Département de génie logiciel

Projet de session :

Mise en place d'un processus d'intégration continue à l'aide de Bamboo

Réalisé par :
Audrey EUGENE
Andres Piraquive
Junior Adolphe
Serge DOGNY

MGL805 - Automne 2019 Montréal, Le
25 OCTOBRE 2019

Avant-propos

Ce document a été rédigé dans le cadre du projet de session du cours MGL805 suivi à l'ÉTS à la session d'automne 2019. Son objectif est d'expliquer la mettre en place d'un processus d'intégration continue à l'aide de Bamboo.

Le rapport doit également, à l'aide d'un exemple concret, décrire pas à pas, comment installer, configurer et effectuer l'intégration continue afin qu'il puisse servir de guide pour une organisation qui aimerait débuter dans cette activité.

Table des matières

Liste des figures

Liste des abréviations, sigles et acronymes

CL: Langage de contrôle
CMD : Command Prompt
EC2: Elastic Cloud Compute
ÉTS: École de Technologie Supérieur
Gib : Giga Binary Byte
IC: Intégration Continue
JMS: Java Message TSC Service
PHP : Personal Home Page
RPG: Report Program Generator
RSS : Rich Site Summary
VCPUS: Virtual central processing unit
YAML : Yet Another Multicolumn Layout

1. Introduction

1.1. Objectif du projet.

Ce projet est réalisé dans le cadre du cours MGL805 à l'ÉTS à la session d'automne 2019. Son but est d'expliquer en détail, comment mettre en place un processus d'intégration continue à l'aide de Bamboo. Enseigner ce qu'est l'intégration continue et ses avantages, et pas-à-pas comment installer, configurer et effectuer l'intégration continue, à l'aide d'un exemple concret, pour qu'une personne d'une organisation qui aimerait débuter cette activité puisse suivre les enseignements.

1.2. Qu'est-ce que l'intégration continue ?

L'intégration continue (IC) est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier que le résultat de chaque modification au code source ne produit pas de régression (c.-à-d. défaut) dans l'application développée [1]. En d'autres termes, l'intégration continue sert à vérifier que l'ajout ou la modification de lignes de code n'a pas introduit d'erreurs ou de problèmes (c.-à-d. du code non conforme) dans le code existant.

La principale différence avec l'intégration classique est que, l'intégration n'a lieu que lorsque toutes les fonctionnalités sont développées et testées. Cette approche entraîne des délais, des coûts supplémentaires et parfois même des logiciels livrés avec d'importantes défaillances qui n'ont pas été corrigées par manque de temps ou de budget. Autrement dit, plus l'intégration est fréquente, moins elle est longue.

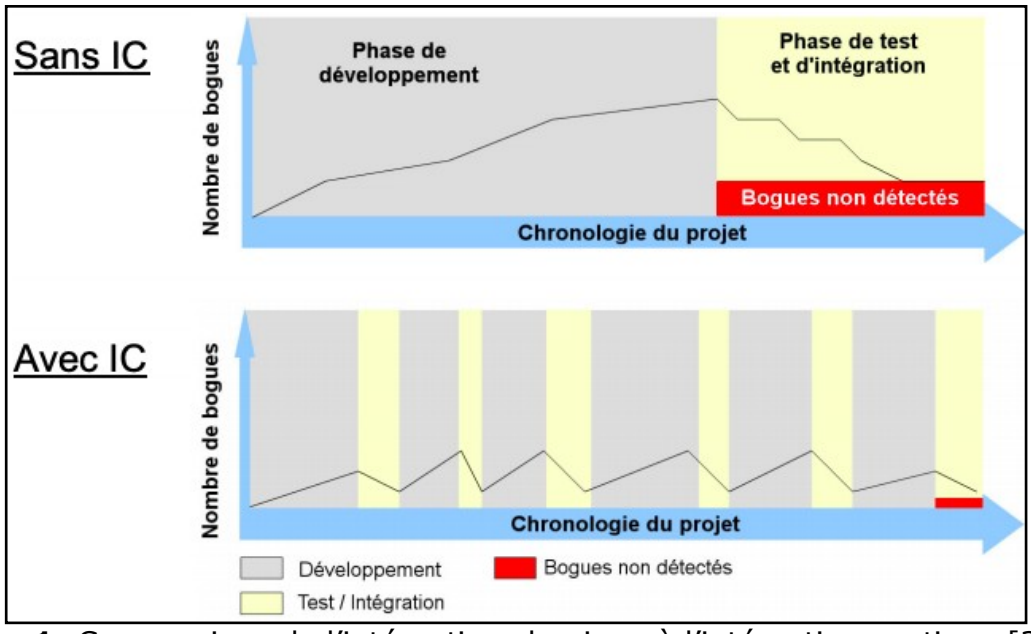


Figure 1: Comparaison de l'intégration classique à l'intégration continue [2]

1.3. Pourquoi l'intégration continue ?

En génie logiciel, plus une erreur est détectée tôt dans le processus de développement, moins elle est coûteuse. L'intégration continue permet de détecter les défauts le plus tôt possible. Elle permet de garantir un code de qualité en production, et donc une meilleure satisfaction des utilisateurs finaux. L'exécution des tests automatiquement de tout le code source lors de chaque ajout/modification de fonctionnalités permet d'éviter l'introduction de régressions en production [4].

Certaines tâches récurrentes (c.-à-d. compilation, tests unitaires et tests d'intégration) qui sont souvent sources d'erreurs humaines sont automatisables. Ainsi les développeurs peuvent simplement configurer des notifications du serveur d'intégration pour être prévenus sur le service de leur choix en cas d'anomalies, gagnant ainsi un temps précieux tout en évitant de surveiller constamment les erreurs humaines générées à ces tâches répétitives.

1.4. Fonctionnement général de l'intégration continue

L'objectif est donc que tout changement du code soit immédiatement vérifié par une tâche automatique qui lance un test d'intégration. En effet, Un serveur d'intégration continue va se synchroniser avec un dépôt de code source. À intervalle régulier, il va vérifier que le dépôt est à jour (c.-à-d. sans défaut). Si une mise à jour est intervenue, il va effectuer une série de tâches (compilation par exemple). En fonction du résultat des tâches (c.-à-d. succès ou échec), le serveur va indiquer l'état du projet et possiblement transmettre des alertes comme l'indique la figure 2.

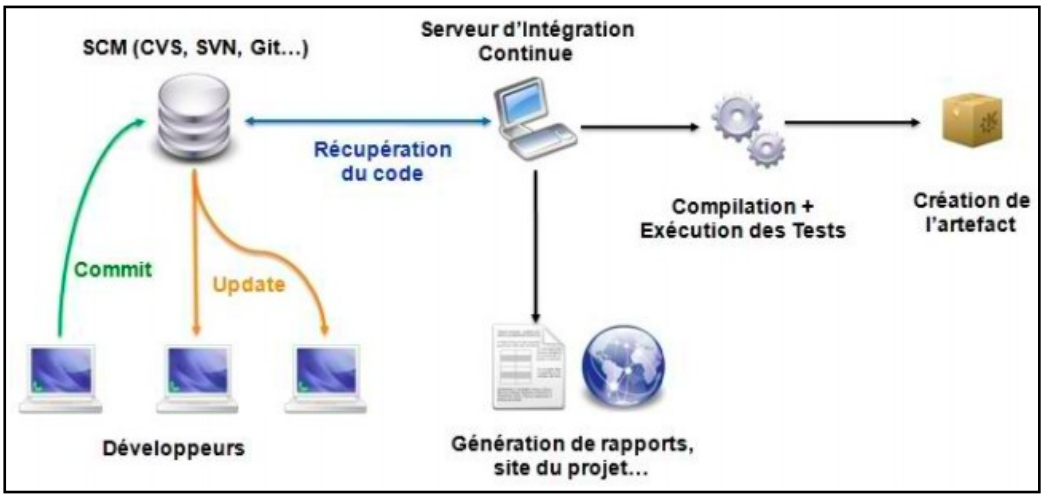


Figure 2: Fonctionnement de l'intégration continue [3]

En pratique, un processus d'intégration continue typique suit le continuum d'étapes suivantes [5]:

- Chargement de la dernière version du projet du gestionnaire de version;
- Compilation;
- Exécution des tests unitaires;
- Inspection de la qualité du code (vérifier des métriques de qualité)
- Génération de documentation, de rapports, de notes de release (par exemple la Javadoc, rapport Checkstyle);

- Construction de version (« *release*»);
- Déploiement de l'application sur l'environnement de test d'intégration;
- Exécution des tests d'intégration.

1.5. Principaux outils d'intégration continue

Plusieurs outils permettent de réaliser l'intégration continue de manière automatique. Voici une liste non exhaustive :

Jenkins

Jenkins est un logiciel d'intégration continue gratuit (c.-à-d. logiciel libre) développé en Java. Il fonctionne dans un conteneur Web ou en mode autonome avec son propre serveur Web embarqué. Jenkins fédère une vaste communauté qui propose régulièrement de nouvelles améliorations du service. Il est assez simple à installer et plus de 1000 plug-ins sont disponibles, permettant ainsi de l'intégrer à la plupart des outils de développement, de test et de déploiement populaires [6].

Travis CI

Travis CI est un logiciel libre d'intégration continue publiée sous licence MIT. Il permet de compiler, tester et déployer le code source des logiciels développés, notamment en lien avec le service d'hébergement du code source GitHub. Sa configuration se fait à l'aide du langage YAML. Travis permet aussi de lancer des tests dans des environnements Docker [7].

Gitlab CI

Gitlab CI est à Gitlab ce que Travis est à Github : un outil intégré de bout en bout.

Bamboo

Bamboo est un logiciel propriétaire d'intégration continue développé par Atlassian. Il est payant et son coût dépend du nombre d'agents distants dont vous avez besoin [8].

TeamCity

TeamCity est un serveur d'intégration continue basé sur la plateforme Java, développé par l'éditeur JetBrains. Il est accessible gratuitement et possède une interface Web épurée qui le rends est assez facile à configurer et à utiliser [10].

Bien que tous ces outils proposent des fonctionnalités assez similaires, plusieurs considérations entrent en jeu au moment de faire un choix. Ce sont entre autres, les connaissances de l'équipe (c.-à-d. avec quel outil l'équipe est-elle le plus familière ?), le coût, la facilité d'intégration aux autres outils déjà en place (par exemple JIRA, Bitbucket...).

Dans le cadre de ce projet, tel que libellé dans l'énoncé, nous avons l'exigence de mettre en œuvre l'intégration continue à l'aide de Bamboo.

1.6. Principaux avantages à utiliser un outil d'intégration continue.

L'utilisation d'un outil d'intégration continue offre beaucoup d'avantages en termes de sécurité et de productivité, notamment :

Gain de temps : tandis qu'avant la présence d'intégration continue on pouvait mettre plusieurs semaines/mois à livrer un projet logiciel en production. Cela nécessitait la création/gestion de plusieurs environnements, des soucis de sécurité et qu'il fallait décrire cette procédure de plusieurs pages pour chaque projet). L'intégration continue réduit ce travail à quelques minutes et à quelques clics. Dans certains cas, la livraison prend seulement le temps de créer un tag.

Fiabilité : éviter les actions manuelles entre les différents environnements permet aussi de limiter les erreurs. On évite ainsi que ces problèmes viennent gâcher/ralentir la mise en production.

Simplicité : le code est mis dans une sorte de pipeline composé d'étapes différentes, mais bien rodées (c.-à-d. compilation, test…), et avec de simples clics, on peut simplement passer d'un environnement à l'autre.

Prise en main du projet facilitée : Une fois que le processus d'intégration continue est bien mis en place, tout le monde peut facilement exécuter des livraisons. C'est utile notamment lorsqu'un nouveau développeur arrive sur un projet et n'a pas à connaître les détails sous-jacents de ce processus.

2. Guide d'installation de Bamboo

2.1. Bamboo, qu'est-ce que c'est ?

« Bamboo est un outil d'intégration continue qui va permettre de construire du code, le « packager » et le livrer sur des serveurs internes »[14]. Ce n'est pas un logiciel libre et il est distribué par Atlassian, l'un des plus populaires fabricant d'outils de développement, par exemple : fabricant de JIRA et Confluence entre autres.

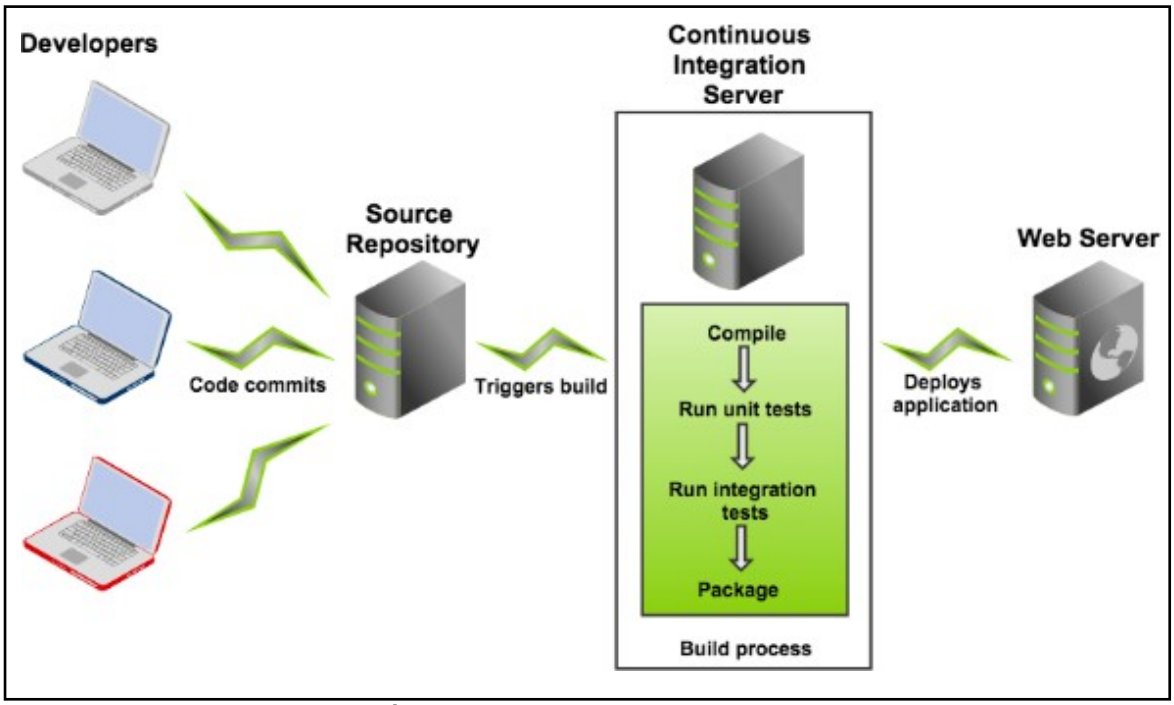


Figure 3 : Intégration continue avec Bamboo extrait de [13]

À chaque fois que du code est migré dans le référentiel de code source utilisé (par exemple, GitHub, GitLab, Bitbucket...), une étape de construction (« build »), des tests unitaires et des tests d'intégration sont effectués (ou déclenchés) afin de garantir que ces nouvelles modifications s'intègrent bien, sans défauts, dans la base de code existante. Les « builds » d'intégration fournissent rapidement un retour d'information rapide concernant l'échec ou le succès et la qualité de ces récents changements.

