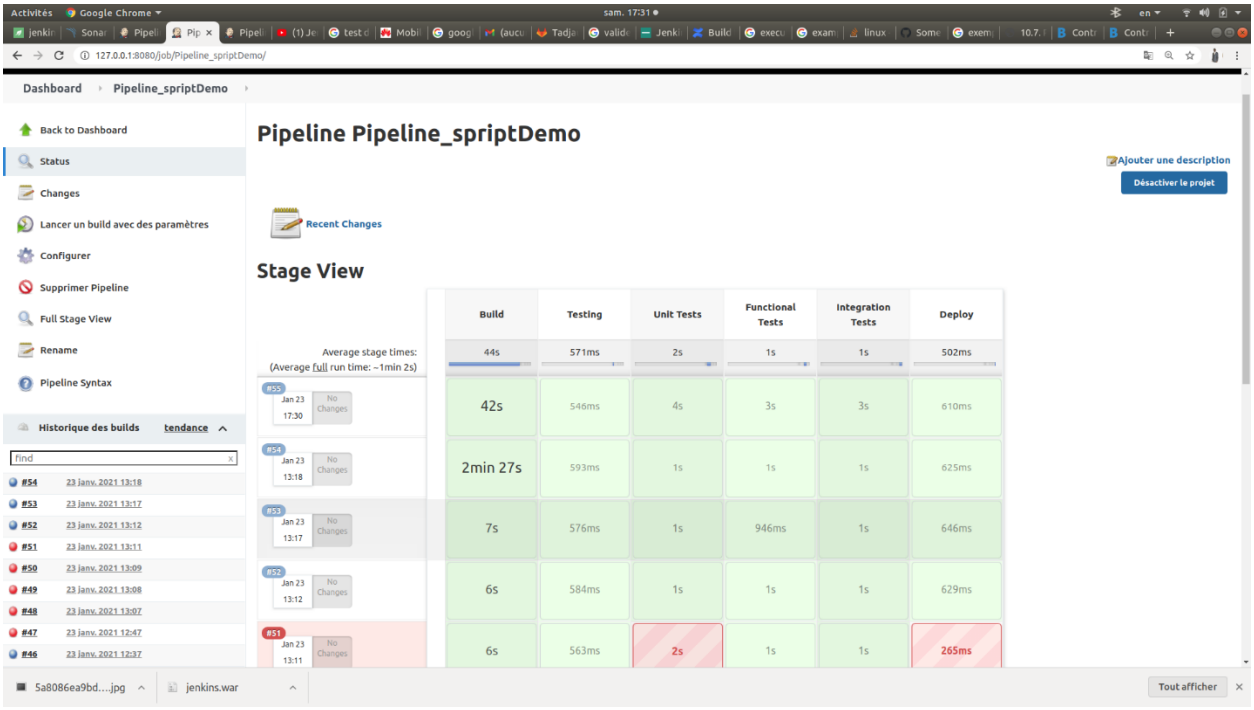


Figure 20: Etat de l'exécution du pipeline

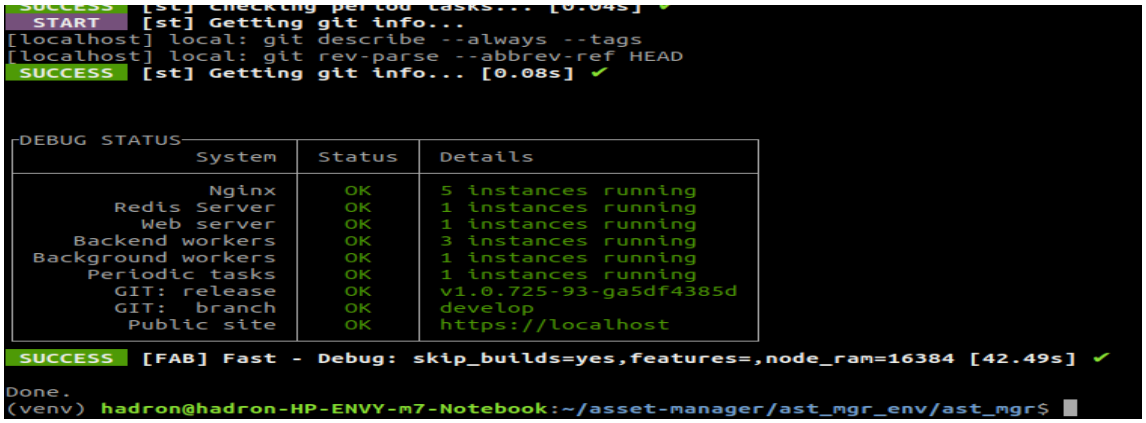


Figure 21: Etat d'exécution des services de l'application plexusERP

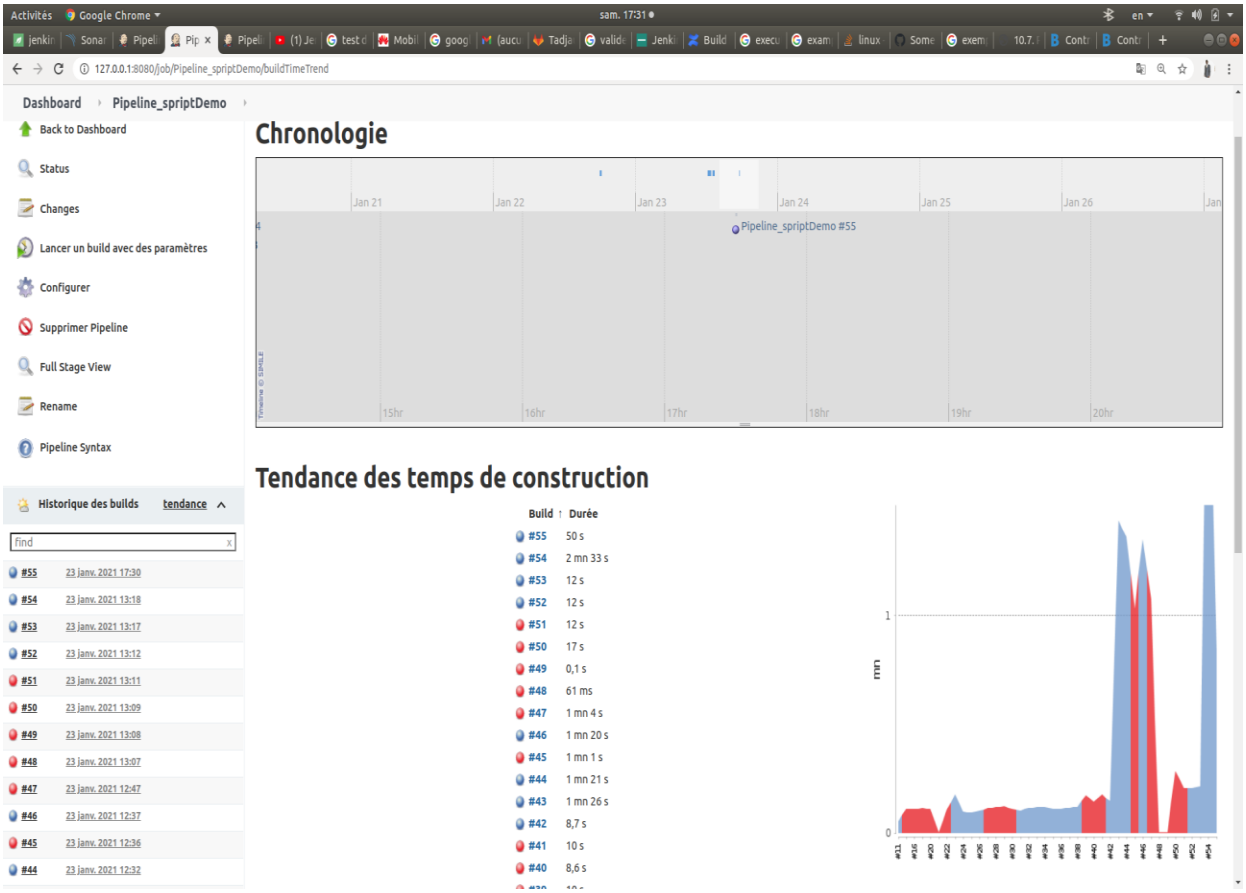


Figure 22: Tendances et chronologie d'exécution du pipeline

5.3 Perspectives

Il ressort de ses résultats que la compilation du projet en utilisant le serveur d'IC Jenkins marche bien. Cependant nous recommandons à Hadron S.A pour des améliorations future et le l'utilisation efficace de Jenkins de :

- Mettre en place outils d'analyse des codes tel que sonarCube
- Améliorer le script d'exécution du pipeline en effectuant des multiples tests
- Utiliser selenium pour effectuer les tests avant le déploiement
- Couplé à Jenkins l'outil Ansible pour effectuer le déploiement continue de l'application.

