École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5e année
2014-2015

Rapport final de projet de fin d'études

Mise en place d'un serveur d'intégration continue et de qualité de code

Encadrants
Nicolas Ragot
nicolas.ragot@univ-tours.fr
Pascal Meichel
pascal.meichel@univ-tours.fr
Vincent T'Kindt
tkindt@univ-tours.fr

Université François-Rabelais, Tours

Étudiant
Florent Clarret
florent.clarret@etu.univ-tours.fr

DI5, 2014-2015

Table des matières

Table des figures

Remerciements

Tout d'abord, je tiens à remercier Messieurs Nicolas Ragot, Pascal Meichel ainsi que Vincent T'Kindt pour avoir encadrer mon projet de fin d'études et m'aider dans la réalisation de celui-ci tout au long de cette année. De plus, j'adresse un remerciement particulier à Monsieur Pascal Meichel pour avoir été le représentant du service informatique ainsi que d'avoir été l'interlocuteur avec le service informatique de l'université, me permettant ainsi d'obtenir le matériel nécessaire au bon déroulement de mon projet.

Je souhaite également remercier l'équipe du service informatique de Polytech pour m'avoir aidé à résoudre divers problèmes que j'ai pu rencontrer au cours de ce projet.

J'adresse un remerciement tout particulier à l'ensemble des étudiants qui m'ont aidé à tester le bon fonctionnement du serveur d'intégration continue et de qualité de code ainsi que pour leurs retours sur le travail que j'ai effectué. Le temps qu'ils ont passé dans le cadre de leur projet de fin d'études, de projets collectifs ou de mini-projets, m'a permis notamment d'améliorer les divers documents que j'ai eu l'occasion de rédiger.

Enfin, j'adresse un pensée particulière à Madame Betty Vias ainsi qu'à tous les étudiants ayant travaillé à la bibliothèque tout au long de cette année avec moi.

Présentation et contexte du projet

2.1 Présentation du projet

Au cours de ma dernière année à l'Ecole Polytechnique de l'Université de Tours (Polytech' Tours), j'ai réalisé mon projet de fin d'études intitulé "Mise en place d'un serveur d'intégration continue et de qualité de code". Ce projet a été encadré par Messieurs Nicolas Ragot, Pascal Meichel et Vincent T'Kindt.

L'objectif de ce projet est de mettre à disposition aux étudiants ainsi qu'aux enseignants-chercheurs et doctorants un serveur leur permettant de mettre en place une démarche d'intégration continue et de qualité de code. Cette démarche pourra se faire dans le cadre de projets tels que des projets de fin d'études, des projets collectifs ou de recherche.

De nos jours, les projets ont tendance à devenir de plus en plus imposants, avec parfois de nombreux intervenants. Il a donc été nécessaire de développer des méthodes qui permettent de faciliter le travail de chacun des membres du groupe de travail, mais aussi de simplifier les étapes d'intégration des nouveaux composants dans le projet. Parmi ces logiciels, nous pouvons distinguer par exemple les logiciels de gestion de versions. Ils permettent de pouvoir simplifier et versionner le travail de chacun des membres du projet. Un serveur de gestion de versions existe d'ores et déjà au sein de Polytech' Tours et fonctionne avec Subversion. Ensuite, il y a les serveurs d'intégration continue tels que Jenkins par exemple. Ces serveurs permettent d'effectuer des vérifications périodiques sur le projet comme le lancement des tests unitaires. Nous pouvons aussi parler des outils d'analyse de code qui permettent de vérifier le bon respect de normes et de conventions dans le code source d'une application. Pour plus d'informations sur ces technologies, vous pouvez vous référer au cahier de spécification rédigé lors de la première phase de ce projet. Ce document est fourni avec ce rapport, il n'est pas fourni directement en annexe pour ne pas surcharger ce rapport. Tous ces outils sont de nos jours fortement utilisés au sein des entreprises, car ils permettent d'assurer un certain niveau de qualité des logiciels et applications produits.

2.2 Présentation de l'existant

Ce projet de fin d'études est la suite directe du projet réalisé durant l'année universitaire 2013/2014 par Anne Castier. Ce dernier avait pour but d'étudier les différentes solutions possibles pour mettre en place au sein du département informatique de Polytech Tours un serveur d'intégration continue. Pour cela, elle avait donc réalisé une première phase de veille technologique afin de pouvoir dresser une liste de toutes les technologies et logiciels que nous pouvions utiliser dans ce cadre. Cette liste comportait l'ensemble des critères qu'il était nécessaire de prendre en compte afin d'effectuer ce choix. Une fois cette liste mise en place, Anne ainsi que ses encadrants ont porté leur choix sur les logiciels les plus adéquats. Le serveur d'intégration Jenkins a été choisi, en parallèle du serveur d'analyse de code SonarQube. Afin de stocker l'ensemble des données, le choix de la base de données s'est porté sur PostgreSQL.

Dans une seconde partie, elle avait réalisé une machine virtuelle qui intégrait les outils d'intégration continue choisis. Cette machine était basée sur un système d'exploitation Windows 7 et comportait un serveur d'intégration continue Jenkins ainsi que SonarQube pour analyser le code source des projets. Cette première machine pouvait travailler avec des projets Java utilisant le moteur de production Maven ou avec

des projets C/C++ ou C# réalisés sous Visual Studio. Le choix du système d'exploitation Windows a été préféré à un système Linux puisque nous avons la nécessité de travailler avec des projets écrits en C#. En effet, afin de travailler avec ce langage dans les meilleures conditions, il est indispensable d'utiliser le logiciel Visual Studio, uniquement disponible sous Windows.

Afin de pouvoir se connecter à l'interface de Jenkins ainsi que SonarQube, nous utilisons l'Active Directory de l'Université. Grâce à cela, toute personne faisant partie de l'université aura la possibilité d'accéder au service. En effet, c'est la méthode la plus adaptée dans notre cas afin de gérer l'authentification des utilisateurs. En plus de cela, il est possible de préciser qui peut au sein de l'université se connecter ou non afin de pouvoir réduire les utilisateurs aux étudiants du département informatique ou informatique industrielle par exemple. Voici un schéma montrant les différents éléments majeurs du serveur d'intégration :

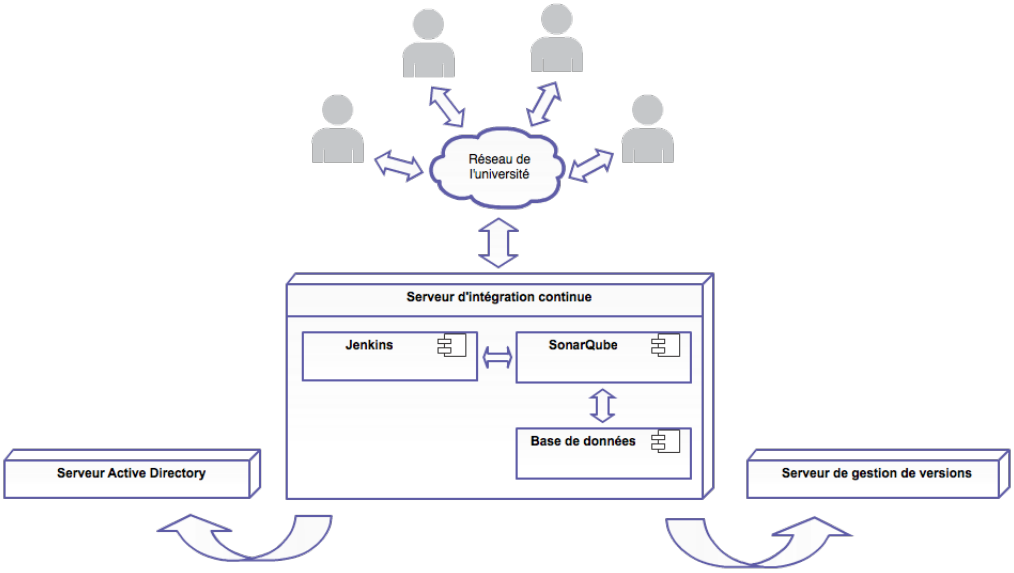

FIGURE 2.1 – Le serveur d'intégration continue

2.3 Description des tâches à réaliser

Dans le cadre de ce projet de fin d'études, il m'a été demandé de déployer une nouvelle machine virtuelle serveur en se basant sur le travail réalisé précédemment par Anne Castier. Cependant, je devais aller plus loin en permettant aux utilisateurs de travailler avec un plus grand nombre de technologies. Par exemple, il a été nécessaire d'intégrer de nouveaux langages comme le PHP ainsi que des moteurs de production tels que Ant ou CMake. Il a aussi fallu travailler sur la capacité de notre serveur à répondre aux sollicitations de ses utilisateurs et se focaliser sur l'administration du serveur pour qu'il puisse être maintenu facilement par la suite. Ces deux parties n'avaient pas été abordées par manque de temps au cours du projet précédent. De plus, notre serveur s'appuie sur un autre serveur mis en place par le service informatique. Ce serveur est muni d'un dépôt de code source Subversion pour versionner les projets au sein de Polytech' Tours.