Le rôle de Bamboo est donc la planification et la coordination des travaux de construction et de tests de votre application. Vous devez déjà avoir la configuration suivante [23]:

- un référentiel de code contenant le code source complet du projet;
- des scripts de construction;
- des suites de tests.

Voici un exemple d'organisation de « builds » à l'aide une structure bien définie. Avec Bamboo, vous créez des plans de « build » (ou de déploiement), puis vous mettez en place des étapes (de l'anglais « Stages »), des activités (de l'anglais « Jobs ») et des tâches (de l'anglais « tasks »), flexibles et faciles à implémenter via l'interface utilisateur telle que présentée à la figure 4 [23].

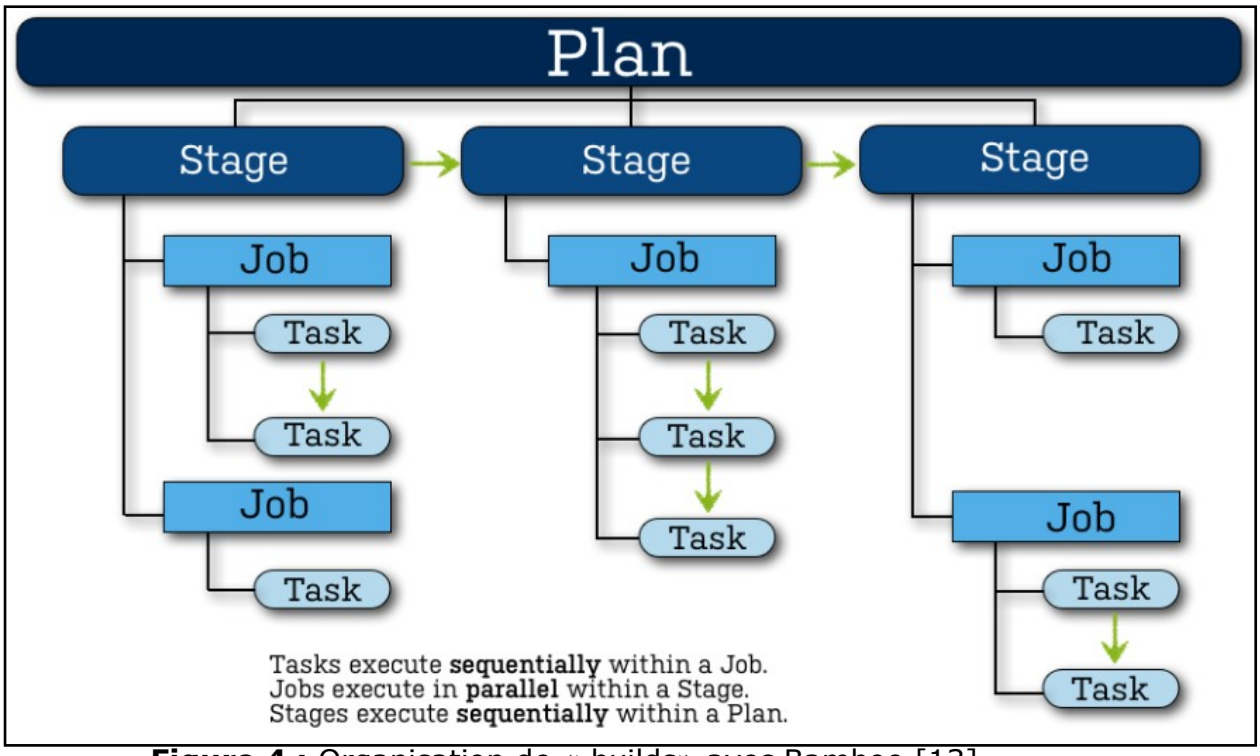


Figure 4 : Organisation de « builds» avec Bamboo [13]

Dans votre Plan, vous pouvez avoir plusieurs étapes. Selon la documentation du site Atlassian [23], chaque étape :

- a une seule activité par défaut, mais peut être utilisée pour grouper plusieurs activités;
- exécute ses activitées en parallèle, via plusieurs agents (si disponibles);
- doit avoir complété toutes ses activités avant de pouvoir démarrer l'étape suivante du plan;
- peut produire des artéfacts qui pourront être utilisés dans une étape ultérieure.

2.2. Pourquoi Bamboo ?

La Sogebank S.A, l'entreprise pour laquelle nous agissons en tant que consultants utilise déjà Bitbucket et Jira. Une des raisons du choix de Bamboo est de profiter de la facilité d'intégration qu'il offre avec les outils existants (c.-à-d. Bitbucket et Jira). En plus, la combinaison (Bamboo + Bitbucket + Jira) permettra de garantit la traçabilité complète de chaque action au sein d'une équipe de développement [14].

Bamboo est aussi le choix des équipes professionnelles concernant l'intégration continue, le déploiement et les livraisons par rapport à Jenkins (l'un des plus populaires outils d'intégration continue). La figure 5 ci-dessous montre les avantages de Bamboo comparativement à Jenkins.

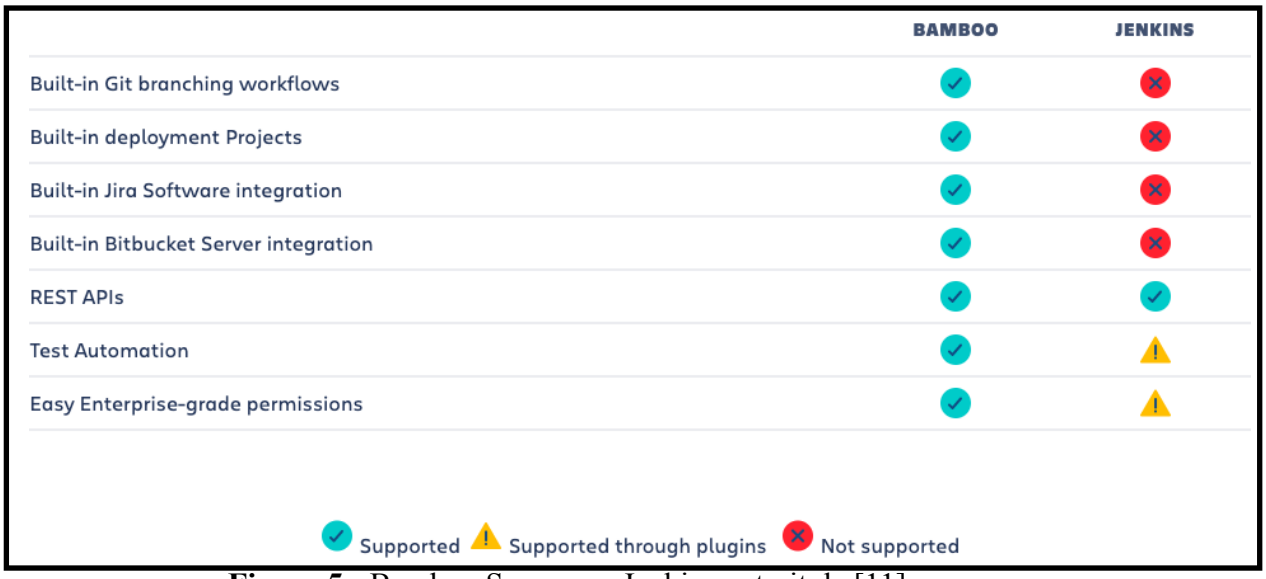


Figure 5 : Bamboo Server vs. Jenkins extrait de [11]

L'annexe A, explique comment installer Bamboo sur votre poste de travail (c.-à-d. le téléchargement, l'installation et la configuration).

2.3. Bamboo et les langages de programmation.

Dans ce document, nous avons présenté l'intégration continue pour un logiciel écrit en Java étant donné que, selon la documentation de Bamboo [16], c'est le langage avec lequel il est directement compatible, car c'est avec ce langage que Bamboo a été initialement créé. Néanmoins, nous savons qu'il est possible d'utiliser des projets logiciels utilisant d'autres langages en intégration continue avec Bamboo.

En effet, d'après les informations disponibles dans la documentation de Bamboo, il est expliqué et présenté comment le configurer afin d'utiliser l'intégration continue avec des projets de trois langages différents: PHP [17], Node.js [18] et .NET [19]. Quel que soit le langage utilisé, la stratégie initiale de démarrage consiste à configurer Bamboo et les outils du langage nécessaires. Une fois cela effectué tel que spécifié dans la documentation, le fonctionnement reste plus ou moins semblable à celui de l'exemple Java présenté ici. De plus, Bamboo propose deux techniques pour le lier à une base de données qui n'est initialement pas supportée [20]. Suite à d'autres recherches, nous avons appris qu'il était également possible de réaliser l'intégration continue de Bamboo avec le langage JavaScript [26] et qu'une méthode tout aussi simple que celle décrite pour les trois langages précédents est possible [27].

Cependant, il s'agit là de tout ce que nous avons pu trouver concernant les autres langages de programmation: il n'existe que très peu, voir aucune information concernant la configuration avec d'autres langages utilisés par la Sogebank tels que Windev/Webdev, RPGet CL.

3. Exemple d'intégration continue avec Bamboo

Afin de tenter de reproduire le plus fidèlement l'environnement de la Sogebank, nous avons créé une machine virtuelle (4 vcpus, 16 GiB memory) avec une image de Windows (Windows Server 2016 Datacenter). Voir Annexe B pour la création de cette VM.

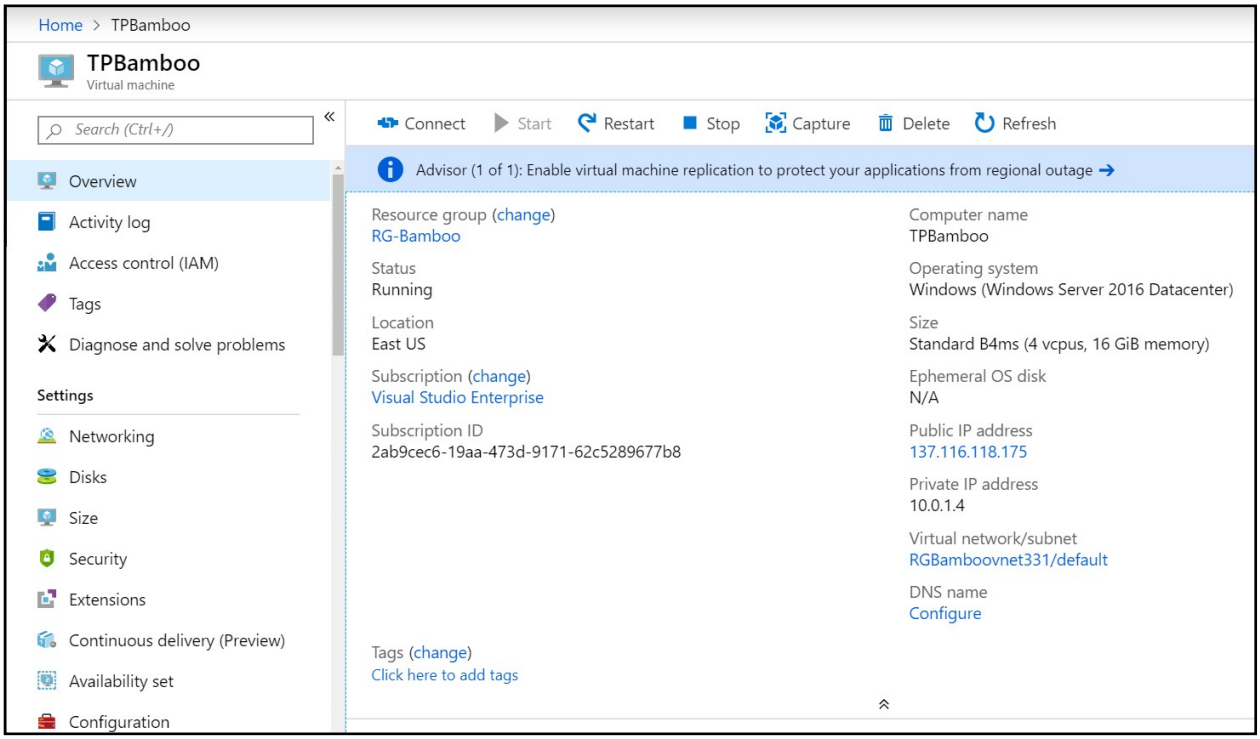


Figure 6 : Machine Virtuelle dans le Portail Azure

Nous avons ensuite installé sur cette machine virtuelle les outils suivants pour la mise en œuvre de l'intégration continue:

- Java 1.7.0_80 (voir Annexe C);
- Bamboo (voir l'annexe A pour l'installation de Bamboo); et
- Bitbucket le gestionnaire de code source.

Une fois l'installation de Bamboo terminée, exécutez la ligne de commande dans la fenêtre CMD (Command Prompt) afin de démarrer le serveur Bamboo, **bin\start-bamboo.bat** (Windows). Valider qu'il n'y a pas d'erreurs dans la console et dans la fenêtre Tomcat qui s'exécute à l'écran, voir **Annexe B**.

Une fois Bamboo démarré, pour accéder au Dashboard, insérez dans une fenêtre de fureteur Web l'URL: **http://localhost:8085** et vous allez être dirigé vers la page principale pour configurer le « Build plan »[15]. La figure suivante présente le « Dashboard » de Bamboo.