CONCLUSION

Ce projet nous a permis d'approfondir nos capacités en administration système et des bonnes compétences en ce qui concerne l'intégration continue et la culture DevOps. La qualité technique et le professionnalisme sont des acquis que nous avons pu bénéficier chez Hadron S.A. il était que question pour moi de mettre sur pied un serveur d'intégration continue fonctionnel qui permettra à l'entreprise automatiser leur workflow.

Pour mener à bien cet objectif, j'ai ressorti la problématique, puis j'ai mis l'accent sur la notion d'intégration continue et des différents outils existants ; et enfin nous avons installer et configurer Jenkins pour assurer l'ensemble des processus liés à l'intégration continue. J'ai également travaillé sur un composant personnalisé de more bouton à intégrer sur le projet plexusERP.

Cependant beaucoup reste à faire pour avoir un système complet renferment tous les composant nécessaire pour assurer le bon fonctionnement de l'IC chez Hadron S.A. nous avons suggéré pour des améliorations futures d'implémenter un ensemble d'outils de la culture DevOps qui fonctionneront en parallèle avec Jenkins.

REFERENCES BIBLIOGRAPHIQUE

[1] Thomas Ropars, DevOps Intégration Continue, 2020, thomas.ropars@univ-grenoble-alpes.fr 2020

[2] Combell, Qu'est-ce que DevOps et pourquoi ce mouvement est-il si important ? 10 juillet 2017 ; https://www.combell.com/fr/blog/devops-pourquoi-si-important/ consulté le 10 Novembre 2020

[3] David Galiana, Qu'est-ce que la méthodologie Agile ? 06 juillet 2017, https://www.planzone.fr/blog/quest-ce-que-la-methodologie-agile consulté le 10 Novembre 2020

[4] TechTrends, DEVOPS, 1ere édition, Publication de Xebia IT Architects, pages 5-6

[5] https://jenkins-le-guide-complet.github.io/html/book.html

[6] Fabian Piau, intégration continue, juillet 2009

[7] Steffen Gebert (@StGebert), DevOps Meetup Frankfurt, 20.10.2016 https://fr.slideshare.net/StephenKing/jenkins-pipelines-67473540

[8] Homada Boumedane. Conception et implémentation d'une infrastructure de déploiement continu pour le logiciel SPHER. Génie logiciel [cs.SE]. 2015. ffdumas-01655738, disponible sur https://dumas.ccsd.cnrs.fr/dumas-01655738/document , page 38.

[9] http://igm.univ-mlv.fr/~dr/XPOSE2012/Integration%20Continue/concept.html consulté le 18 Décembre 2020

[10] Site officiel jenkins, disponible sur https://www.jenkins.io/doc/book/ consulté le 02 janvier 2021

[11] https://www.atlassian.com/fr/software/bamboo consulté le 02 janvier 2021

[12] https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443872-travis-ci-logiciel-libre-d-integration-continu-alternative-a-jenkins/ consulté le 02 janvier 2021

[13] https://openclassrooms.com/fr/courses/5641721-utilisez-git-et-github-pour-vos-projets-de-developpement/6113136-utilisez-le-gitlab-integration-continue-ic#:~:text=GitLab%20est%20un%20service%20qui,test%20et%20bien%20plus%20encore%20! consulté le 02 janvier 2021

[14] Site officiel jetBrains https://www.jetbrains.com/fr-fr/teamcity/ consulté le 02 janvier 2021

[15] Saurabh, Jenkins Tutorial | Continuous Integration Using Jenkins | Edureka, Last updated on May 22,2019 https://www.edureka.co/blog/what-is-jenkins/ consulté 04 Janvier 2020

[16] Saurabh, What is Jenkins? | Jenkins For Continuous Integration | Edureka last update on Sep 2020 https://www.edureka.co/blog/jenkins-tutorial/ consulté 04 Janvier 2020

[17] https://jch.blog4ever.com/contruire-un-pipeline-devops

[18] https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04-fr