Nous pouvons représenter le système mis en place de la manière suivante :

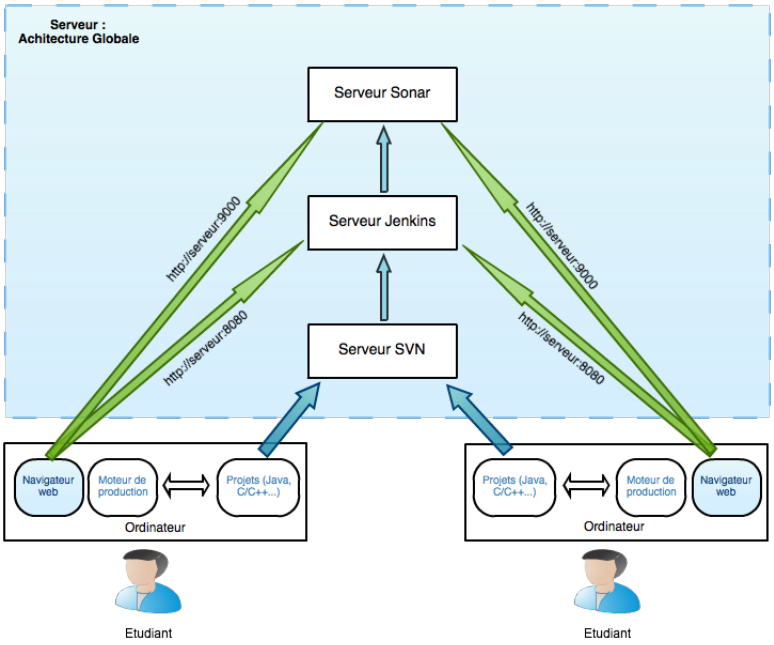

FIGURE 2.2 – Architecture de notre système

Comme nous pouvons le voir, les utilisateurs travaillent de manière classique sur leur projet en utilisant le serveur de gestion de versions Subversion mis à disposition par l'université. Notre serveur d'intégration continue va ensuite se greffer à ce projet. Les utilisateurs ont la possibilité d'ajouter leur projet via l'interface Web de Jenkins. A partir de là, de nombreuses possibilités s'offrent à eux comme compiler son code de manière automatique et périodique, lancer des tests et analyser le code source via un second outil qu'est SonarQube. Pour plus d'informations, je vous recommande de consulter le cahier de spécification fourni avec ce rapport.

Le travail réalisé

Au cours de cette partie je vais m'attacher à présenter le travail que j'ai accompli au cours de mon projet de fin d'études. Les résultats de celui-ci sont principalement constitués de la machine virtuelle serveur ainsi que de la documentation complète que j'ai pu rédiger tout au long du projet.

3.1 La prise en main de l'existant

Comme nous l'avons vu, ce projet de fin d'études faisant suite à un autre projet réalisé au cours de l'année universitaire 2013/2014. La première étape a été de reprendre l'ensemble des documents qu'avait réalisé Anne ainsi que de prendre en main les machines virtuelles mises en place lors de ce projet. Cette étape a été une étape majeure de mon projet dans le sens où Anne avait déjà rédigé une documentation conséquente en plus des machines virtuelles réalisées. Il a fallu comprendre l'ensemble des choix qui avaient été réalisés mais aussi être en mesure de déployer de nouveau les machines virtuelles produites pour utiliser le serveur d'intégration qu'elle avait mis en place. J'ai donc commencé par travailler sur l'ensemble des documents qu'elle avait rédigés. Ensuite, je me suis attaché à remettre en service les machines virtuelles pour que je puisse m'imprégner du fonctionnement de Jenkins ainsi que de SonarQube.

Cette étape n'a pas été la plus simple pour moi. En effet, je n'avais jusque-là jamais utilisé de serveur d'intégration continue mais j'avais déjà eu l'occasion de travailler avec l'outil SonarQube au cours de projets passés. Grâce à la très forte documentation laissée par Anne, j'ai pu réaliser ce travail de manière relativement aisée.

3.2 Les machines virtuelles mises en place

Le principal résultat de mon projet de fin d'études est représenté par une machine serveur permettant de mettre en place une démarche d'intégration continue ainsi qu'un ensemble de machines virtuelles clientes. Afin de soutenir la machine serveur, j'ai aussi mis en place un ensemble de machines esclaves visant à répartir la charge au maximum en parallélisant les tâches à réaliser.

3.2.1 La machine serveur

Le résultat le plus important de ce projet de fin d'études réside dans la machine virtuelle serveur que j'ai eu l'occasion de mettre en place. Cette machine virtuelle fonctionne sous l'environnement de virtualisation VMWare vSphere. Elle est directement implantée au cœur du réseau de l'université, dans le Data Center situé au Plat d'Etain. Cette localisation nous permet de simplifier grandement les actions de maintenance du serveur et ne nous oblige pas à allouer une machine directement au sein du département informatique, comme cela avait été réalisé l'année précédente. Ainsi, notre serveur est accessible de manière très simple depuis n'importe quel réseau de l'université. De plus, le fait de travailler avec une machine virtuelle nous permet de simplifier là encore la maintenance de ce dernier.

Le système d'exploitation que nous avons choisi d'utiliser est un Windows Server en version 2012. Ce système est entièrement conçu pour réaliser les tâches d'un serveur et nous convient donc parfaitement. Par rapport au projet de fin d'études précédent, nous avons modifié le choix du système d'exploitation

puisque nous utilisions jusque-là une machine Windows 7. Ce choix n'était pas optimal considérant les besoins que nous avions, Windows 7 n'étant pas conçu pour jouer un rôle de serveur. Une alternative à un système d'exploitation Windows aurait été d'utiliser un système Linux spécialisé pour les serveurs comme Debian Server par exemple. Ce choix a cependant été rejeté dès l'année dernière. En effet, le département de recherche travaille avec de nombreux projets écrits en C#. Ce langage est difficile à mettre en place sur une plateforme différente de Windows, voire impossible, ce qui nous a poussé à nous diriger vers un système basé sur Windows qui permet l'installation de ce langage très facilement via la suite logicielle Visual Studio.

Notre machine virtuelle est composée d'un grand nombre de logiciels et de bibliothèques de développement différents. Tout d'abord, et cela constitue la partie la plus importante, nous avons installé le logiciel Jenkins qui permet de mettre en place l'intégration continue ainsi que SonarQube qui permet d'effectuer des analyses du code source des projets. Pour pouvoir fonctionner dans des conditions idéales, il a été nécessaire de munir SonarQube d'une base de données. Le choix de la base de données s'est porté sur PostgreSQL. Ces choix avaient été faits lors du projet précédent. Ces trois éléments représentent les éléments fondamentaux de notre système d'intégration continue. De plus, l'utilisateur final n'aura accès qu'aux interfaces dédiées des deux logiciels Jenkins et SonarQube, via un navigateur web comme nous l'avons déjà abordé. Le serveur en lui-même ne pourra être accessible que par les administrateurs de la machine à des fins de maintenance et d'évolutions, le cas échéant.

Les autres logiciels que nous avons installés sur la machine virtuelle sont essentiellement liés aux technologies que nous souhaitions pouvoir supporter. Ainsi, pour pouvoir travailler avec des projets réalisés sous Visual Studio, qu'ils soient en C#, en C ou en C++, nous avons installé l'environnement Visual Studio de manière complète. Nous avons aussi installé le kit de développement Java ainsi que Maven afin de pouvoir intégrer les projets Java gérés à l'aide de ce moteur de production. Lors du projet précédent, ces deux environnements étaient les seuls à être installés. En plus de cela, nous avons cette année souhaité étendre les possibilités du serveur afin de le rendre plus polyvalent. Ainsi, nous permettons désormais le fonctionnement de projet écrits en C et C++ et utilisant les moteurs de production CMake ainsi que Make. La bibliothèque de développement C++ nommée Qt est aussi supportée avec son moteur de production QMake. En plus de pouvoir travailler avec des projets Maven, il est aussi désormais possible de travailler avec le moteur de production Apache Ant quelque soit le langage utilisé dans le projet, C ou Php. En effet, bien que le langage Php ne soit pas un langage compilé, il est tout de même possible d'exécuter une batterie de tests afin de vérifier le bon fonctionnement de ce dernier. L'ajout de ces technologies nous permet donc de pouvoir offrir un service plus large aux utilisateurs. En effet, l'une des principales contraintes de notre serveur a été qu'il doit pouvoir être utilisé par le plus grand nombre, et non pas une minorité d'étudiants ou d'enseignants-chercheurs. L'ensemble des composants que nous avions prévu d'installer lors de la phase de spécification du projet a été correctement intégré à notre serveur. Pour de plus amples informations sur cette partie, vous pouvez vous référer en cas de besoin aux différents manuels que j'ai pu rédiger ainsi qu'au cahier de spécifications défini lors de la première partie de ce projet. Tous ces documents sont fournis avec ce rapport.

Comme je l'avais défini lors de l'élaboration du cahier de spécification, la machine virtuelle est située dans le réseau de l'université. Afin de pouvoir être accessible, notre machine possède l'adresse IP 10.197.172.3 et elle ne change jamais. Afin d'accéder à l'interface de gestion de Jenkins, il est nécessaire d'effectuer une requête sur le port 8080 de cette machine via un navigateur web. Concernant SonarQube, c'est sur le port 9000 qu'il faut interroger notre serveur. La machine n'est accessible que depuis le réseau de l'université. Cela comprend le réseau filaire, le VPN ainsi que le réseau wifi lorsque la machine sera déplacée en production. En effet, au cours de mon projet la machine virtuelle était en état de pré-production, ce qui ne me permettait pas d'y accéder directement depuis le wifi. Cette restriction sera levée à la suite du projet puisque la machine va être déplacée en production. Des exemples de résultats que les utilisateurs peuvent obtenir en utilisant Jenkins et Sonarqube se trouvent en annexe. (Section 7.1, p. 26)

La principale difficulté de cette partie était de mettre en place l'ensemble des outils nécessaires au bon fonctionnement. En effet, le serveur d'intégration se devait de pouvoir faire fonctionner des projets réalisés avec de nombreuses technologies différentes. Une liste complète des composants a dû être mise en place en amont et il a été important de s'assurer que ces logiciels ne se perturbaient pas mutuellement. Il a ensuite été nécessaire de configurer correctement l'ensemble de ces outils afin de pouvoir les utiliser au sein de Jenkins. Bien que la documentation en ligne soit assez riche, tous les cas de figure ne sont pas forcément décrits et il a été parfois difficile d'intégrer certaines fonctionnalités. De plus, certaines limitations des outils que nous utilisons nous a forcé à supprimer certaines fonctionnalités. A titre d'exemple, nous souhaitions au départ permettre à l'utilisateur de lancer des analyses SonarQube sur leur projet sans devoir passer par Jenkins. Cela est possible avec un outil nommé SonarRunner. Le problème avec cet outil est qu'il est nécessaire de fournir du côté du client les accès à la base de données utilisée par SonarQube. Cela signifie donc qu'il faut donner à l'ensemble des utilisateurs le nom de l'utilisateur ainsi que le mot de passe nécessaire pour y accéder. Évidemment, nous avons immédiatement rejeté cette option et ainsi, si l'utilisateur souhaite lancer une analyse Sonar, il doit obligatoirement passer par Jenkins.