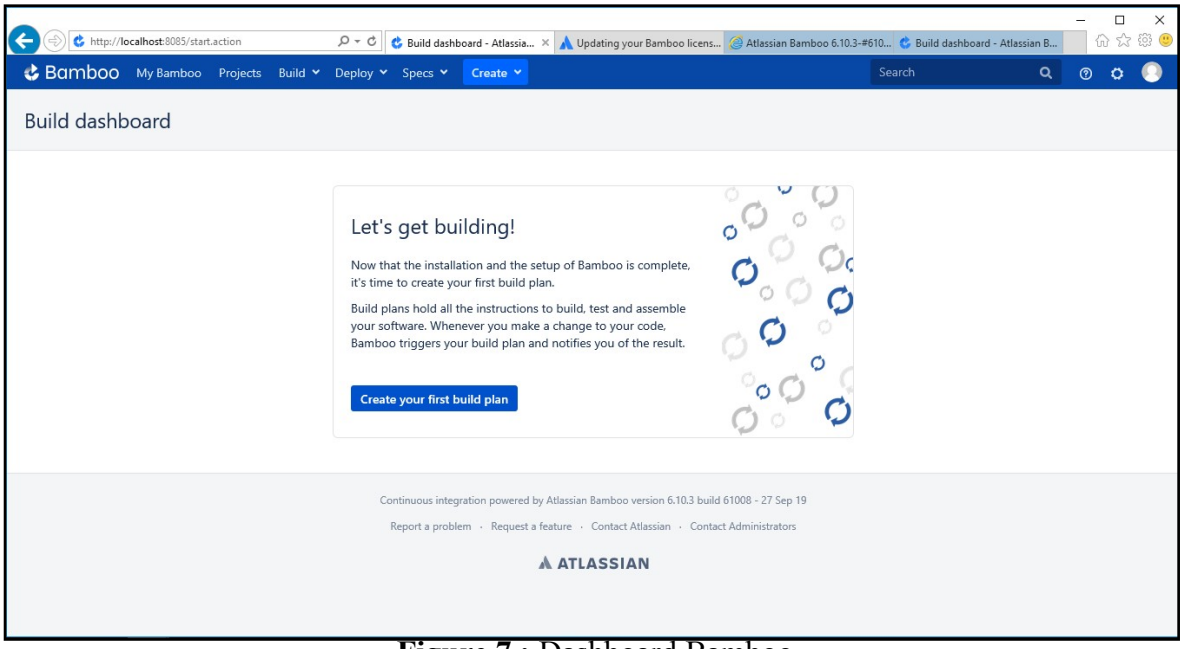


Figure 7 : Dashboard Bamboo

Pour la suite de cet exemple, nous allons mettre en place une organisation de « builds » à travers la structure de Bamboo. Plus précisément, nous allons créer un plan de « build », puis mettre en place un « Stage », des « Job » et des « tasks ».

3.1. Création et installation d'un agent

Un agent Bamboo est un service qui offre des fonctionnalités permettant d'exécuter des travaux (tâches). Les 3 types d'agents Bamboo suivants peuvent être configurés:

1 - **Les agents locaux** s'exécutent sur le serveur Bamboo, dans le processus du serveur, c'est-à-dire, dans la même machine virtuelle que le serveur Bamboo [21];
2 - **Les agents distants** s'exécutent sur des ordinateurs, autres que le serveur Bamboo. Ils s'exécutent dans son propre processus, c'est-à-dire qu'il possède sa propre machine virtuelle JAVA et utilise le protocole JMS (Java Message TSC Service) pour communiquer avec le serveur Bamboo [22];
3- **Les agents élastiques** s'exécutent dans Amazon Elastic Compute Cloud (EC2).

Pour plus d'information sur les agents élastiques, consultez la référence [23]. Notez qu'un agent local, avec le nom par défaut « Agent par défaut », est automatiquement créé après l'installation de Bamboo.

Voir l'Annexe E, pour les procédures de création de chacun de ces agents.

Recommandations :

Comme nous l'avons précisé, il existe plusieurs types d'agents dans la configuration de Bamboo. En fonction, de ce que vous voulez faire, vous pouvez décider d'utiliser un type d'agent ou un autre. [25]

Dans le cadre de ce projet, nous avons décidé d'utiliser des agents locaux parce que l'objectif principal, c'était de montrer comment faire de l'intégration continue en utilisant Bamboo. L'autre raison, notre licence d'essai ne nous autorise pas l'ajout d'agents distants, et les agents locaux correspondent largement à notre besoin. Mais dans une équipe de développement, comme celle de la Sogebank S.A, vu le nombre de développeurs nous recommandons de considérer l'utilisation d'agents distants pour les raisons suivantes : L'utilisation concurrent de trop d'agents locaux peuvent à avoir un impact sur les performances de Bamboo, car ils s'exécutent dans la même machine virtuelle que Bamboo;

Pourquoi utiliser des agents distants? Il existe de bonnes raisons pour lesquelles, nous recommandons l'utilisation des agents distants, par exemple:

- Lorsque les ressources JVM du serveur Bamboo sont limitées ;
- Lorsque l'agent a besoin de fonctionnalités qui ne sont pas disponibles sur le serveur Bamboo ;
- Vous pouvez en fait avoir de nombreux agents installés sur plusieurs ordinateurs; seul l'agent en cours d'exécution doit être sous licence.

Ensuite, en termes de nombre d'agents à utiliser, tout dépend du nombre de tâches que vous souhaitez exécuter en parallèle, car un agent ne peut exécuter qu'une tâche à la fois. Donc, si vous construisez et testez séquentiellement, tout ira bien avec un seul agent. Mais dès que vous souhaitez exécuter plusieurs versions et tests simultanément, vous pouvez utiliser autant d'agents que nécessaire.

3.2. Création du projet

3.2.1. Création du projet Bamboo

Pour créer un projet Bamboo, connectez-vous à l'URL Bamboo et cliquez sur Create -> Create project.

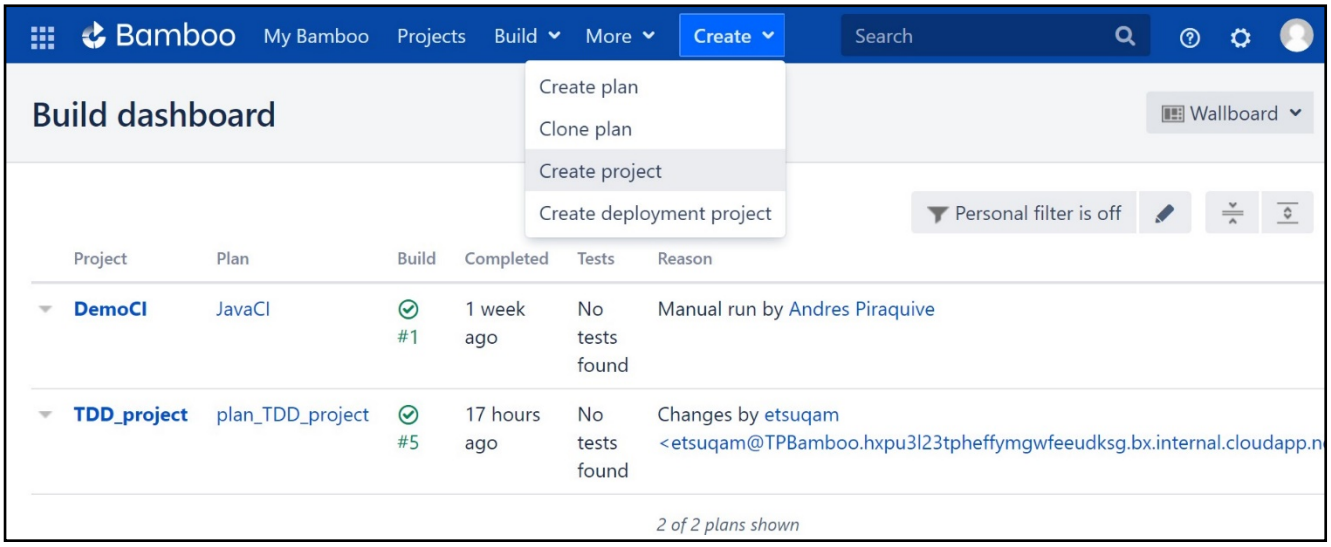


Figure 8 : Création du Projet

Entrez un nom et une description et cliquez sur *Save*.

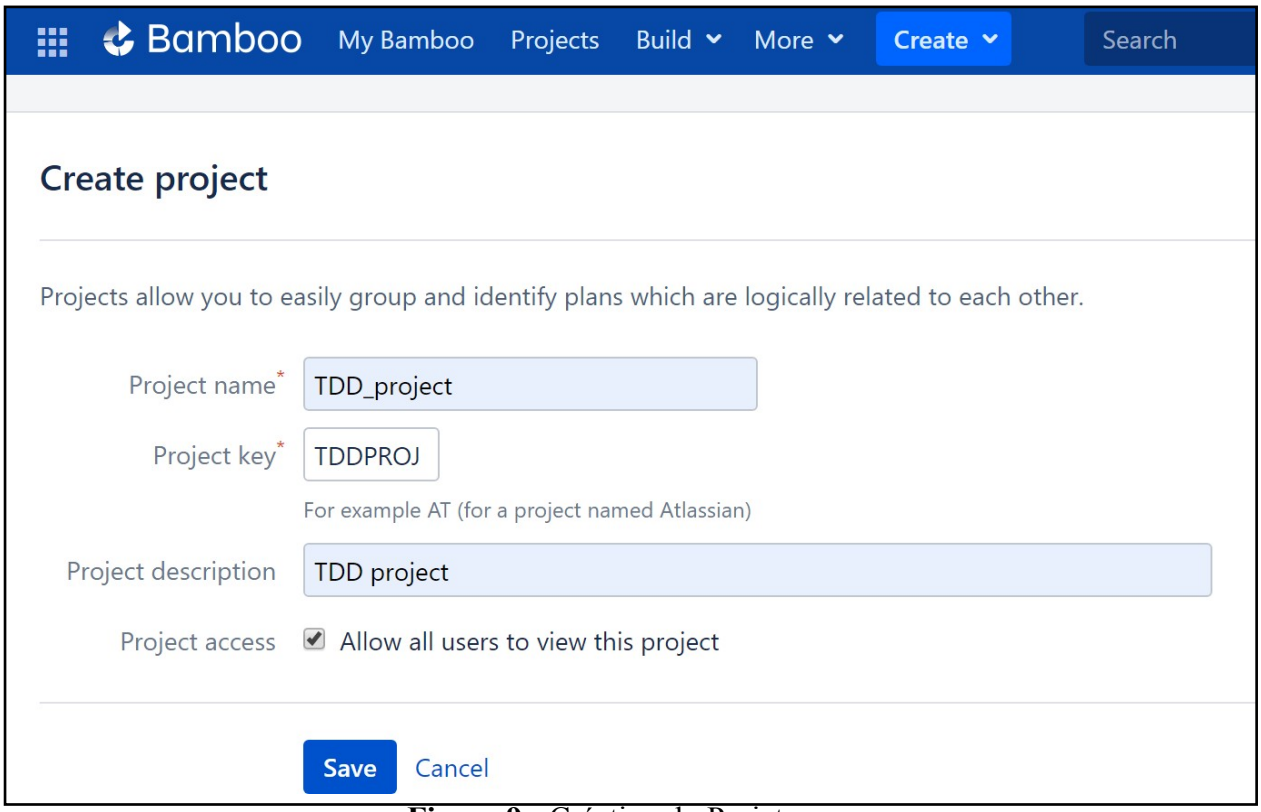


Figure 9 : Création du Projet

3.2.2. Liaison du projet Bitbucket Server à Bamboo

Par la suite, il nous faut lier notre projet dans le dépôt Bitbucket à Bamboo. Les étapes suivantes expliquent comment faire ce lien.

Une fois connecté à l'URL Bamboo, accédez à *Administration* -> *Manage applications* -> *Applications Link*. Ajoutez l'URL du serveur Bitbucket et cliquez sur le lien *Create new link*. Cela créera automatiquement un lien réciproque dans le serveur Bitbucket.

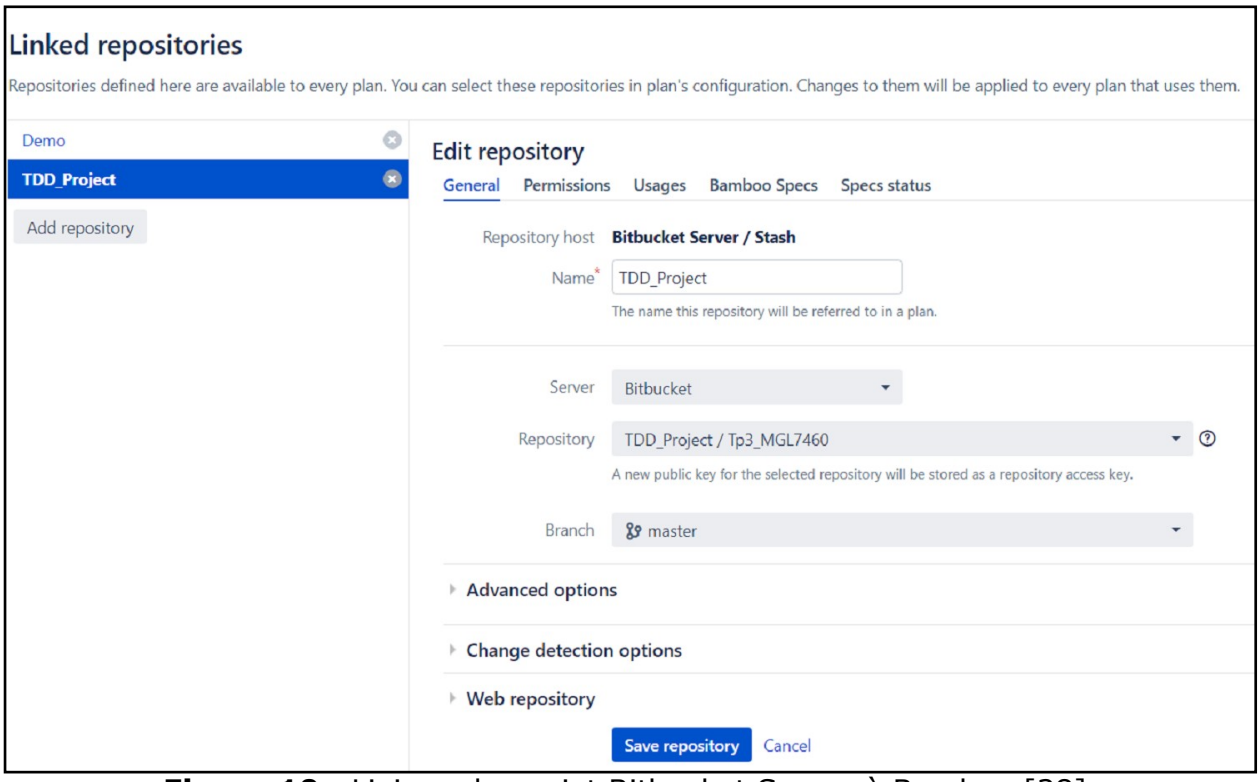


Figure 10 : Liaison du projet Bitbucket Server à Bamboo [28]

3.3. Création du plan

Une fois que le projet est lié à Bamboo, la prochaine étape vise la création du plan. Cliquer sur le menu *Create.* Ajouter les informations du plan. Le plan contient les informations du dépôt (« repository »). Dans notre cas, nous utilisons un projet Java avec Maven versionné dans **Bitbucket Server**. D'autres détails sont requis, tels que le contrôle d'accès pour le plan qui est également mentionné dans la configuration du plan.