Au-delà de la simple mise en place de ce serveur, j'ai aussi travaillé sur l'administration de ce dernier. En effet, une fois le projet terminé, il sera nécessaire que le serveur soit simple d'administration pour l'équipe du service informatique. Des procédures ont dont été mises en place pour leur simplifier la tâche. De plus, des sauvegardes régulières ont été mises en place afin de permettre la restauration de la machine virtuelle en cas de panne. Ces sauvegardes sont effectuées par le biais de notre outil de virtualisation vSphere. Toutes les descriptions des opérations de maintenance du serveur sont précisées dans un document particulier : le manuel d'administration du serveur. Cette partie d'administration avait été complètement mise de côté l'année dernière par manque de temps. Le guide sera présenté dans une prochaine partie. (Section 3.3.3.2, p. 16)

3.2.2 Les machines virtuelles esclaves

Une fois la machine virtuelle serveur mise en place, nous avons effectué un ensemble de tests. Ces tests ont porté à la fois sur la vérification du bon fonctionnement des technologies installées sur le serveur mais aussi des performances de ce dernier. Ces tests sont décrits dans une prochaine partie. (Section 3.4, p. 16)

Tout d'abord, rappelons qu'au début de ce projet nous fonctionnons avec des clients multiples et un serveur unique afin de répondre à la demande de l'ensemble des utilisateurs de l'université. Nous pouvons donc représenter l'architecture de notre serveur de la façon suivante :

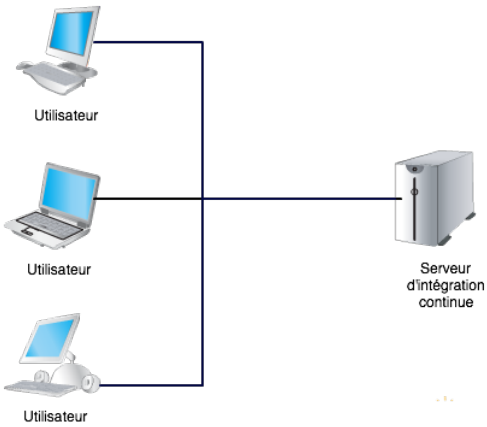

FIGURE 3.1 – Organisation originale du système

A la suite de ces tests, nous avons décidé de mettre en place des machines virtuelles esclaves afin de pouvoir soutenir notre serveur d'intégration continue. Les objectifs de ces machines esclaves étaient multiples. Tout d'abord, je souhaitais répartir la charge de travail sur un ensemble de machines plutôt que sur une machine unique. En effet, les tests de performance que j'ai réalisés n'avaient pas été concluants à mesure que le nombre de projets augmentaient sur le serveur. Il était donc impératif de trouver une solution. Le second point extrêmement intéressant dans la mise en place de machines esclaves et qu'il n'est pas obligatoire d'avoir un ensemble de machines homogènes. Par homogène, j'entends sur des systèmes d'exploitation différents. La possibilité de disposer de machines fonctionnant sous Linux et sous Windows par exemple nous permet donc d'ajouter la possibilité de pouvoir traiter des projets de différentes natures et fonctionnant sur des systèmes d'exploitation différents de Windows. En effet, jusqu'à présent les utilisateurs étaient limités à des projets fonctionnant sous Windows avec une liste de technologies assez restreinte. Le fait d'ajouter des machines Linux par exemple nous permet d'élargir les possibilités de notre serveur.

Au cours de ce projet, j'ai mis en place deux types de machines esclaves différentes. La première machine était une machine fonctionnant sous le système d'exploitation Windows. Sur celle-ci, j'ai décidé de mettre en place l'ensemble des outils et technologies installées sur le serveur maître. L'objectif visé est pour cette machine de seconder la machine principale en traitant exactement les même types de projets. L'intérêt d'avoir mis en place cette machine sous la forme de machine virtuelle est que nous sommes ensuite en mesure de la multiplier sur de nombreuses machines physiques. Ainsi, au lieu d'avoir une unique machine serveur, nous avons un serveur principal secondé entièrement par plusieurs machines esclaves. La charge de travail est ensuite répartie de manière optimale entre l'ensemble des esclaves.

La seconde machine que j'ai mise en place est une machine fonctionnant sous le système d'exploitation Linux, et plus précisément avec la distribution Ubuntu. Sur cette machine, j'ai mis en place un environnement de développement complet permettant de prendre en charge les projets C ou C++ mis en place ou non avec des makefiles. Ainsi, notre serveur d'intégration est en mesure de travailler avec des projets C ou C++ non seulement sous Windows, mais aussi sous Linux. En effet, certains projets fonctionnent uniquement sous un certain type de système d'exploitation et ne sont pas multi-plateformes. Ensuite, sur cette seconde machine j'ai mis en place les différents kits de développement Java nécessaire ainsi que les outils utiles à la mise en place de projets écrits en Php. L'ajout de ces technologies qui elles sont multi-plateformes permet de répartir une fois de plus la tâche du serveur maître sur un ensemble d'esclaves. Sur cette machine Linux, nous pouvons donc distinguer deux types de technologies : celles qui sont dépendantes de la plateformes et qui nous permet d'élargir les possibilités offertes par le serveur (C ou C++ par exemple), et les technologies indépendantes de la plateforme qui nous permettent de soutenir la charge du serveur principal (par exemple Java ou Php).

A la suite de la mise en place de ces machines virtuelles, la structure générale de notre système à quelque peu était modifiée. D'un point de vue utilisateur, il n'y a que de très légers changements. Comme nous l'avions vu, il doit toujours se connecter au serveur maître via une interface web. Ensuite, c'est Jenkins qui va se charger automatiquement de la répartition des projets sur les différents esclaves. Cette étape est complètement transparente du point de vue de l'utilisateur. Le nombre de machines esclaves n'est pas limité, il est possible d'en ajouter autant que vous le souhaitez, ou autant que vous le pouvez. Le fait d'avoir mis en place des machines virtuelles permet de simplifier grandement la multiplication de celles-ci.

Nous pouvons donc désormais représenter l'organisation de notre système de la façon suivante :

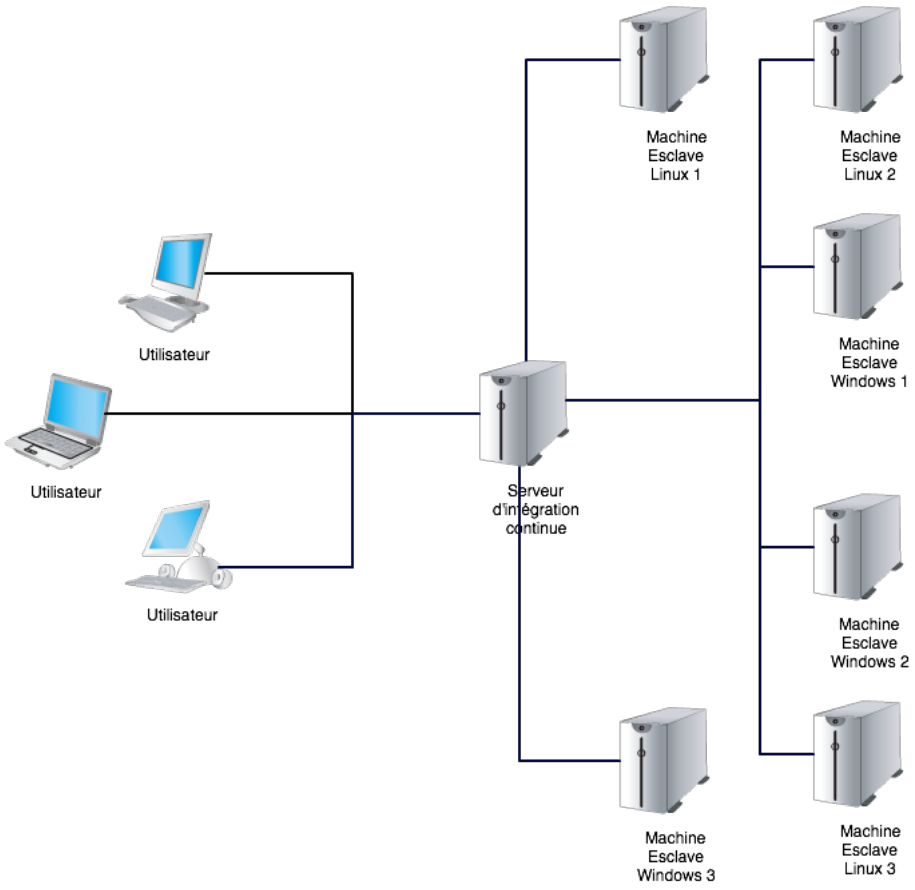

FIGURE 3.2 – Organisation finale du système

Malheureusement, je n'ai pas pu mettre en place de machines virtuelles MacOs. La mise en place d'esclaves fonctionnant sous MacOs nous permettrait de pouvoir intégrer des projets qui fonctionnent exclusivement sous cette plateforme, comme les projets écrits en Objective-C par exemple. Cependant, j'ai pu faire des essais avec ma propre machine et le test a été concluant, il est tout à fait possible d'ajouter des machines MacOs dans notre groupe de machines esclaves.

3.2.3 La machine cliente

Lors du projet de fin d'études précédent réalisé par Anne Castier, une machine virtuelle cliente avait été spécialement réalisée afin de pouvoir interagir avec le serveur d'intégration continue. Cette année, après avoir étudié l'ensemble des composants nécessaires à l'interfaçage de cette machine avec le serveur, nous avons choisi de ne pas mettre en place de machine dédiée, même virtuelle. En effet, les solutions que nous avons choisies d'installer sur notre serveur ne nécessitent pas des outils spécifiques afin de pouvoir être utilisées. L'ensemble des actions peuvent être réalisées simplement depuis un navigateur internet classique tel que Firefox ou Chrome. La décision que nous avons prise est donc que les utilisateurs peuvent, au choix, utiliser directement leur machine personnelle ou bien les machines virtuelles de développement mises à disposition par le service informatique.

Pour pouvoir travailler dans de bonnes conditions, il faut que la machine sur laquelle l'utilisateur travaille soit équipée d'un client Subversion afin de pouvoir interagir avec le serveur Subversion de l'université. Sous Windows, on peut par exemple utiliser le logiciel TortoiseSVN. Ensuite, il sera nécessaire d'avoir l'IDE

permettant de développer dans le langage utilisé comme par exemple Visual Studio pour des projets C#. Le dernier outil, comme nous l'avons vu, est simplement un navigateur web afin de pouvoir accéder aux interfaces de notre serveur. Si l'utilisateur souhaite simplement ajouter un projet dans le serveur d'intégration et que ce projet se situe déjà sur le serveur Subversion, alors un simple navigateur web suffit.

Dans cette situation, le seul intérêt de préparer une machine virtuelle aurait été d'installer le logiciel SonarRunner et de le configurer. Ce logiciel permet de réaliser une analyse de code SonarQube sans pour autant passer par Jenkins. Le problème avec ce logiciel est que pour fonctionner, il est obligatoire de fournir le mot de passe ainsi que l'identifiant de l'utilisateur qui gère la base de données SonarQube, ce qui n'était absolument pas envisageable dans notre cas de figure. Nous ne pouvons pas nous permettre de distribuer ces identifiants pour des raisons évidentes de sécurité. Nous avons donc abandonner cette possibilité.

Ce choix nous a permis d'éviter de passer du temps sur l'élaboration et la mise en place d'une machine virtuelle cliente qui n'était pas utile, les machines de développement classiques pouvant d'ores et déjà satisfaire nos besoins.

3.3 La documentation réalisée

Au cours de la mise en place des machines virtuelles, j'ai eu l'occasion de rédiger une large documentation permettant de retracer toutes les opérations que j'ai réalisées. Ce nombre de documents annexes explique en particulier la petite taille de ce rapport final. Grâce à cette documentation, il sera très facile à quiconque de pouvoir installer l'ensemble des éléments composant ces machines. Aussi, cela simplifiera l'administration de la machine par l'équipe du service informatique ainsi que l'utilisation du serveur pour les utilisateurs. Une documentation avait d'ores et déjà été réalisée par Anne l'année précédente, cependant nous avons souhaité la reprendre à partir de zéro et l'enrichir avec toutes les nouvelles fonctionnalités que le serveur offre désormais. De plus, certains de ces documents n'avaient pas été réalisés au cours du précédent projet. La rédaction de cette documentation n'a pas toujours été facile dans la mesure où il était nécessaire d'être le plus précis et exhaustif possible, ce qui peut parfois s'avérer complexe.

3.3.1 Le cahier de spécification

Au tout début du projet j'ai rédigé le cahier de spécification et le plan de développement de celui-ci. Ce document est un document essentiel à tous les projets de fin d'études. Il nous a permis de définir les objectifs que nous souhaitions atteindre dans le cadre de ce projet, de définir quelles sont les cibles visées par le serveur que nous allions mettre en place ainsi que les technologies que nous devions utiliser. Ce document a marqué la fin de la première partie du projet et a servi de base pour la soutenance de mi-parcours ainsi que toute la suite du projet.

3.3.2 Les manuels utilisateurs

Les premiers documents que j'ai mis en place constituent la documentation à destination des utilisateurs du serveur d'intégration continue. Cette documentation est découpée en deux parties. Le premier volet est le manuel utilisateur complet, le second constitue un guide de démarrage rapide.

3.3.2.1 Le guide utilisateur complet

L'objectif de ce document est d'accompagner le plus possible l'utilisateur lors de l'utilisation du serveur d'intégration continue. Ainsi, il va lui permettre de pouvoir créer un projet, l'ajouter sur le redmine de Polytech Tours et utiliser le serveur d'intégration continue. Ce document se veut être le plus exhaustif et le plus complet possible et décrit l'ensemble des possibilités offertes par le serveur ainsi que la façon de les mettre en place, à la fois au niveau de Jenkins que de SonarQube. Bien que certaines fonctionnalités

ne seront peut-être utilisées que par quelques utilisateurs, elles sont tout de même présentées. N'importe quel utilisateur sera alors en mesure grâce à ce manuel de pouvoir d'adopter une démarche d'intégration continue sur son projet, en considérant que les technologies utilisées soient bien celles disponibles sur le serveur. Pour rappel, il permet de gérer les projets basés sur les technologies suivantes :

- Les projets Visual Studio écrits en C/C++/C#.
- Les projets Java avec les moteurs de production Maven ou Ant.
- Les projets PHP. Le moteur de production Ant étant recommandé pour faciliter le lancement des tests.
- Les projets C/C++ utilisant des makefile ou le moteur CMake, avec ou sans la bibliothèque Qt.

A la fin du projet précédent, seuls les projets utilisant Visual Studio ainsi que les projets Java basés sur Maven pouvaient fonctionner. Vous pourrez trouver ce manuel dans un document séparé fourni avec ce rapport. L'utilisation de moteurs de production n'est pas obligatoire mais est tout de même fortement recommandée pour faciliter les opérations sur votre projet.

3.3.2.2 Le guide utilisateur rapide

En plus du manuel complet, nous avons aussi souhaité mettre à disposition des utilisateurs un guide leur permettant d'adopter une démarche d'intégration continue très rapidement en ne précisant que les étapes majeures et vitales au bon fonctionnement de leurs projets sur le serveur. Par conséquent, il est beaucoup plus court que le manuel utilisateur complet mais permet de mettre en place exactement les mêmes projets. L'objectif de ce manuel était de permettre aux utilisateurs ne souhaitant pas se servir de fonctionnalités avancées de tout de même pouvoir travailler avec le serveur d'intégration. Si par la suite ils désirent approfondir leurs connaissances, ils peuvent alors utiliser le manuel complet présenté précédemment.

Tout comme pour le manuel complet, ce document se trouve avec l'ensemble de la documentation que j'ai eu l'occasion de rédiger.

3.3.3 Les manuels administrateurs

Le second groupe de documents que j'ai eu l'occasion de rédiger ne concerne cette fois-ci plus les utilisateurs finaux du serveur mais ses administrateurs. Ils permettent de pouvoir maintenir et faire évoluer le serveur en cas de besoin. Ces documents permettent aussi de comprendre comment j'ai pu mettre en place l'ensemble des outils que nous utilisons et comment ils ont du être configurés.

Comme nous l'avons vu, contrairement à ce qui a été fait dans le précédent projet, nous n'avons pas mis en place de machine virtuelle cliente. Ainsi, nous n'avons pas mis non plus en place de manuel d'installation de cette machine, ses composants étant des logiciels classiques utilisés dans le développement d'applications. De plus, l'installation de la plupart de ces logiciels est décrite dans le manuel d'installation du serveur.

3.3.3.1 Le manuel d'installation serveur

La première étape de mon projet consistait à réinstaller l'ensemble des logiciels nécessaires sur la nouvelle machine virtuelle située à la DTIC [1]. Le fait de l'installer à la DTIC nous permet d'avoir notre serveur au cœur du réseau de l'université. Afin de l'installer je me suis aidé du manuel d'installation qu'avait réalisé Anne l'année passée. Cependant, nous avons souhaité rédiger un nouveau document qui se voulait un peu plus précis que celui-ci et surtout mis à jour. De plus, j'ai installé de nombreux composants supplémentaires qui n'étaient pas du tout installés sur le premier serveur. Ils figurent donc maintenant dans ce nouveau

1. Direction des Technologies de l'Information et de la Communication

manuel.

Ce document permet de retracer l'installation de tous les logiciels qui composent notre serveur ainsi que tous leurs pré-requis. Grâce à celui-ci, les utilisateurs seraient en mesure de pouvoir réinstaller le serveur à partir d'une machine vierge sans rencontrer de grandes difficultés. En effet, certaines configurations des logiciels peuvent s'avérer complexes à mettre en œuvre et le document les précise et explique toutes.

En plus de la description complète de la démarche pour installer la machine virtuelle serveur, j'ai aussi précisé comment installer, déployer et activer les machines virtuelles esclaves à la fois sous le système d'exploitation Windows ainsi que Linux.

3.3.3.2 Le manuel d'administration

Une fois l'installation de la machine virtuelle serveur réalisée, il est nécessaire d'effectuer certaines opérations de maintenance sur notre serveur. Toutes ces opérations sont listées et expliquées au sein d'un manuel spécial que j'ai mis en place : le manuel d'administration. Parmi ces opérations nous trouvons par exemple :

- Mise à jour des différents logiciels utilisés.
- Administration des serveurs SonarQube et Jenkins ainsi que la base de données.
- Gestion des utilisateurs de notre serveur.
- Administration et gestion des projets hébergés.
- Gestion des machines esclaves.
- Ajout de nouvelles fonctionnalités.