Pour terminer cliquer sur le bouton *Configure plan*.

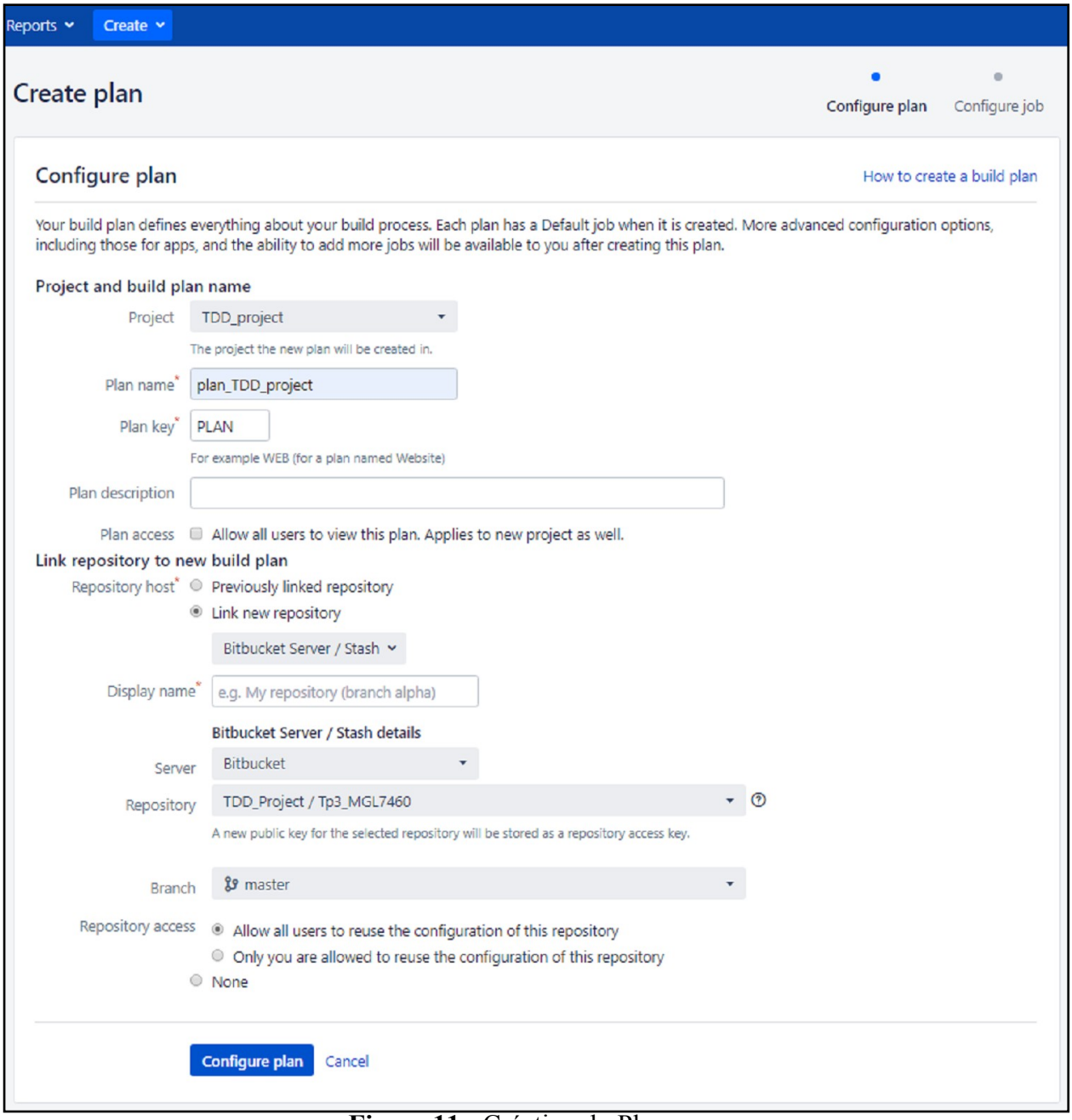


Figure 11 : Création du Plan

3.4. Création de stages

Tous les plans contiennent par défaut un Stage nommé *Default Stage*. Pour le configurer, sélectionnez dans le menu *Settings* > *Plan Configuration*.

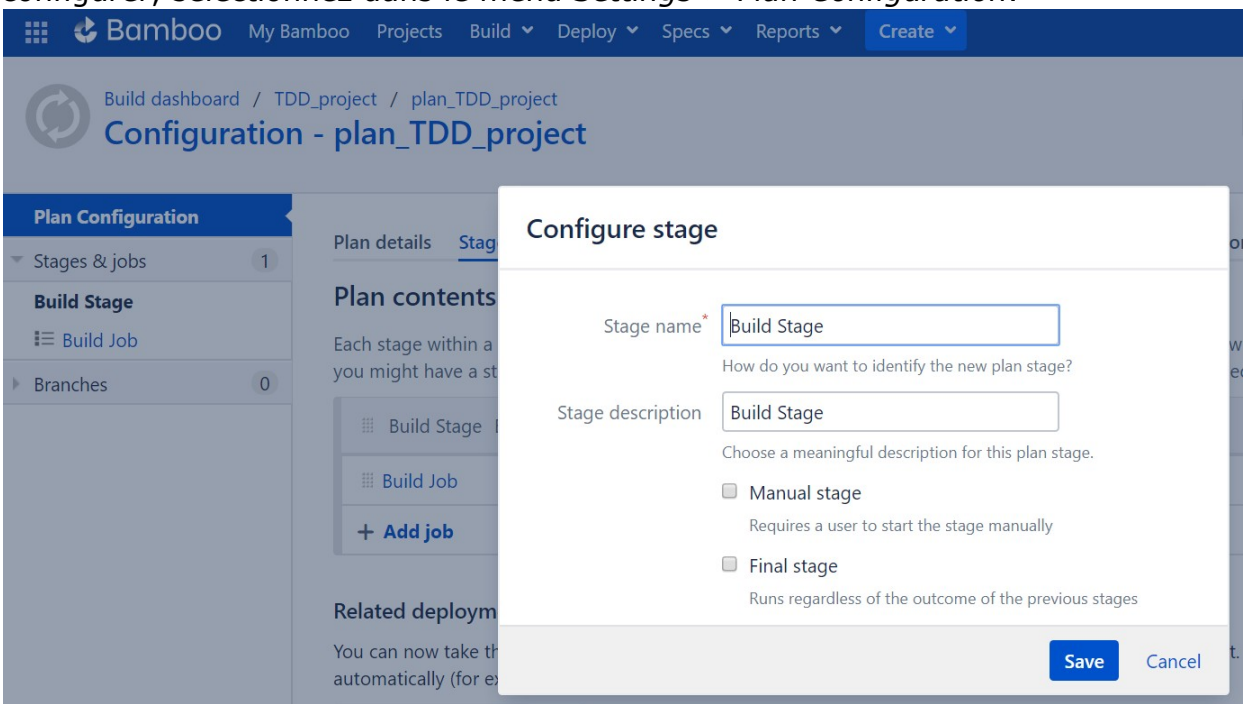


Figure 12 : Configuration du stage

Ensuite, éditez les informations dans la fenêtre *Configure stage*, et cliquer sur le bouton *Save*. Une fois la configuration terminée vous devez avoir un plan avec le *Build Stage* comme dans l'image de la figure 13 ci-dessous :

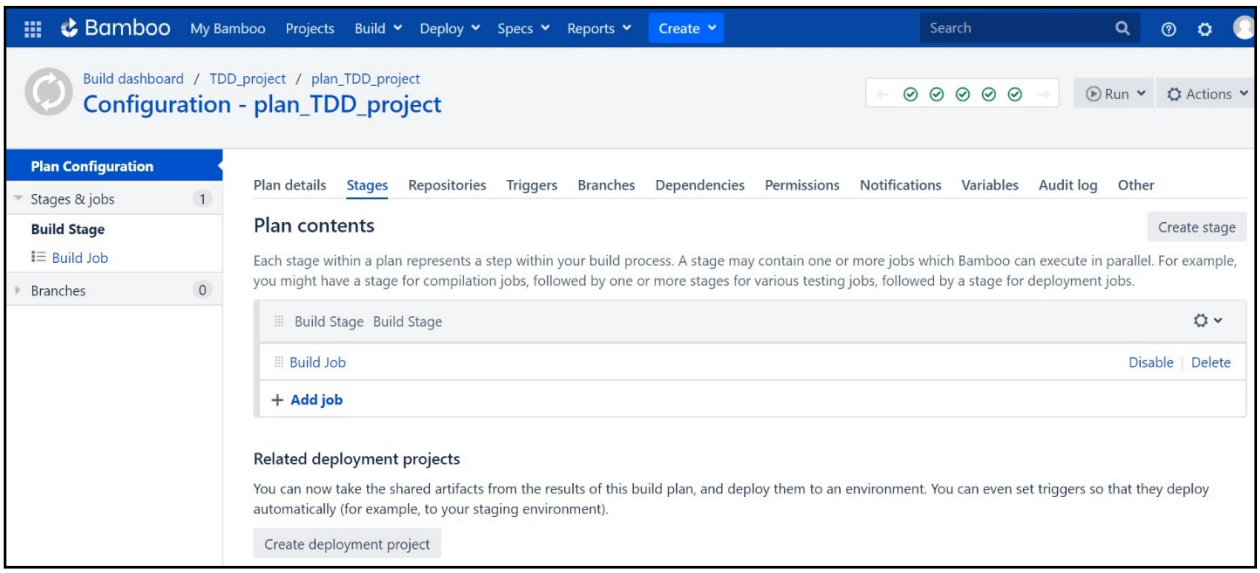


Figure 13 : Configuration du stage

3.5. Création de Jobs

Un job contient une ou plusieurs tâches qui roulent en parallèle. Nous avons utilisé un seul job pour notre démonstration, qui utilise un Local Agent dans le serveur de Bamboo. Un job est créé par défaut lors de la configuration du Plan, donc nous avons édité les informations du job dans le menu > *Job détails*.

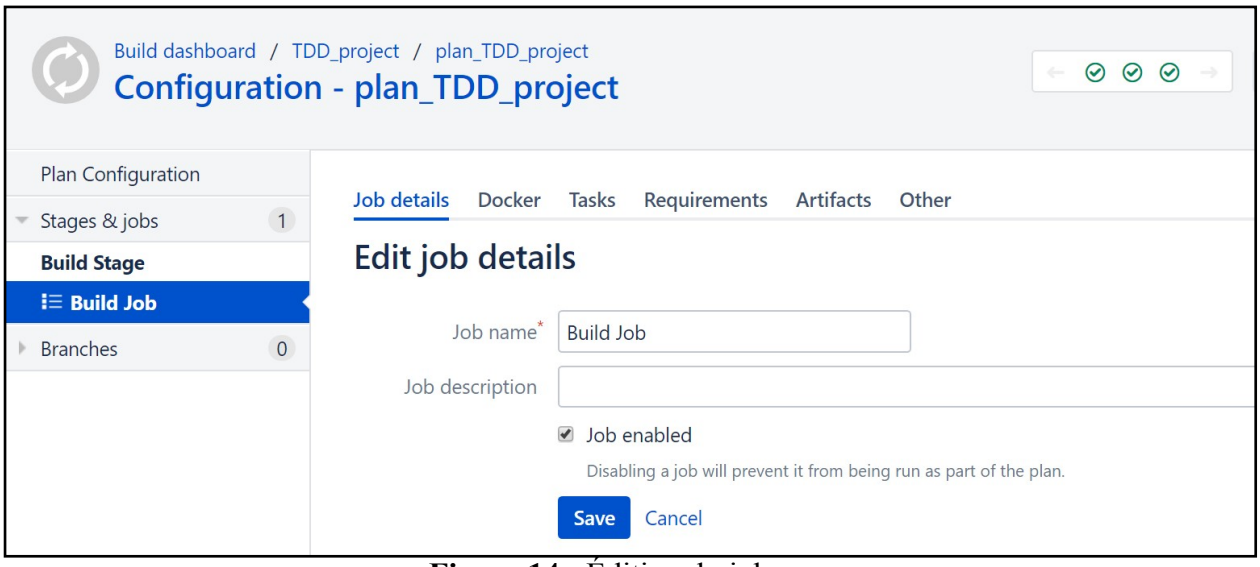


Figure 14 : Édition du job

Pour terminer, cliquer sur *Save.*

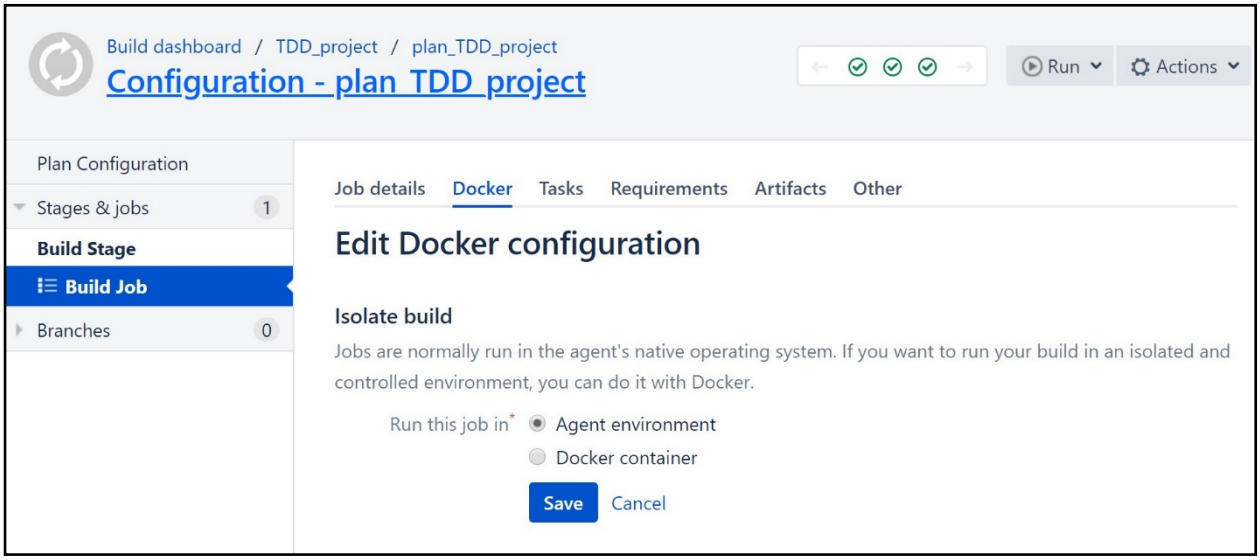


Figure 15 : Détails du job

3.6. Création des task

Cliquez sur *Add task*, puis sélectionner *Builder* dans la fenêtre de Task types et choisissez un type de tâche qui correspond à l'outil de génération de votre projet. Dans notre cas nous avons choisi *Maven* pour exécuter le Build du Projet Java.