Ce document a été réalisé dans le but de faciliter la tâche aux futurs administrateurs du serveur. Cela leur permettra d'éviter de passer de longues heures à trouver comment effectuer les opérations de maintenance souhaitées.

3.4 Les tests mis en place

3.4.1 La démarche de test

Une fois la machine virtuelle installée complètement, nous sommes passés dans une phase du projet qui consistait à effectuer des tests. Ces tests étaient de différentes natures.

Le premier type de test était relativement simple. Il consiste à vérifier que notre serveur d'intégration Jenkins était bien capable de traiter toutes les technologies que nous avions dans nos objectifs. Sans les rappeler toutes, nous avons par exemple vérifié que nous parvenions à compiler et à exécuter les tests correctement sur des projets construits à l'aide de Visual Studio ou des projets Java avec Maven ou Ant. Pour cela, j'ai toujours opéré de la même façon. Dans un premier temps, j'ai commencé par construire un projet basé sur la technologie que nous souhaitions tester. A titre d'exemple, j'ai créé un projet C# grâce à Visual Studio. Ensuite, j'ai exporté le projet dans un dépôt Subversion de l'université afin qu'il soit accessible depuis notre serveur. Enfin, j'ai mis en place un job au sein de Jenkins capable de compiler ce projet et lancer les tests unitaires. Si ces étapes ne fonctionnaient pas, je corrigeais alors la configuration pour être capable de traiter ce projet. Une fois le projet fonctionnel, je relançais ce même test une fois de plus cette technologie avec un autre projet afin de confirmer le bon déroulement du processus. A la suite de ces tests, je passais à la technologie suivante. Cette étape était la première dans la phase de tests.

Dans un second temps, nous avons tenté de trouver des personnes susceptibles d'être intéressés par l'intégration de leur projet au sein de notre serveur pour effectuer des tests sur des projets "réels", et non

plus des projets simplistes comme jusqu'à présent. Pour cela, nous avons diffuser une annonce auprès des étudiants de 4eet de 5eannée du département informatique. En effet, ils étaient les plus susceptibles de répondre à notre annonce dans le cadre du projet d'ingénierie collective ou du projet de fin d'études. Nous avons reçu quelque retours positifs, ce qui nous a permis de tester le serveur sur leur projet, et ainsi de leur permettre de bénéficier des services offerts par nos serveurs. Malheureusement, je n'ai pas reçus beaucoup de réponses, ce qui ne nous a pas permis de tester l'ensemble des possibilités du serveur. Au-delà du fait que ces tests nous permettaient de vérifier le fonctionnement correct de notre serveur ainsi que des outils que nous avions mis en place, cela m'a permis de valider les différents manuels qui avaient été réalisés. En effet, pour intégrer les projets des étudiants, je leur avais simplement fourni le manuel utilisateur et les avons laissé travailler avec. Cet étape a permis de pouvoir peaufiner ces manuels en clarifiant certains passages qui ne paraissaient pas clairs pour des personnes extérieures au projet. Cet étape a été primordial puisque je n'arrivais pas toujours à savoir si les instructions étaient claires ou non, travaillant sur le projet depuis désormais plusieurs mois. Concernant les manuels d'installation et d'administration, c'est Monsieur Meichel qui s'est chargé de la relecture ainsi que de la validation avec l'aide de Monsieur Ragot. Là aussi, cette avis extérieur m'a été précieux afin de lever les ambiguïtés qui avaient pu se glisser dans les documents.

Dans une troisième partie, je me suis attaché à effectuer des tests de montés en charge du serveur afin de voir s'il était capable de soutenir des pics d'activités. Pour rappel, notre machine possède pour l'instant 16Go de RAM ainsi qu'un processeur Intel Xeon cadencé à 2,40 GHz. Ces caractéristiques pourraient être revus à la hausse en cas de besoin. Les tests sont importants puisqu'il est possible que le serveur soit utilisé par vagues, notamment lors de séances travaux pratiques par exemple. Pour ce faire, j'ai simulé l'activité que peut produire ce genre de conditions en demandant à plusieurs de mes camarades de travailler sur leur projet. En parallèle, j'ai rajouté un grand nombre de projets sur le serveur en multipliant certains. Les résultats de ce tests sur Jenkins n'ont toujours été probants, notre machine ne parvenait pas toujours à travailler simultanément sur les projets. En ce qui concerne l'utilisation de SonarQube, celle-ci est un peu plus problématique dans la mesure où il n'est capable de lancer que quelques analyses simultanément. C'est à la suite de cette phase que j'ai décidé de mettre en place les machines virtuelles esclaves décrites dans les parties précédentes afin de pouvoir aider le serveur en cas de forte monté en charge. Enfin, il a fallu veiller à l'occupation mémoire de l'ensemble de ces projets. Pour cela notre machine virtuelle possède une partition dédiée entièrement au stockage des projets qui mesure 1 To. Pour l'instant, cet espace est très largement suffisant pour accueillir l'ensemble des projets des étudiants. Cependant, si un jour il est nécessaire d'augmenter cet espace, il sera très facile d'allouer plus de mémoire en demandant à la DTIC d'augmenter l'espace disque de notre machine. De plus, la fait d'utiliser des machines esclaves permet là aussi de répartir un peu plus le volume des données. L'ajout de machines esclaves à permis de résoudre la quasi-totalité des problèmes que nous avons rencontrés. L'unique problème subsistant concerne SonarQube qui ne permet pas encore de mettre en place le même système que Jenkins. Il existe cependant une possibilité qui sera décrite une prochaine partie. (Section 4.3, p. 21)

3.4.2 Les projets utilisés

Comme nous l'avons vu dans la partie précédente, j'ai été amené à utiliser de nombreux projets afin de tester le bon fonctionnement du serveur d'intégration continue lors de la période de tests. Les premiers projets que j'ai utilisés sont des projets très simplistes composés de quelques lignes de code ainsi que quelques tests unitaires. Par exemple, j'ai mis en place un programme ressemblant à une calculatrice, possédant plusieurs tests unitaires afin de vérifier l'exactitude des calculs. J'ai réalisé cette calculatrice dans la plupart des langages supportés par le serveur. J'ai aussi utilisés les programmes réalisés par Anne l'année dernière pour effectuer ses propres tests afin de gagner du temps. Ces projets n'ont donc que très peu d'intérêts du point de vue de l'analyse du code source, ils ne sont utiles qu'à valider le fonctionnement de Jenkins et de SonarQube. Ils seront fournis avec le projet comme nous le verrons dans une prochaine partie. (Section 3.5, p. 18)

Ensuite, j'ai indirectement pu utiliser les projets de fin d'études de plusieurs de mes camarades. Ainsi, Bastien Meunier, dans le cadre de son projet nommé "Recherche par mot-clés "image" dans des masses d'images : méthode", a été l'un des premiers à ajouter son projet sur le serveur d'intégration. Ensuite, Loreen Lambin a pu aussi ajouter son projet sur la plateforme avec son projet : "Recherche par mot-clés "image" dans des masses d'images : plateforme". Ils étaient tous les deux encadrés par Mr Ragot. En plus de tester le fonctionnement de Jenkins, ils ont aussi permis de faire des tests complets de SonarQube puisque cette fois-ci les projets possédaient un réel intérêt ainsi que sur la documentation que j'avais mise en place. Armand Renaudeau a aussi souhaité participer à notre test pour son projet. Malheureusement, il utilise majoritairement la table Microsoft Pixel Sense, et cela nécessite l'installation d'un SDK particulier pour pouvoir le compiler. Nous n'avons pas jugé pertinent d'installer ce SDK dans son intégralité pour seulement un projet. Enfin, le projet de Valentin Montmirail portant sur les problèmes SAT m'a permis de pouvoir vérifier le support de projets multi-plateforme du serveur puisqu'il travaille exclusivement dans un environnement Linux. D'autres étudiants de 4^{e}ont aussi participé à ces tests dans le cadre de leur projet d'ingénierie collective.

Enfin, j'ai utilisé aussi plusieurs projets open source que j'ai pu trouver sur internet. Par exemple, le projet Php que j'utilise pour effectuer mes tests sur ce langage et le moteur de production Ant est issu du plateforme de partage de projets. Il permet de pouvoir convertir de nombreuses devises entre-elles.

3.5 Les livrables du projets

Au cours de ce projet, j'ai été amené à fournir de nombreux livrables afin que mon travail puisse être facilement pris en main par les futurs administrateurs ainsi que les utilisateurs du serveur d'intégration. Nous pourrions classer ces livrables en deux grandes catégories : les machines virtuelles développées ainsi que la documentation fournie. Dans cette partie, je vais m'attacher à résumer l'ensemble de ces livrables.

Comme nous venons de le voir, le premier ensemble de livrables consiste à un ensemble de machines virtuelles :

1. La machines virtuelle serveur sous Windows Server 2012 composée de :
   - Jenkins permettant l'intégration continue.
   - SonarQube réalisant l'analyse de code des différents projets.
   - Une base de données PostgreSQL pour stocker les informations relatives à SonarQube.
   - L'ensemble des outils de développement permettant la compilation et l'exécution des tests dans les différentes technologies. Par exemple les compilateurs fournis dans Visual Studio, le kit de développement Java, etc...
2. La machine virtuelle esclave fonctionnant sous Windows. Elle est composée de :
   - Le kit de développement Java permettant de lancer le client Jenkins servant à la liaison entre la machine et le serveur maître.
   - L'ensemble des outils de développement permettant de traiter les projets fonctionnant sous Windows, tout comme la machine maître.
3. La machine virtuelle esclave fonctionnant sous Linux. Elle est composée de :
   - Tout comme pour l'esclave Windows, les outils Java permettant de démarrer le client Jenkins.
   - Les outils de développement permettant de travailler avec des projets C/C++ fonctionnant sous Linux.
4. Les machines virtuelles clientes qui sont celles que le service informatique a mis à disposition aux étudiants.