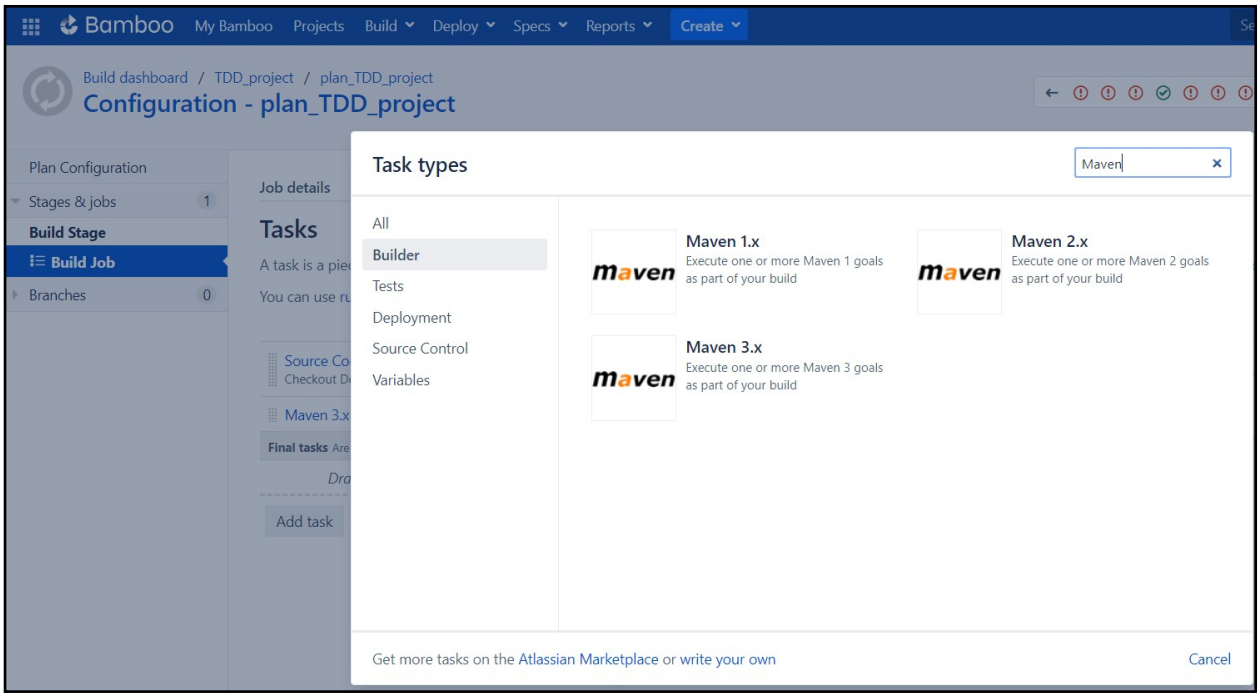


Figure 16 : Sélection du type de tâche

Sélectionner la version de *Maven* selon le votre projet, dans notre cas nous avons utilisée *Maven 3.x* qui est la dernière version disponible. Ensuite, insérer les informations dans la fenêtre pour configurer le *Build Task,* et cliquer *sur Save.*

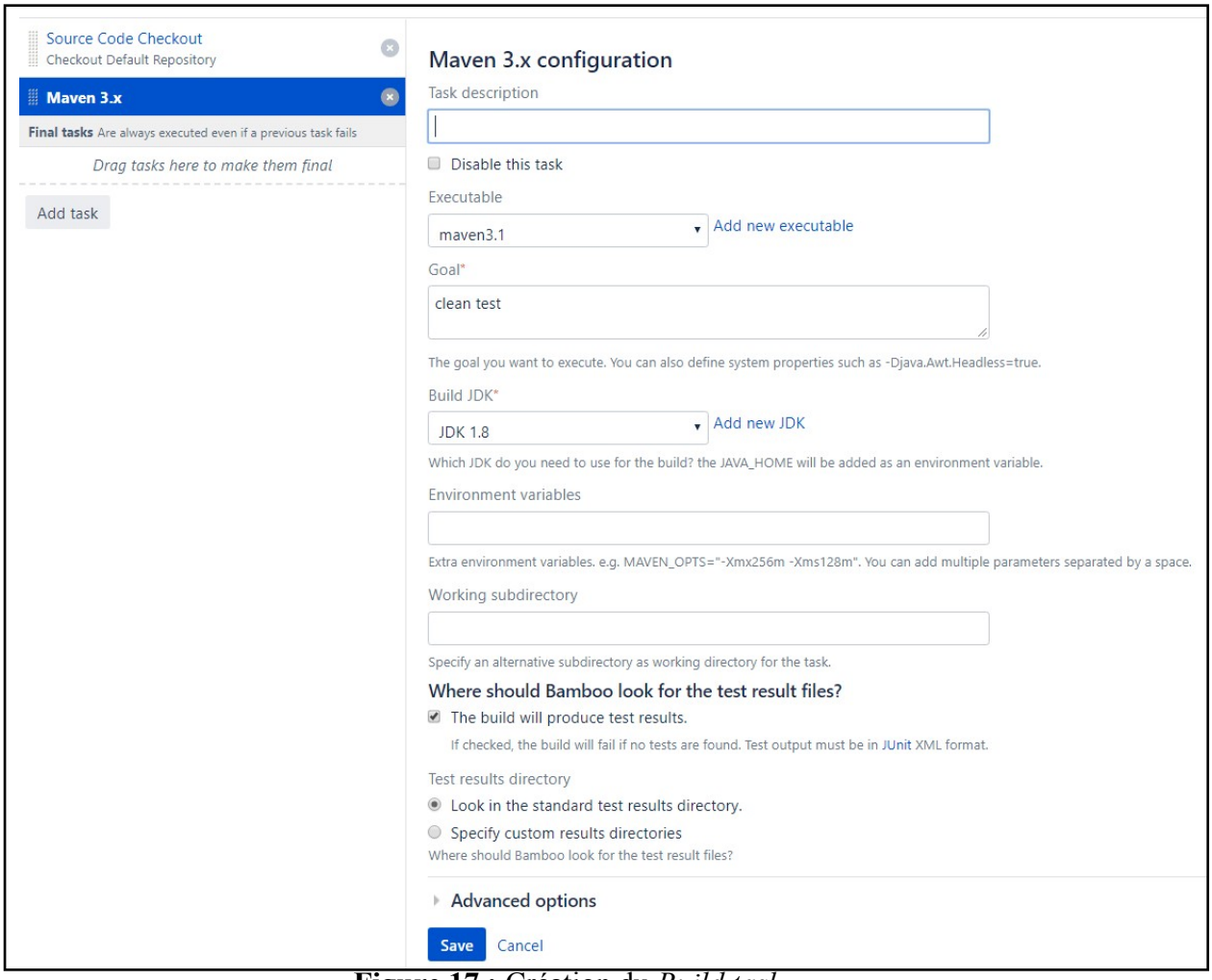


Figure 17 : Création du *Build task*

Lorsque vous terminer la création du *Build task,* le plan est prêt pour exécuter le *Build*. Vous pouvez voir dans l'image de la figure 18 ci-dessous, le Projet avec le plan, le stage *Build Stage* et le *Build Job.*

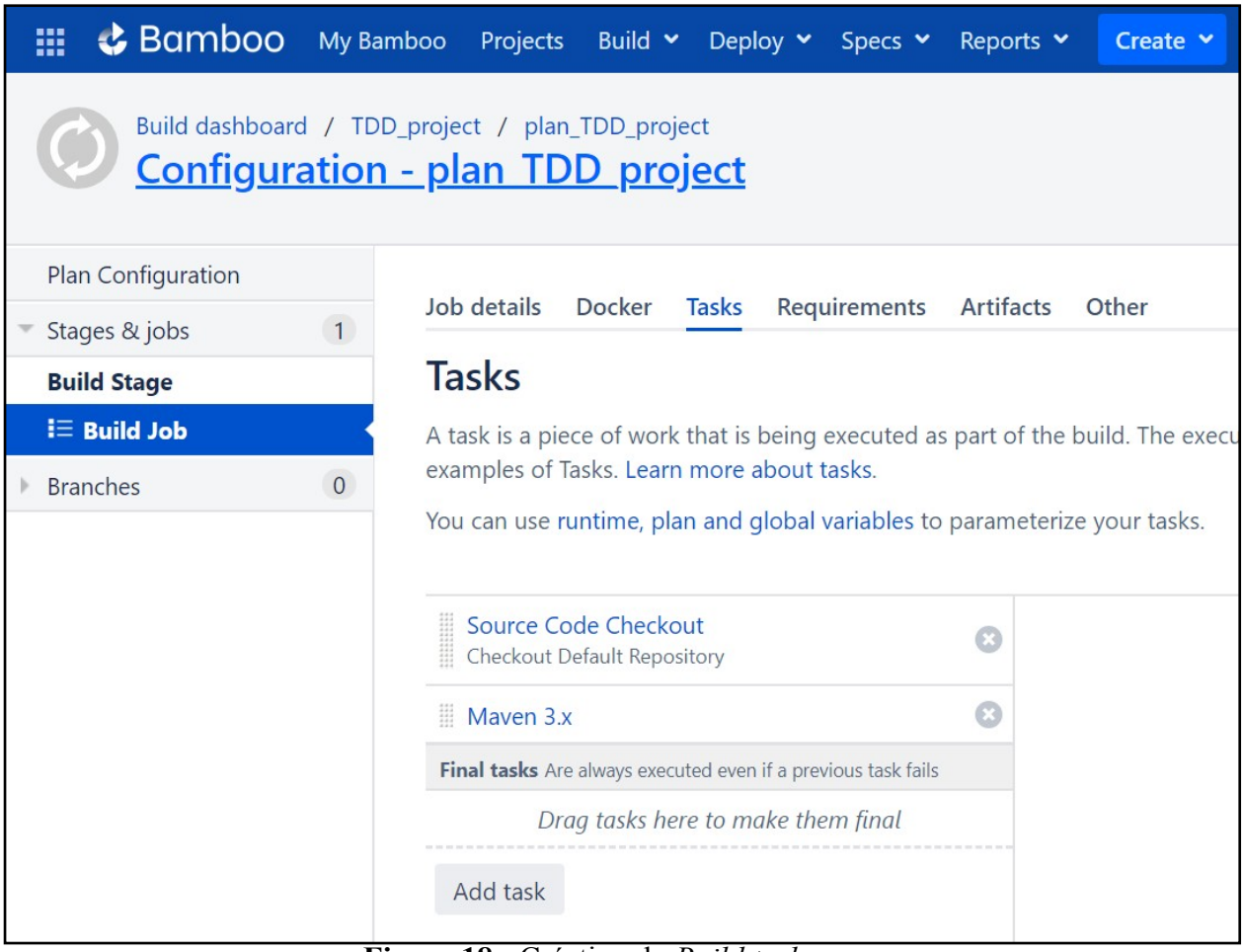


Figure 18 : Création du *Build task*

3.7. Envoi des notifications

Tel que décrit plus haut dans le processus de l'intégration continue, si la construction échoue, le serveur d'intégration continue envoie une notification aux personnes indiquées afin qu'une action soit prise. C'est la dernière étape du processus d'intégration continue. Bamboo dispose de trois stratégies pour envoyer ces notifications: 1) un *Wallboard* Bamboo [29]; 2) l'envoi des notifications par Email ou message instantané [30]; ou 3) l'affichage dans des flux RSS [31]. Nous avons choisi dans ce travail, les notifications par Email.