Ensuite je fournis l'ensemble de la documentation décrite au travers des parties précédentes, à savoir :

1. Le cahier de spécification réalisé lors de la première partie de ce projet qui le décrit entièrement ainsi que ses objectifs.
2. Le manuel d'installation de la machine virtuelle serveur. Il permet aussi de décrire la création et la mise en place des différentes machines esclaves.
3. Le manuel d'administration permettant de pouvoir réaliser de nombreuses tâches d'administration et de maintenance sur le serveur. Il permet aussi de pouvoir le faire évoluer.
4. Le manuel utilisateur complet permettant aux utilisateurs de prendre en main complètement l'environnement fourni par le serveur d'intégration.
5. Le manuel de prise en main rapide qui se focalise sur la démarche minimale afin de mettre un projet en place sur le serveur d'intégration continue.

Comme je l'ai dit tout au long de ce rapport, tous ces manuels et documents sont fournis avec ce rapport final. Je n'ai pas souhaité les ajouter en annexe à celui-ci afin de ne pas le surcharger inutilement.

L'avenir du serveur

Au sein de cette partie, je vais traiter de ce que va devenir le serveur une fois que ce projet sera terminé.

4.1 Mise en production et déploiement

Tout au long de ce projet, j'ai travaillé sur la machine virtuelle dans un environnement de pré-production. Lorsque le serveur sera utilisé par les étudiants, elle sera déplacée du statut de l'environnement de pré-production à celui de production. Ce déplacement se fait très rapidement grâce à l'outil de virtualisation vSphere. Dans l'absolu, ce déplacement ne changera que très peu de chose, il n'y aura quasiment aucune différence avec l'utilisation qui en est faite à l'heure actuelle. Cela apportera néanmoins un léger avantage puisque le serveur sera directement accessible depuis n'importe quelle machine, qu'elle soit connectée de manière filaire ou sur le wifi de l'université, sans devoir passer par le VPN de l'école comme c'est le cas pour le moment avec le wifi. En revanche, si vous n'êtes pas sur le réseau de l'école, vous devrez tout de même utiliser le VPN. Jusque-là, je ne considérais que pour la machine maître. Concernant les machines esclaves, il faudra que le service informatique décide de la démarche à adopter. Il est possible de mettre en place des machines virtuelles dédiées extrêmement rapidement afin de pouvoir répartir la charge. L'ensemble des étapes pour déployer ces machines esclaves sont détaillées au sein du manuel d'installation du serveur.

4.2 La maintenance du serveur

Après avoir mis le serveur en production, un nombre assez restreint d'actions manuelles seront nécessaires afin de le maintenir en fonction. En effet, beaucoup d'outils de surveillance sont à disposition et permettent d'éviter de devoir aller vérifier directement si l'espace disponible sur le serveur est encore suffisant. Évidemment, il est nécessaire de fournir les renseignements concernant le SMTP pour permettre l'envoi automatique de données en cas de problème. En effet, nous ne disposons pour l'instant pas d'accès SMTP, ce qui empêche le serveur d'envoyer les informations par mails. Cependant, ces informations sont directement accessibles depuis le panneau d'administration du serveur.

Malgré le nombre de processus automatisés, il sera quand même nécessaire d'avoir des actions régulières de la part des administrateurs. La première sera, tout comme cela est déjà fait pour le serveur Subversion, de supprimer régulièrement les projets qui ne sont plus utiles en fin d'année. Par exemple, il est possible de supprimer les projets relatifs à des séances de travaux pratiques ou à des mini-projets. A l'inverse, il faudra veiller à conserver les projets qui sont réalisés sur plusieurs années, comme les projets de fin d'études par exemple. Ce tri permettra de pouvoir faire de la place sur le disque des machines.

En plus de cela, le service informatique devra veiller à ce que le serveur fonctionne toujours correctement. En cas de besoin, il sera peut être nécessaire d'augmenter l'espace disque par exemple ou bien d'augmenter le nombre de machines esclaves. Cela pourra se faire rapidement grâce à l'utilisation des machines virtuelles que j'ai mises en place à la fin de ce projet. La démarche pour rajouter une machine est très rapide, elle est décrite dans le manuel d'administration que vous pourrez retrouver avec les autres documents fournis avec ce rapport.

4.3 Les évolutions potentielles

Comme nous venons de le voir, une des évolutions potentielles du serveur serait de déployer de nouvelles machines esclaves afin de pouvoir traiter un plus grand nombre de projets simultanément. Cependant, ce n'est pas vraiment une réelle évolution en soit puisque cela consiste simplement à renforcer le serveur. Il serait très intéressant d'être capable de déployer les machines virtuelles esclaves de manières entièrement automatique. Par automatique, j'entends tout d'abord le déploiement de la machine et sa configuration avec la machine maître. Cette opération peut être possible en utilisant notamment des scripts de configuration et en faisant transiter la copie de la machine virtuelle esclave via le réseau. Au-delà du simple déploiement automatique, nous pourrions imaginer installer nos machines esclaves sur l'ensemble des machines du département pour transformer ces machines en esclaves. Bien évidemment, de nombreux étudiants utilisent ces machines tous les jours. Cependant, les machines ne sont pas utilisées en permanence, et encore moins au maximum de leurs capacités. Nous pourrions donc concevoir un système qui active les machines esclaves lorsque l'ordinateur hôte n'est pas ou peu utilisé, et le désactiver dans le cas contraire. En l'occurrence, Christophe Forycki travaille sur un système similaire dans le cadre de son projet de fin d'études : "Grappillage de ressources". Nous pourrions alors avoir à disposition un grand nombre de machines esclaves sans pour autant immobiliser une machine en particulier.

Une réelle évolution consisterait à augmenter le nombre de langages supportés par le serveur, comme je l'ai fait au cours de ce projet. Cependant, nous avons d'ores et déjà tous les langages les plus utilisés à Polytech Tours. L'ajout de langages serait alors limité à des langages moins utilisés par les étudiants. Il faudra donc savoir si l'ajout d'un langage est réellement utile ou si l'effort pour l'ajouter n'est pas trop important par rapport à son intérêt. Grâce à l'utilisation de machines esclaves, nous ne sommes plus limités à un système d'exploitation précis, ce qui offre beaucoup plus de possibilités. Il est par exemple possible de rajouter des machines fonctionnant sous MacOS afin de pouvoir travailler avec des projets Objective-C notamment.

Au delà du simple rajout de nouveaux langages, il est aussi possible de rajouter le support de moteurs de production. Cependant, j'ai pu intégré les moteurs les plus utilisés que sont Maven et Ant. Cette perspective est donc assez limitée.

Une avancée importante serait d'intégrer SonarQube au niveau des machines esclaves afin de répartir le travail sur celles-ci. Pour l'instant, il n'est pas possible de déployer de telles machines comme nous l'avons fait avec Jenkins. La seule option serait d'installer de nouvelles instances de SonarQube et de les relier toute à la même base de données située sur notre serveur. Le principal problème de cette solution serait que toutes les instances seraient vues par Jenkins comme des serveurs différents et il ne serait pas capable d'effectuer la répartition du travail lui-même comme le fait Jenkins. Il serait intéressant de surveiller l'actualité du développement de SonarQube pour savoir si un jour une telle option serait ajoutée, ce qui aiderait grandement notre serveur. Les machines esclaves Jenkins pourraient alors servir en même temps de machine esclave pour SonarQube et le serveur maître serait en charge de la répartition de la charge. Augmenter la capacité de traitement du serveur serait alors une avancée simple et peu coûteuse, autant pour Jenkins que pour SonarQube. Comme c'est déjà le cas, le déploiement d'une nouvelle machine virtuelle serait extrêmement rapide.

Le déroulement du projet

Dans cette partie, je parlerai de la réalisation du projet d'un point de vue beaucoup plus général et me focaliserai sur comment il s'est déroulé ainsi que sur ma manière de l'aborder et de réaliser les différentes tâches qui m'étaient confiées.

5.1 L'organisation du projet

Afin de mener à bien ce projet, j'ai dès le début défini l'ensemble des tâches que je devais réaliser pour mettre en place le serveur d'intégration continue. Cette partie n'a pas été facile à faire dans la mesure où je n'avais encore jamais travaillé sur un projet aussi conséquent. Avec l'aide de mes encadrants, j'ai réussi à recenser l'ensemble des tâches et à mettre au point un planning pour ce projet. Je présenterai le planning du projet dans la partie suivante (Section 5.2, p. 22). Pour pouvoir réaliser l'ensemble des tâches que je venais de définir, je me suis en grande partie appuyer sur le redmine de l'université. En effet, le redmine est un outil très intéressant du point de vue de la gestion de projet. Il m'a permis de saisir l'ensemble des tâches que je réalisais tout au long de ce projet. Grâce à lui, j'ai pu savoir à tout moment si j'étais en retard ou non sur ce qui avait été prévu au début de l'année. En plus de cela, le redmine me permettait d'informer en temps réel mes encadrants sur l'avancée du projet. En effet, nous étions inscrits sur le projet du redmine, ce qui permettait d'avoir des alertes à mesure que je renseignais des informations. En plus de ce suivi indirect de mon travail, j'essayais au maximum d'avoir des points réguliers avec Monsieur Ragot afin de lui présenter de vive voix le résultat de mon travail. Cela permettait de pouvoir éclaircir certains points sur le projet et me permettait aussi d'avoir un retour sur ce que je venais de réaliser. Cela nous permettait aussi de pouvoir prendre des décisions lorsque certains problèmes étaient soulevés. En plus de ces points réguliers, mes encadrants étaient aussi présents afin de relire la documentation que je rédigeais au fur et à mesure du projet. Le fait qu'ils relisent ce que je rédigeais me permettais d'avoir un avis "extérieur" sur mon travail et ainsi de pouvoir modifier certains points qui n'étaient pas toujours clairs. En effet, il est assez difficile de percevoir ce qui a besoin d'être clarifier lorsque nous travaillons sur un document depuis plusieurs semaines. Cet aspect est d'autant plus important que les manuels rédigés sont destinés à des utilisateurs n'ayant, pour certains, jamais utilisés les outils que j'ai mis en place.