Cliquer sur *Administration > Overview,* puis sélectionner dans l'option Communication *Mail Server.* Ajouter les informations dans la page *Email settings*, puis cliquer sur *Save.*

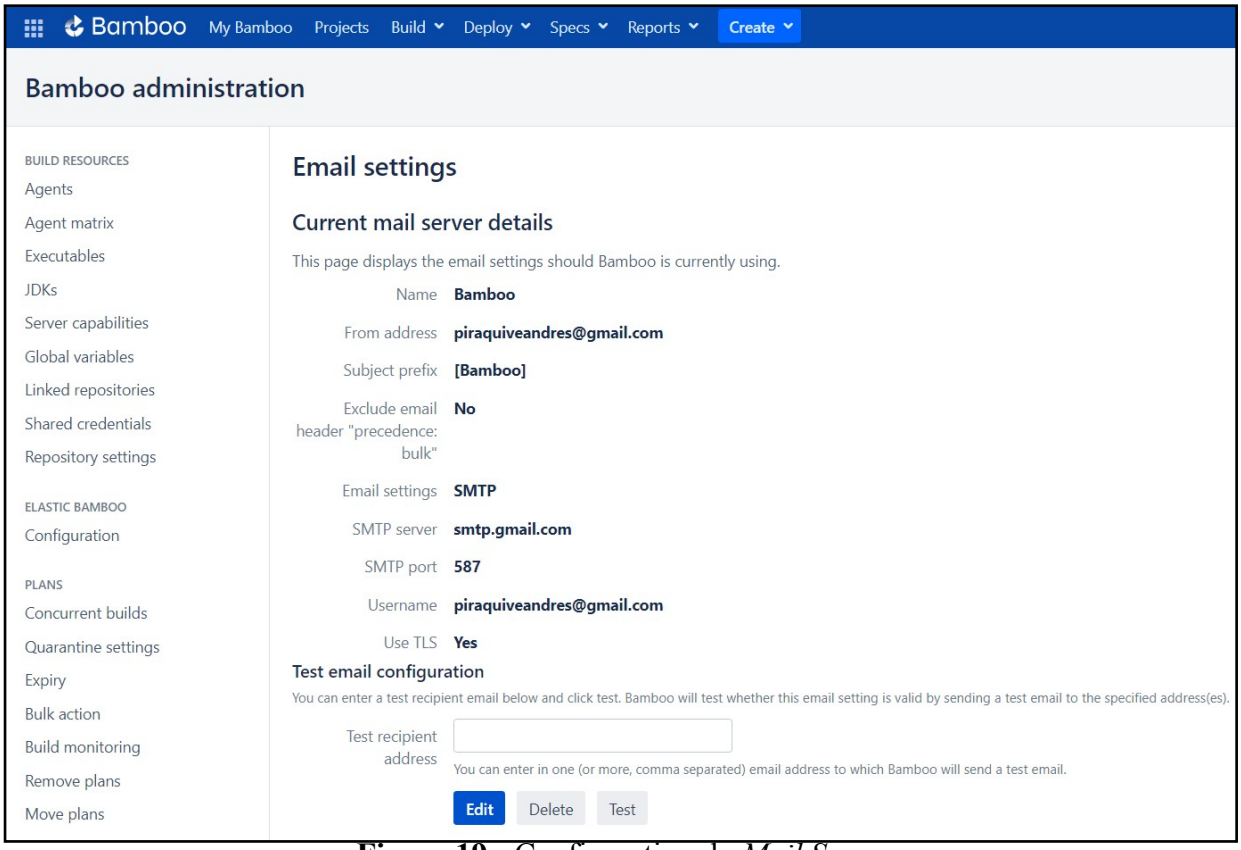


Figure 19 : Configuration du *Mail Server*

Une fois la configuration du *Mail Server* terminée, allez dans le *Plan Configuration* > *Notifications* > *Add notification.* Sélectionnez un *Event* et sélectionner un *User* ou un *Group.*

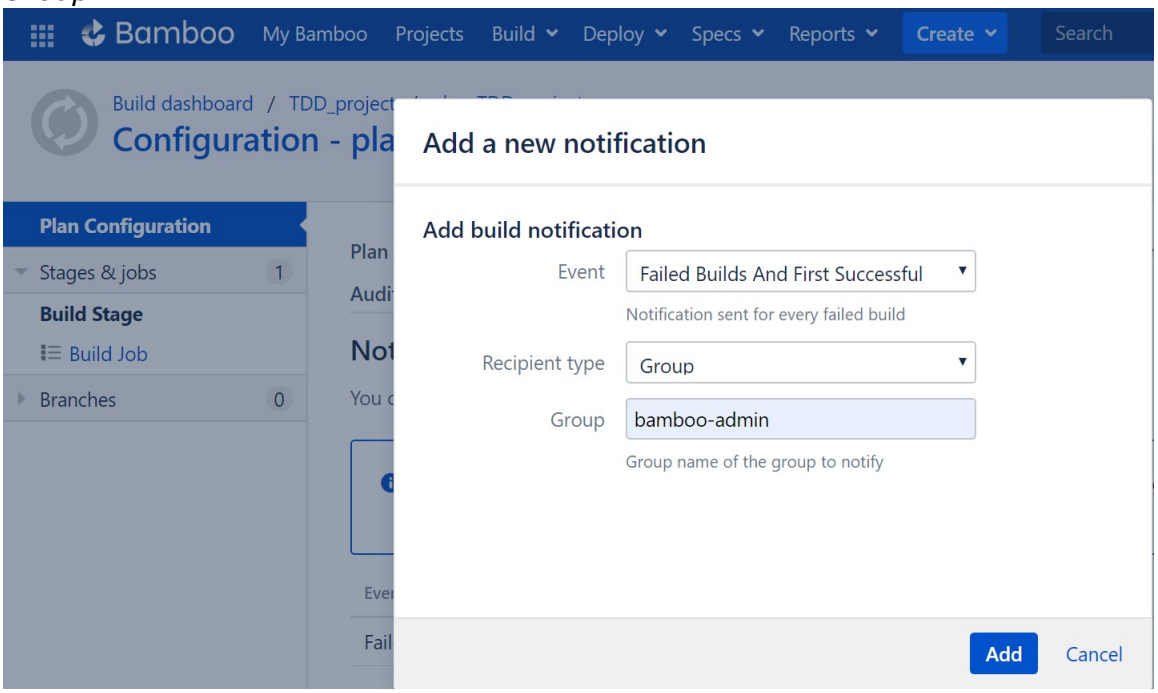


Figure 20 : Ajout d'une notification de type *Failed Builds*

Pour terminer cliquer sur *Add.*

Une fois terminée la configuration de la notification, allez dans le projet sur Bitbucket puis modifiez le code pour générer une erreur dans le *Build.* Puis une fois le code dans le master, validez que les personnes du groupe ont toutes reçues un email pour informer les utilisateurs que le *Build n'est pas passé avec succès.*

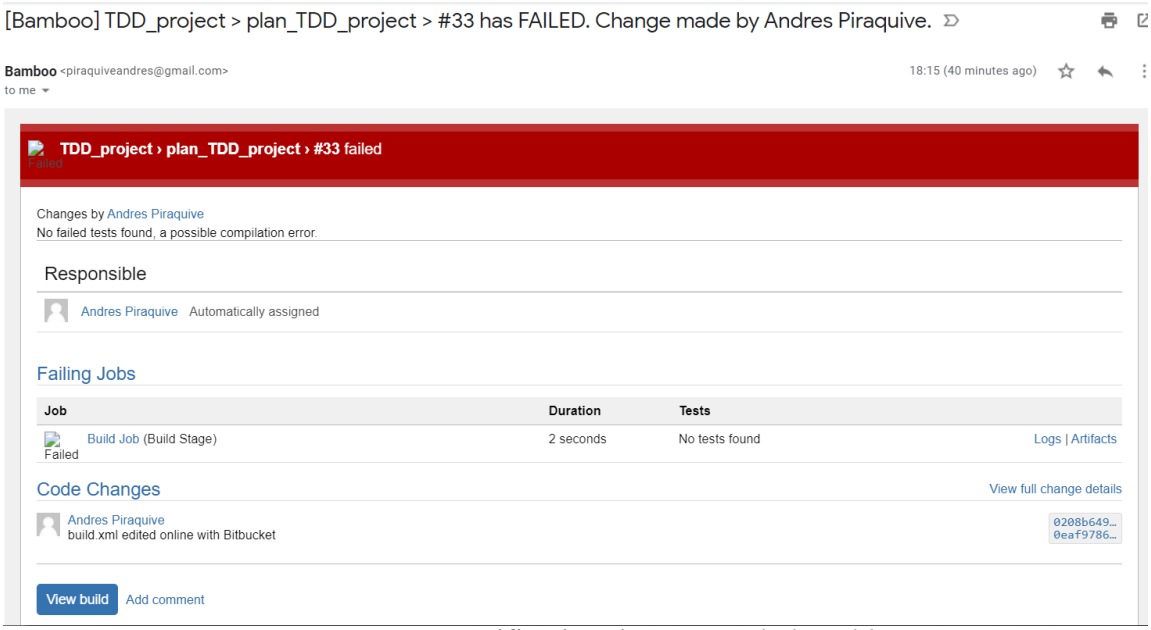


Figure 21 : Notification de type *Failed Builds*

4. **Annexe A** : Installation de Bamboo

Selon la documentation du site Atlassian [12], avant d'installer Bamboo, vous devrez vous assurer que votre système d'exploitation est bel et bien pris en charge par Bamboo et que la variable JAVA_HOME est correctement définie. La procédure d'installation suit les étapes suivantes :

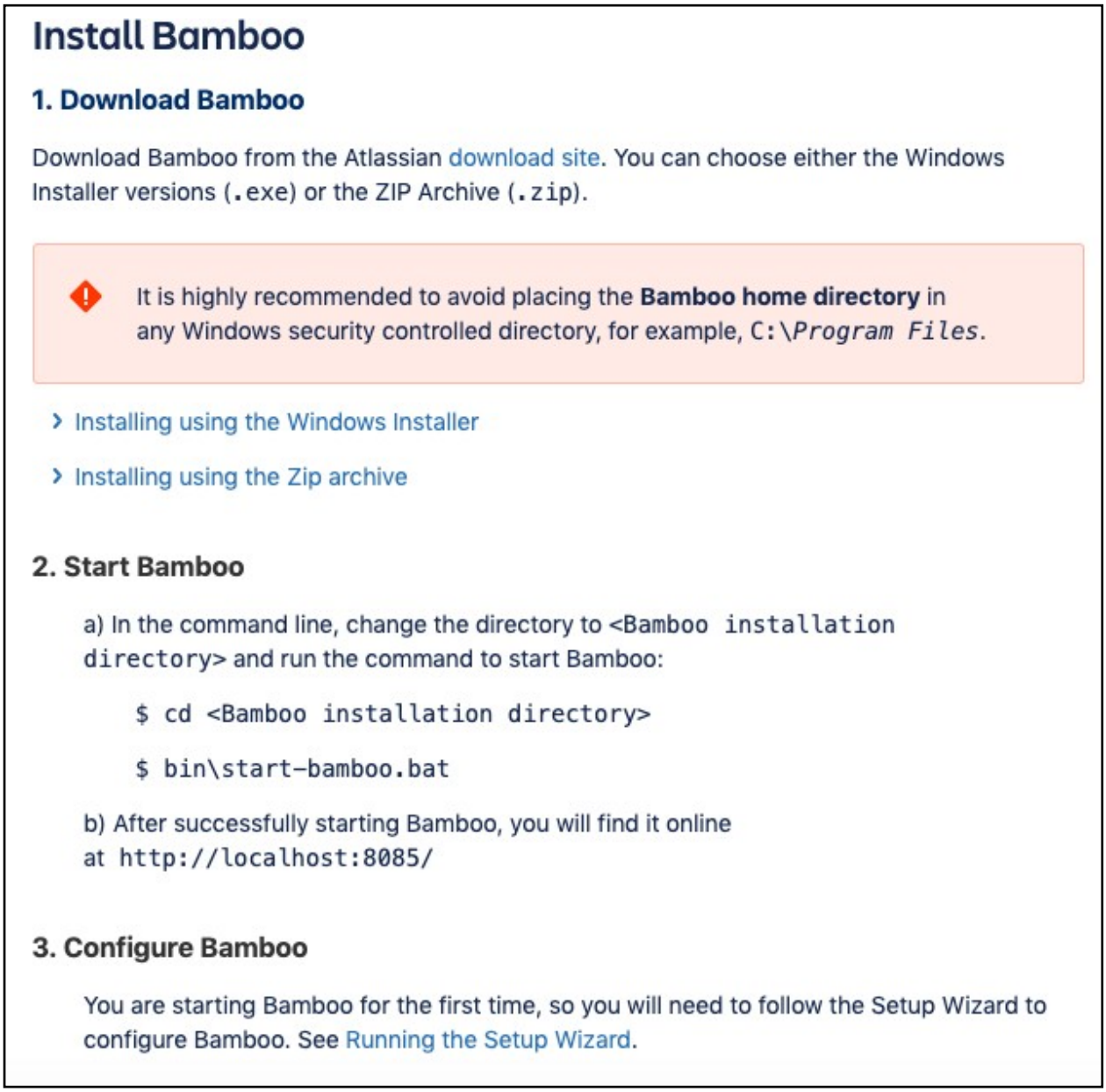


Figure 22 : Installation de Bamboo

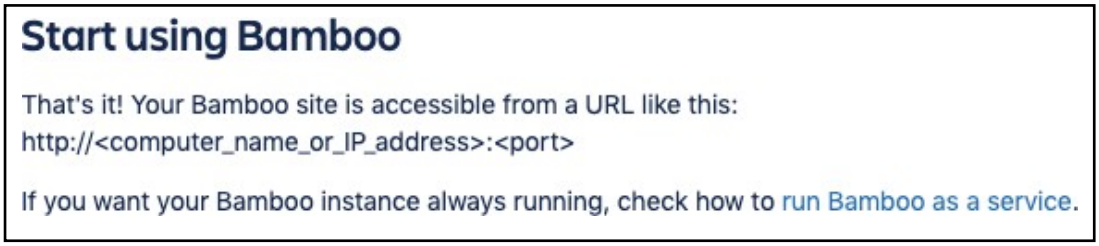


Figure 23 : Démarrer Bamboo

Annexe B : Démarrage de Bamboo

Ouvrir la fenêtre de Command Prompt (CMD), puis aller dans le répertoire où se trouvent les fichiers d'installation et exécuter la ligne de commande *bin\start-bamboo.bat,* voir l'image ci-dessous.

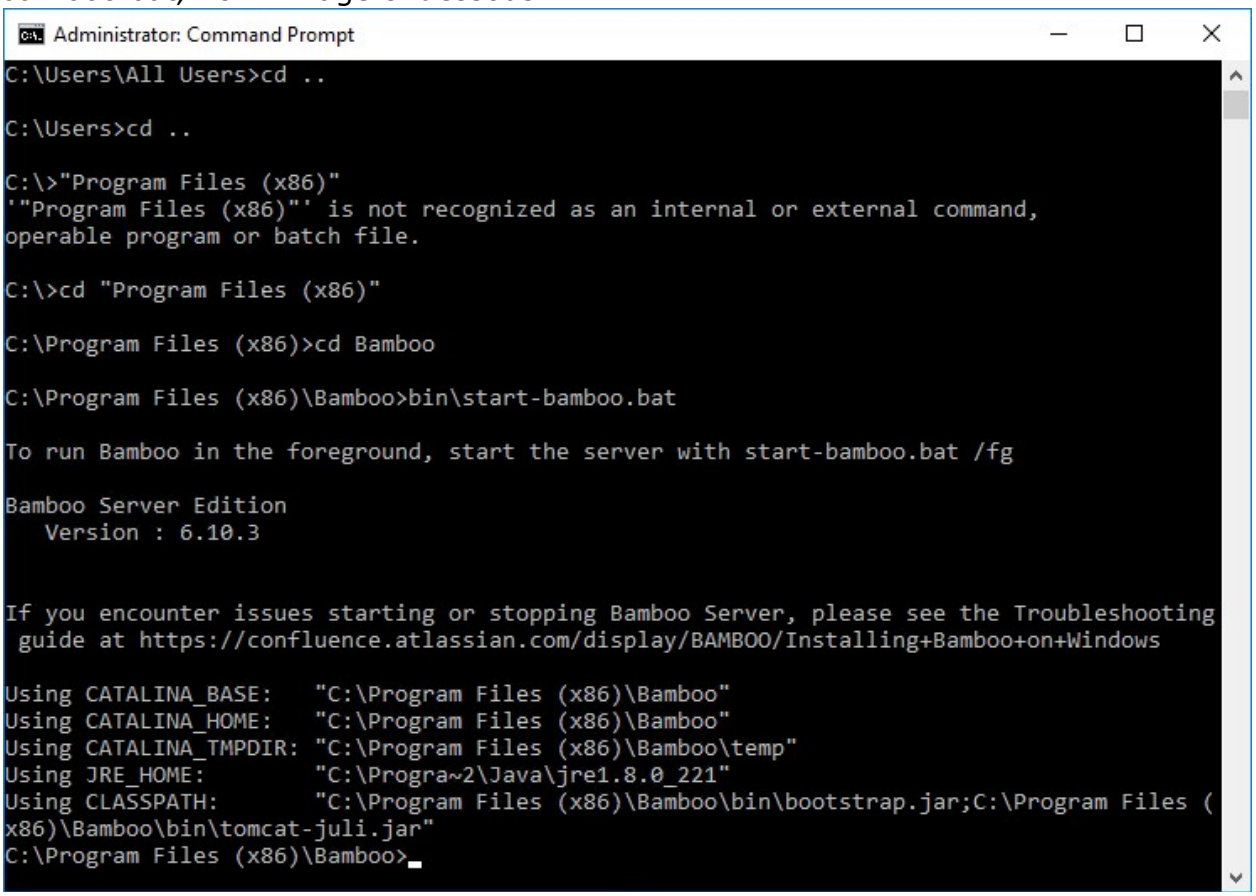


Figure 24 : Démarrage de Bamboo

Valider dans la console de Tomcat, qu'il n'y a pas d'erreurs. Puis ouvrir le navigateur et insérer l'adresse : http://localhost:8085/ dans l'URL.