En plus de l'organisation interne de ce projet avec mes encadrants, il a aussi été nécessaire de m'organiser avec certains de mes camarades. En effet, afin de vérifier le bon fonctionnement du serveur d'intégration, j'ai sollicité leur aide pour obtenir un plus grand nombre d'avis sur mon travail. Cela n'a pas toujours été facile car cela les obligeait à prendre du temps sur mes manuels. De plus, je devais être présent afin de répondre à leurs interrogations s'ils en avaient, il fallait donc que nous trouvions des créneaux sur lesquels nous pouvions travailler ensemble sur cela.

5.2 La planification du projet

Comme je l'expliquais dans la partie précédente, j'ai mis au point au tout début de ce projet la liste des tâches à réaliser pour le mener à bien, ainsi qu'un planning prévisionnel. Cette tâche a été particulièrement difficile pour moi puisque je n'avais encore jamais travaillé sur ce type de projet. En effet, j'avais jusque-là uniquement travaillé sur des projets de développement. Il a donc été difficile d'estimer le temps que devraient me prendre les différentes tâches. Malgré cela, j'ai réussi à réaliser un estimation avec l'aide de

mes encadrants. Voici un représentation de la liste des tâches que je devais réaliser. Ce tableau est issu de mon cahier de spécification.

#	Titre	Travail Planifié Donné	Drapeau	# Prédécesseurs	Date Début Attendue
0	**Projet de Fin d'Etude – Serveur d'intégration continue**				18/09/2014
1	Découverte et prise en main des outils	3 jours			18/09/2014
2	Rédaction du cahier de spécification système et poster	7 jours		6	13/11/2014
3	**Mise en place de la machine virtuelle serveur**				09/10/2014
4	Installation des composants du serveur	1 jour		1	09/10/2014
5	Configuration des outils du serveur	1 jour		4	16/10/2014
6	Rédaction du manuel d'installation serveur	2 jours		5	23/10/2014
7	Rédaction du manuel d'administration	3 jours		2	13/01/2015
8	**Mise en place de la machine virtuelle client**				20/01/2015
9	Installation des composants du client	1 jour		7	20/01/2015
10	Rédaction du manuel d'installation client	3 jours		9	21/01/2015
11	Rédaction du manuel utilisateur complet	4 jours		10	28/01/2015
12	Rédaction du manuel de démarrage rapide	3 jours		11	05/02/2015
13	Tests de fonctionnement en condition réelles	15 jours		12	12/02/2015
14	Mise en place des outils de surveillance	6 jours		13	26/03/2015
15	Rédaction du rapport final du projet	6 jours		14	09/04/2015
16	**Ajout de fonctionnalités**				01/04/2015
17	Ajout du support de nouveaux langages	3 jours		13	01/04/2015
18	Ajout du support de nouveaux moteurs de production	3 jours		17	08/04/2015
19	**Mise en place de la plateforme pédagogique**				15/04/2015
20	Création des projets types	2 jours		18	15/04/2015
21	Rédaction du sujet de TP	3 jours		20	21/04/2015

FIGURE 5.1 – Liste des tâches à réaliser

Afin de mieux visualiser l'agencement des différentes tâches, voici le diagramme de Gantt associé à ce tableau :

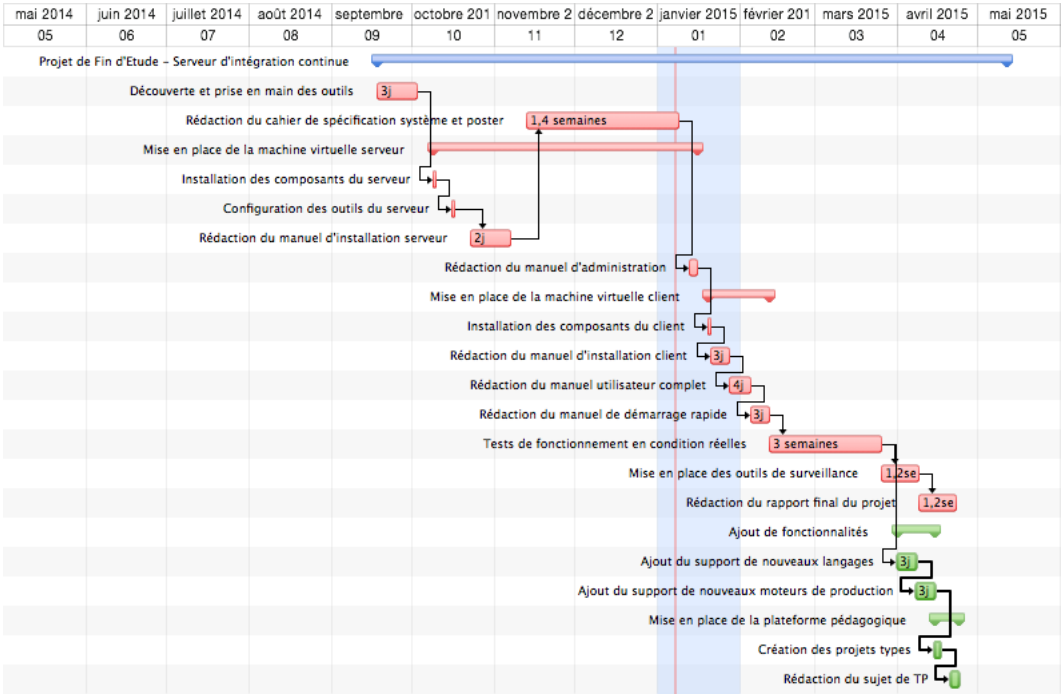

FIGURE 5.2 – Diagramme de Gantt prévisionnel du projet

Nous pouvions découper notre projet en plusieurs grandes phases :

1. Étude de l'existant et rédaction des spécifications du projets.
2. Mise en place de la machine serveur.
3. Mise en place des machines clientes. Après étude, cette partie a été supprimées.
4. Rédaction des différents manuels pour les administrateurs et les utilisateurs.
5. Tests sur le serveur.

De manière générale, je suis parvenu à suivre le planning que nous nous étions fixés. Bien évidemment, quelques ajustements ont du être réalisés au cours du projet. Par exemple, la mise en place de machines clientes a été annulée. Comme je l'ai déjà évoqué, les machines de l'université peuvent tout à fait remplir ce besoin. Cela m'a permis d'avoir plus de temps pour travailler sur la partie concernant l'administration du serveur ainsi que les tests de bon fonctionnement. Sans cela, je n'aurais pas eu le temps de mettre en place les machines esclaves, que je considère comme vitales pour le serveur d'intégration continue. Sans elles, le serveur ne sera pas en mesure de gérer l'ensemble des tâches qui lui sont confiées.

Concernant les tâches relatives aux manuels, j'ai du m'adapter aussi. En effet, j'ai commencé par rédiger le document. Une fois ce document rédigé, mes encadrants le relisaient et me le rendaient afin que je puisse y apporter les modifications nécessaires. Ce mode de fonctionnement m'a donc obligé à effectuer des retours en arrière pour retravailler les différents manuels, cependant cette méthode de travail a été très importante pour obtenir les documents les plus clairs possibles. J'ai utilisé la même méthode pour mettre au point les manuels utilisateurs avec mes camarades.

Me restant un peu de temps sur la fin du projet, j'ai pu, en plus de mettre en place le serveur d'intégration continue, ajouter différentes fonctionnalités ainsi que le support de nouveaux langages et moteurs de production. Cette tâche n'avait pas été considérée comme prioritaire lors de la rédaction des spécifications. Il était en effet primordial que le serveur soit opérationnel avant la fin du projet pour pouvoir être utilisé au plus tôt. Cependant, il est aussi important que notre serveur puisse être polyvalent afin de pouvoir en faire bénéficier le plus grand nombre, ce qui a pu être réalisé.

Au début du projet, j'avais proposé de mettre au point des tutoriels basés sur un ensemble de projets préparés. Ces documents auraient permis aux étudiants de prendre en main le serveur d'intégration sans devoir nécessairement avoir un projet. Malheureusement, je n'ai pas eu l'occasion de rédiger ces documents par manque de temps à la fin du projet. En effet, il était plus important de se focaliser sur le serveur en lui-même. Pour compenser ce manque, rien n'empêche les utilisateurs de se servir des deux manuels utilisateurs que j'ai rédigé et de travailler sur un petit projet qu'ils auront créé rapidement pour pouvoir prendre en main les possibilités offertes par le serveur. Ils auront aussi à leur disposition les projets de tests que j'ai moi-même utilisé au cours du projet.

Le bilan du projet

Pour conclure ce rapport, je pense pouvoir dire que j'ai réussi à atteindre les différents objectifs que nous nous étions fixés dans le cadre de ce projet de fin d'études intitulé "Mise en place d'un serveur d'intégration continue et de qualité de code". Bien que je n'ai pas eu le temps de travailler sur certains points définis comme non-essentiels, tous les objectifs majeurs du projet ont pour moi été atteints. En effet, le serveur d'intégration continue est désormais déployé au sein du département informatique et il est prêt à être mis en production. De plus, l'ensemble de la documentation nécessaire à la fois aux utilisateurs et aux administrateurs a été rédigée. Même si tout n'est pas encore parfait, ils seront en mesure de prendre en main cet outil.