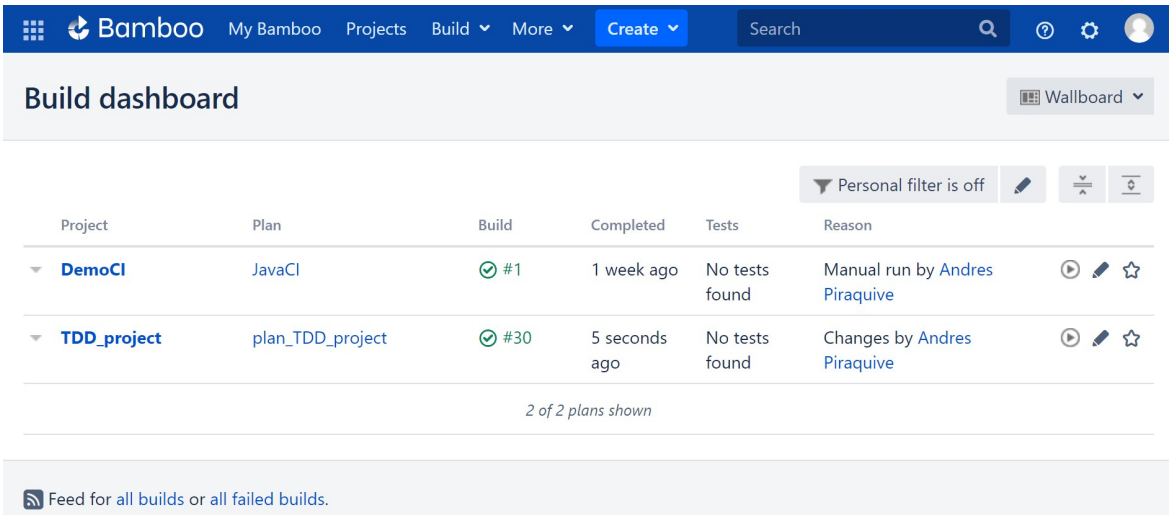


Figure 25 : Liste de projets

Annexe C : Installation de JAVA

Télécharger le Java JDK pour Windows x64 et faire l'installation dans la Machine Virtuelle, par la suite faire l'Installation du JRE de Java avant de débuter l'installation du Bamboo. Lorsque l'installation de JDK et JRE sont terminés, configurer les variables d'environnement pour Java dans Windows, cette étape est très importante, car lors du démarrage du serveur Bamboo, si les variables ne sont pas configurées, des erreurs seront affichées dans la console.

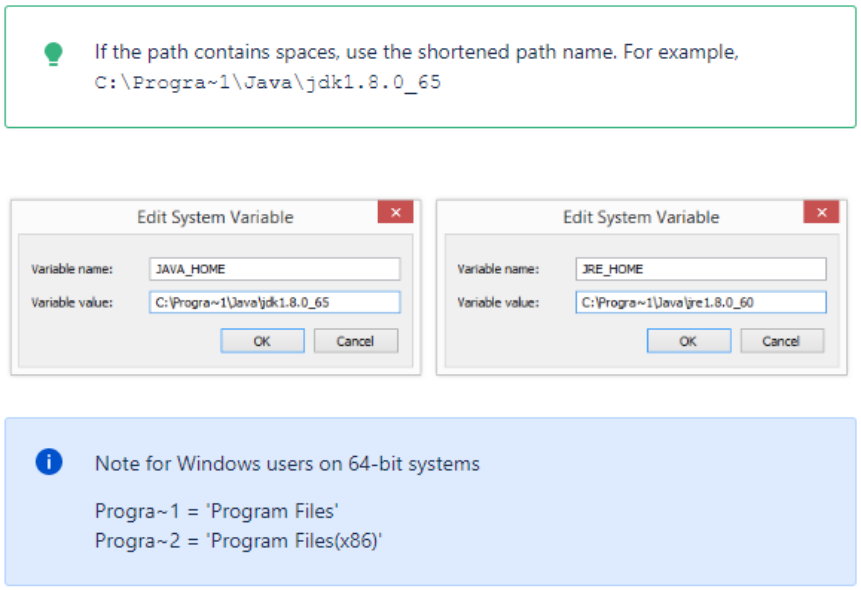
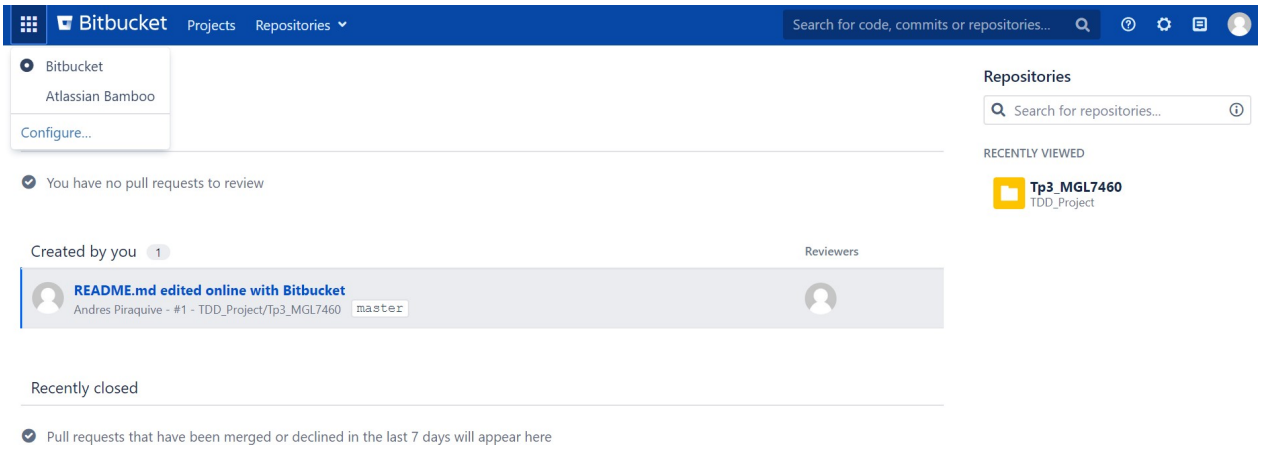


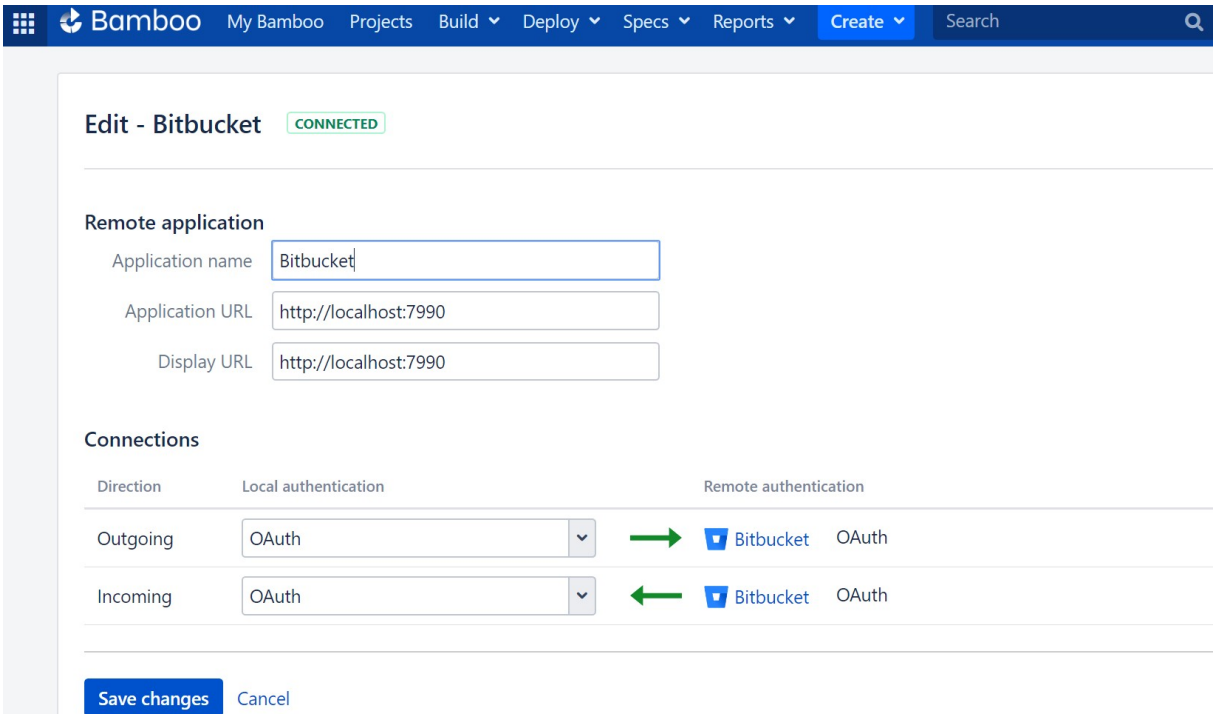
Figure 26 : Installation de JAVA

Annexe D: Bitbucket Server

Nous avons utilisé un projet Java avec Maven, que nous avons réalisé dans un autre cours en utilisant la technique de TDD. Puis nous avons ajouté le projet dans *Bitbucket Server*, et nous avons par la suite configuré dans Bamboo *Application Link* pour faire la liaison du projet entre Bitbucket et Bamboo.



Pour configurer le lien aller dans le menu Bamboo administration > configure Application Links > Create Link, et ajouter les informations pour créer le lien avec *Bitbucket Server.*



Cliquer sur *Save changes*.

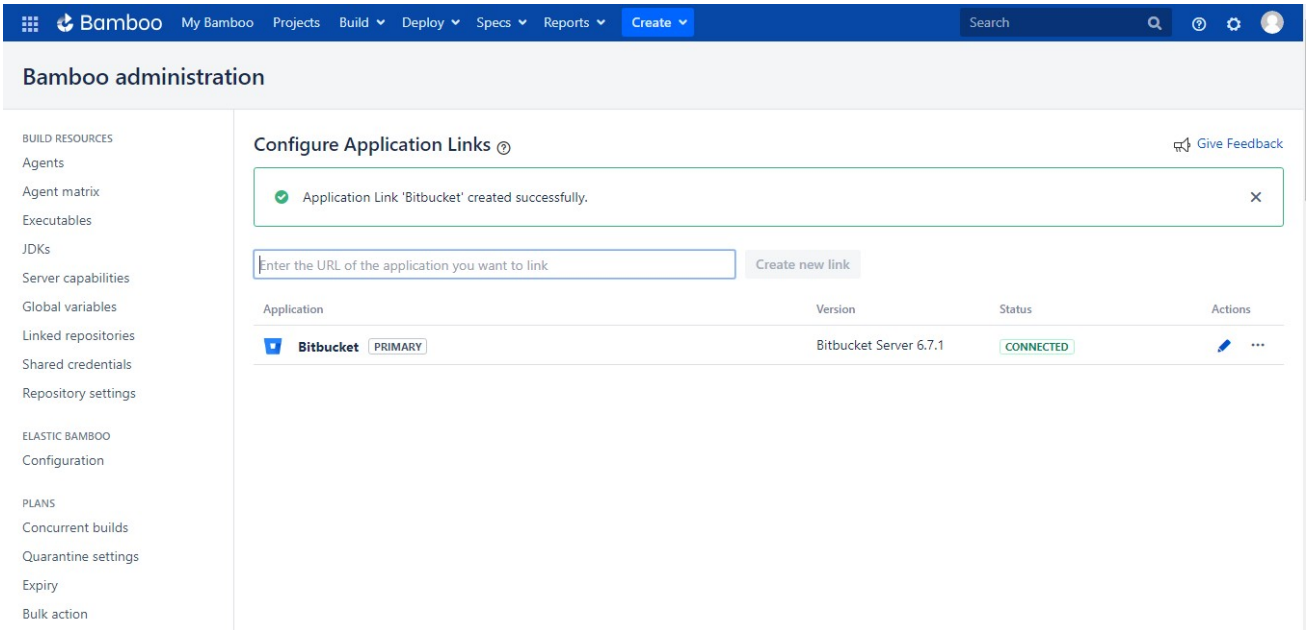


Figure 27 : Bitbucket Server

Annexe E: Création d'un agent

Création d'un agent local

1. Clique-le  icône dans l'en-tête Bamboo et choisissez **Présentation**.
2. Cliquez sur **Agents** dans le panneau de gauche (sous "Construire des ressources") pour afficher la liste de tous les agents locaux et distants existant dans votre système Bamboo.
3. Cliquez sur **Ajouter un agent local**.
4. Entrez les détails de l'agent. Le nom est affiché sur le tableau de bord. La description n'est visible que par les administrateurs.
5. Cliquez sur **Ajouter**.

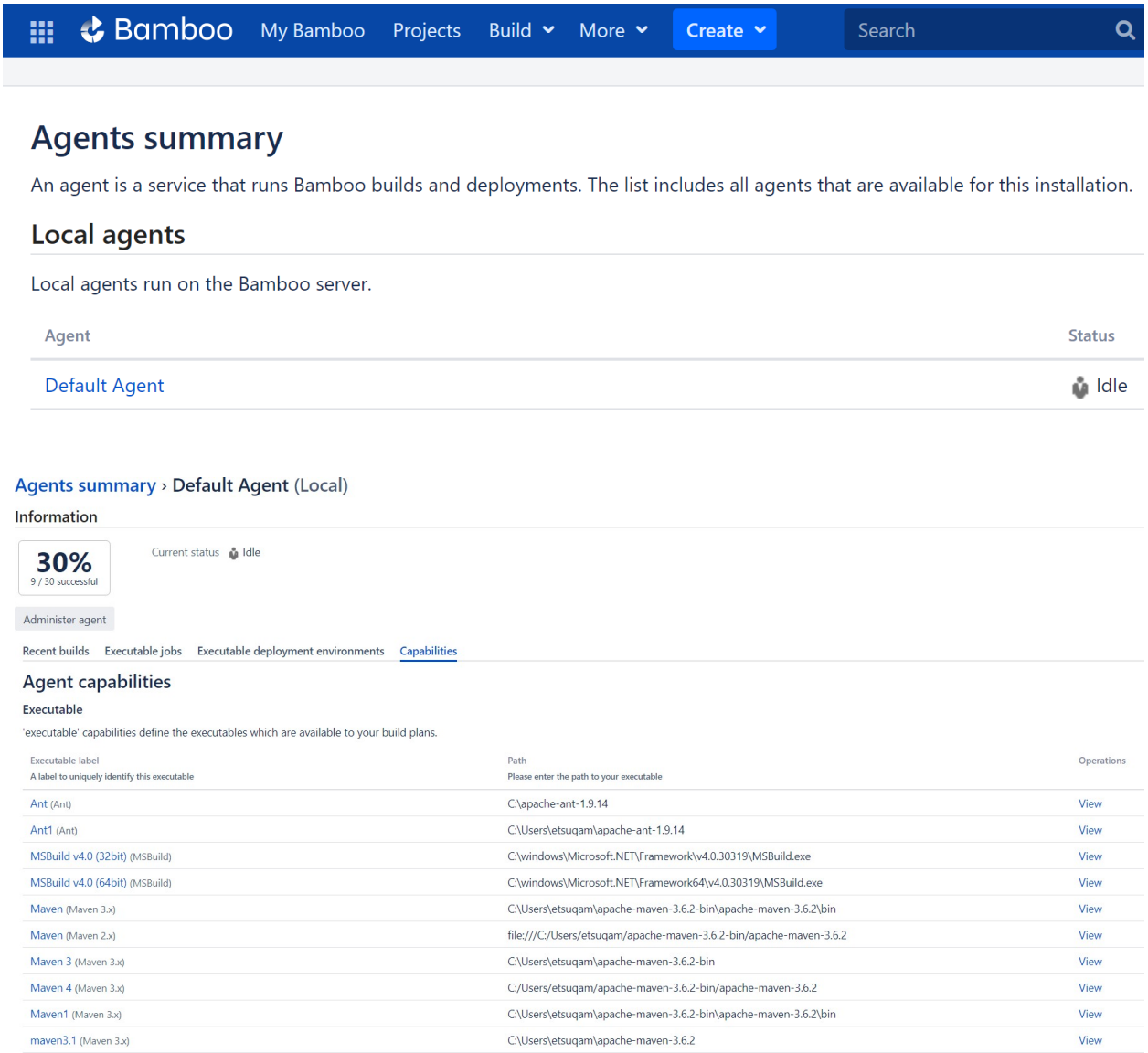


Figure 28 : Création d'un agent local

Notez que le nouvel agent local :

- sera activé par défaut;
- hériteront de toutes les fonctionnalités du serveur local définies dans le système Bamboo.

Prise en charge des agents distants

La désactivation de la prise en charge des agents distants dans Bamboo désactivera tous les agents distants et empêchera tout utilisateur de créer de nouveaux agents distants. Cette fonction ne supprimera aucun agent distant que vous avez déjà créé. Pour supprimer un agent distant, voir Désactivation ou suppression d'un agent.

Notez que la prise en charge des agents distants doit être activée pour utiliser les Agents élastiques de Bamboo.

Pour activer ou désactiver la prise en charge des agents distants [22]**:**

1. Clique-le icône dans l'en-tête Bamboo et choisissez **Présentation**.
2. Dans le panneau de gauche, sous Ressources de construction, cliquez sur Agents.
3. Cliquez sur **Activer la prise en charge de l'agent distant** ou **Désactiver la prise en charge de l'agent distant.**

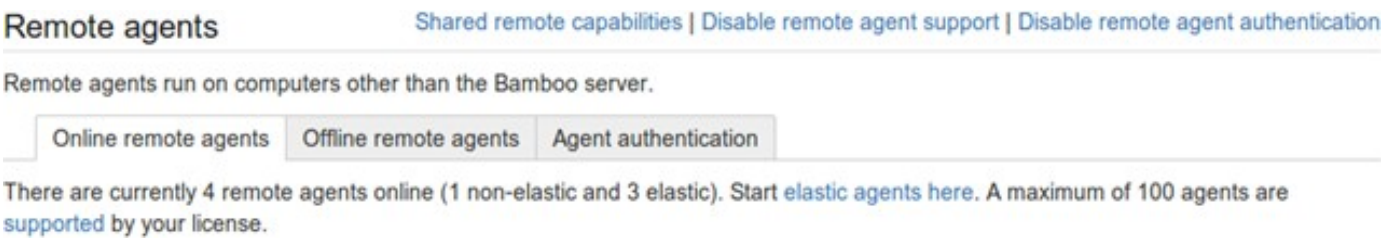


Figure 29 : Activer ou désactiver la prise en charge des agents distants [22]

Téléchargez et installez un agent distant [22]

1. Créez un répertoire sur la machine de l'agent (par exemple, bamboo-agent-home) pour servir de répertoire principal de l'agent Bamboo pour l'agent distant.

2. Clique-le icône dans l'en-tête Bamboo et choisissez **Présentation**.

3. Dans le panneau de gauche, sous Créer une ressource, cliquez sur A **Messieurs** dans le panneau de gauche.

L'écran "Agents" affiche les listes de tous les agents locaux et de tous les agents distants qui existent actuellement sur votre système Bamboo.

4. S'il n'est pas déjà activé, cliquez sur le lien **Activer le support de l'agent distant.**

5. Cliquez sur **Installer l'agent distant.** L'écran 'Installation d'un agent distant' s'affichera.

6. Cliquez sur **DOWNLOAD Remote Agent JAR** et enregistrez le fichier JAR dans le répertoire de la machine d'agent que vous avez créé ci-dessus.

7. Copiez la commande sous "Exécution d'un agent distant" dans le presse-papiers pour pouvoir l'utiliser à l'étape 3 ci-après.

Lancez l'agent distant

Une fois installé, exécutez l'agent distant en exécutant la ligne de commande obtenue ci-dessus. Cette commande ressemblera à ceci :

Exemple : java -jar atlassian-bamboo-agent-installer-XX-SNAPSHOT.jar http://bamboo-host-server:8085/bamboo/agentServer/

N.B : Le nom du fichier jar, par exemple, **atlassian-bamboo-agent-installer-5.4-SNAPSHOT.jar**, varie en fonction de la version de Bamboo que vous exécutez.

Vous pouvez exécuter l'agent distant avec un certain nombre de paramètres de ligne de commande. Les options de configuration comprennent le stockage des données de l'agent distant, la détection et la journalisation des fonctionnalités, la suppression du certificat auto signé et l'exécution sans le superviseur d'agent distant ou avec des commandes de démarrage différentes. Pour plus d'informations, voir les options supplémentaires pour l'agent distant [28].

Désactiver ou supprimer un agent [24]

Bamboo vous permet de désactiver ou de supprimer un agent afin d'empêcher cet agent d'exécuter d'autres tâches.

- La désactivation d'un agent vous permet de conserver l'agent dans Bamboo, mais l'empêche d'exécuter des travaux.

- La suppression d'un agent le supprimer complètement de Bamboo. Si vous devez utiliser l'agent à nouveau ultérieurement, vous devrez le créer à nouveau

1. Cliquez le  icône dans l'en-tête Bamboo et choisissez Présentation.
2. Cliquez sur Agents dans le panneau de gauche pour afficher l'écran "Agents", qui répertorie tous les agents existant dans votre système Bamboo. La colonne "Statut" indique les agents actuellement activés ou désactivés. Faites défiler vers le bas si vous avez besoin d'agents distants.
3. Cochez la case correspondant à l'agent (ou aux agents) que vous souhaitez désactiver ou supprimer.
4. Cliquez sur le bouton Désactiver (ou Supprimer) au-dessus du tableau.

Capture d'écran: Agent - Supprimer ou désactiver l'agent distant

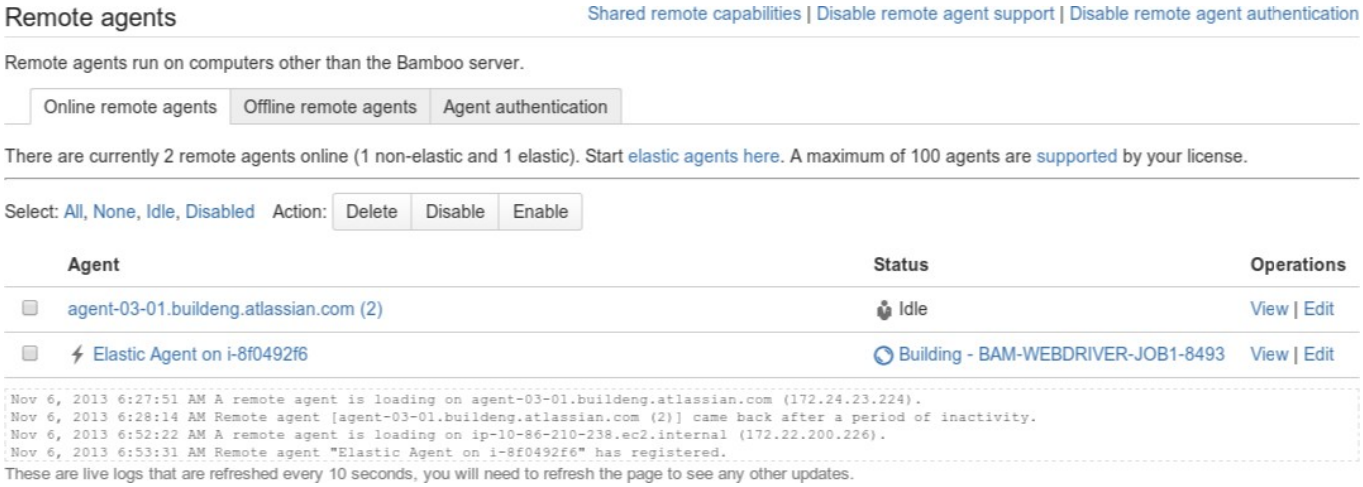


Figure 30 : Désactiver ou supprimer un agent [24]

(Voir Création d'un agent local [21] et Création d'un agent distant [22] pour plus d'informations).

1. Références

[1] (Juillet 2016). Intégration continue https://fr.wikipedia.org/wiki/Intégration_ continue.

[2] (Mars 2010), Intégration continue, Agile Nantes. http://www.agilenantes.org/wp-content/uploads/2010/02/Presentation-IC-agileNantes1.pdf

[3] http://lyoncalcul.univ-lyon1.fr/ed/DOCS_2016-2017/cours_CI.pdf

[4]
Paul M. Duvall, Steve Matyas, Andrew Clover.(February 2008): Improving Software Quality and Reducing Risk

[5] Martin Fowler. (01 May 2006) Continuous Integration

[6] https://plugins.jenkins.io/

[7] https://docs.travis-ci.com/

[8] https://confluence.atlassian.com/bamboo/understanding-the-bamboo-ci-server-289277285.html

[9] https://www.jetbrains.com/teamcity/

[10] Consulté le 18 septembre 2019. Bamboo tiré de https://en.wikipedia.org/wiki/Bamboo_(software)

[11]https://www.atlassian.com/fr/software/bamboo/comparison/bamboo-vs-jenkins

[12]https://confluence.atlassian.com/bamboo/installing-bamboo-on-windows-289276813.html

[13]https://confluence.atlassian.com/bamboo/understanding-the-bamboo-ci-server-289277285.html

[14]https://docs.vmware.com/fr/VMware-Horizon-Cloud-Service/services/hzncloudmsazure.admin15/GUID-078E46E1-CBC2-43B0-B1E4-A4E0E9571AB1.html

[15]https://confluence.atlassian.com/doc/setting-the-java_home-variable-in-windows-8895.htm

[16]https://confluence.atlassian.com/bamboo/getting-started-with-java-and-bamboo-289277286.html

[17]https://confluence.atlassian.com/bamboo/getting-started-with-php-and-bamboo-289277284.html

[18]https://confluence.atlassian.com/bamboo/getting-started-with-node-js-and-bamboo-687213472.html

[19]https://confluence.atlassian.com/bamboo/getting-started-with-net-and-bamboo-289277288.html

[20]https://confluence.atlassian.com/bamboo/how-do-i-connect-bamboo-to-an-unsupported- database-628720208.html

[21]https://confluence.atlassian.com/bamboo/creating-a-local-agent-289277175.html

[22]https://confluence.atlassian.com/bamboo/bamboo-remote-agent-installation-guide-289276832.html

[23]https://confluence.atlassian.com/bamboo/working-with-elastic-bamboo-289277117.html

[24]https://confluence.atlassian.com/bamboo/disabling-or-deleting-an-agent-289277174.html

[25] https://confluence.atlassian.com/bamkb/difference-between-local-agents-and-remote-agents-457703602.html

[26]https://stackoverflow.com/questions/2070499/running-javascript-unit-tests-headlessly-in-a-continuous-integration-build/

[27]https://www.atlassian.com/blog/archives/continuous_integration_javascript_jquery_qunit

[28] https://confluence.atlassian.com/bamboo/linking-to-source-code-repositories-671089223.html

[29]https://confluence.atlassian.com/bamboo/displaying-the-wallboard-289276971.htm

[30] https://confluence.atlassian.com/bamboo/configuring-notifications-for-a-plan-and-its-jobs-289276973.html

[31]https://confluence.atlassian.com/bamboo/subscribing-to-rss-feeds-289276970.html