Réaliser ce projet n'aura pas toujours été facile pour moi. En effet, je n'avais jusque-là encore jamais travaillé sur ce type de projet. Je suis beaucoup plus habitué aux projets de développement qu'à ceux se rapprochant de l'administration de système. Malgré cela, je pense avoir tout de même réussi à m'adapter et à mettre en œuvre mes compétences pour le mener à bien. De plus, cela m'a permis d'en apprendre plus sur ce domaine, ce qui me permet d'avoir une meilleure connaissance de ce domaine de l'informatique.

Enfin, je suis désormais en mesure de travailler avec des environnements d'intégration continue ainsi que de qualité de code. Je n'avais jusque-là presque jamais eu l'occasion de travail avec un serveur d'intégration continue. Ces connaissances me seront très utiles dans le futurs dans la mesure où toutes les entreprises utilisent ce mode de fonctionnement pour leurs projets.

Annexes

Dans cette partie d'annexes, je vais présenter quelques résultats que l'on peut obtenir avec Jenkins ou SonarQube.

7.1 Exemple de résultats avec Jenkins

Voici un exemple de ce que l'on obtient lorsque l'utilisateur travaille avec Jenkins sur son projet. Cette page est la page d'accueil du projet :

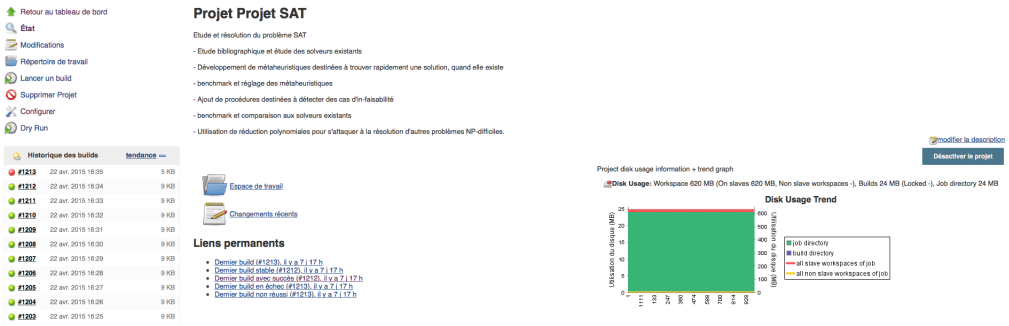

FIGURE 7.1 – Accueil du projet dans Jenkins

Il est aussi possible d'avoir l'historique des builds du projet sous Jenkins. Une bulle verte signifie que la construction du projet s'est déroulée correctement, une bulle rouge signifie qu'il y a eu un problème :

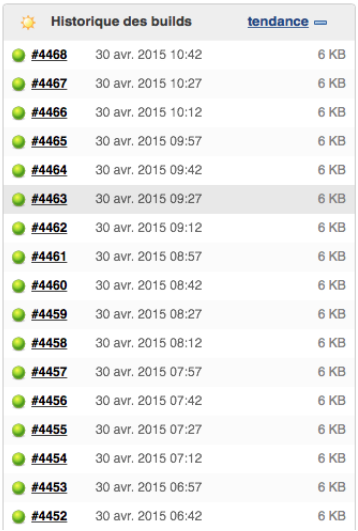

FIGURE 7.2 – Historique du projet

Dans le cas où un problème est survenu, il est possible d'avoir le détail de l'ensemble des actions

effectuées par le serveur dans la console, comme ceci :

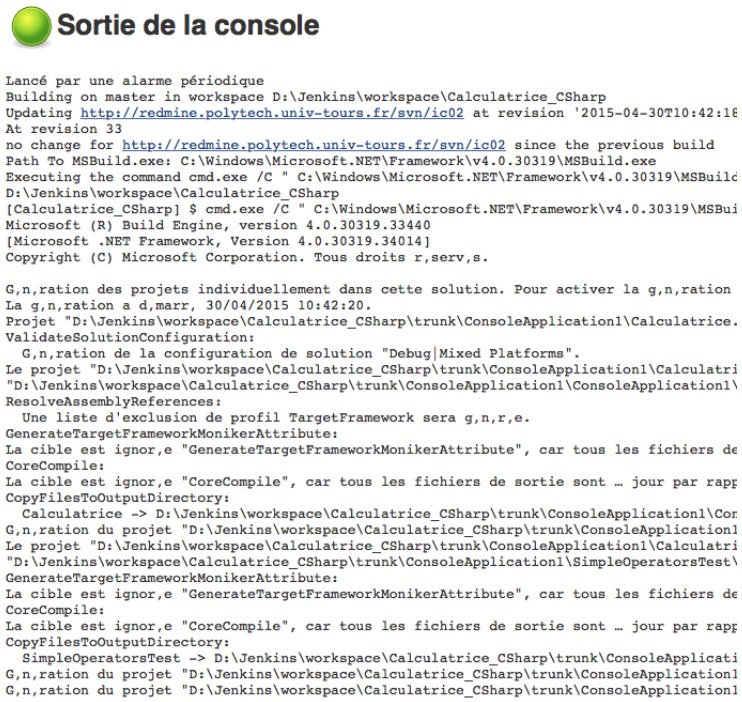

FIGURE 7.3 – Sortie console d'un build

.

7.2 Exemple de rapport SonarQube

Tout comme pour Jenkins précédemment, voici un exemple de rapport obtenu après une analyse de code source par SonarQube :

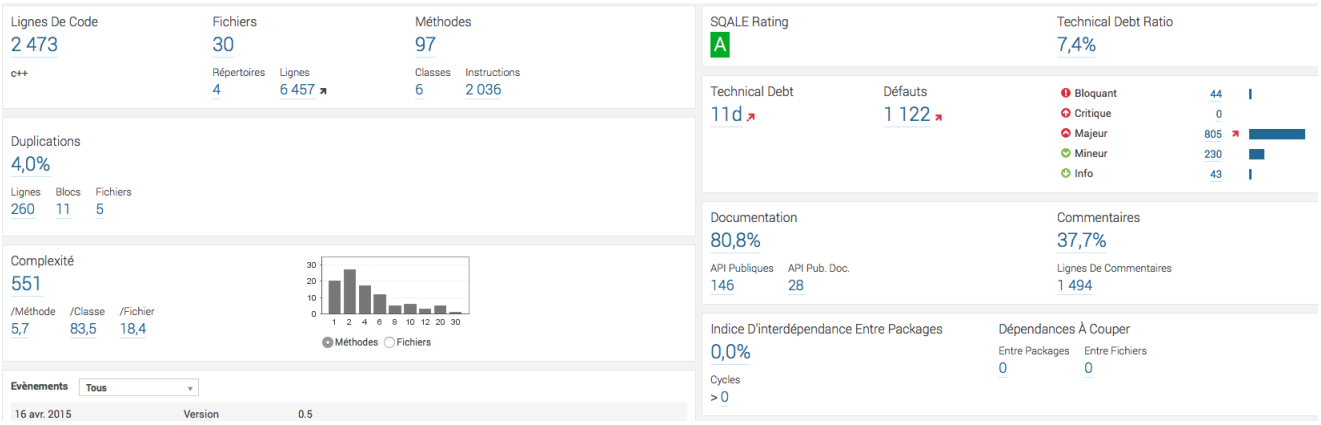

FIGURE 7.4 – Exemple de rapport SonarQube

.

Nous pouvons obtenir les mesures suivantes par exemple :

- Le nombre de lignes de code, de fichiers, de méthodes, de fonctions...
- La complexité de chacune des méthodes/fonctions.
- La présence ou non de cycles dans le code.
- Le respect ou non des règles mises en place.

Concernant le non-respect de règles Sonar, il est possible de voir exactement le problème avec les lignes correspondantes via l'interface. Il est souvent expliqué comment résoudre le problème rencontré ou comment l'éviter :

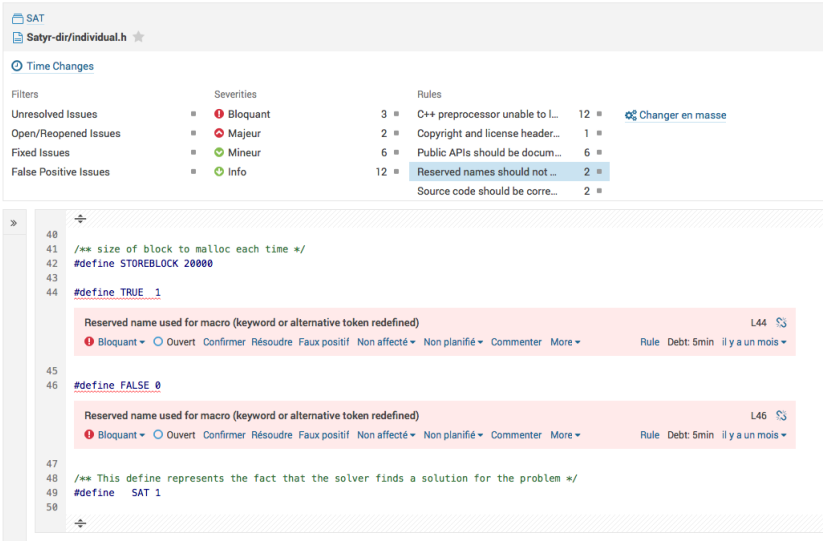

FIGURE 7.5 – Non-respect d'une règle de SonarQube

.

SonarQube offre la possibilité d'ajouter des règles personnalisées. Ainsi, nous pouvons définir pour chaque projet certaines règles qui n'existent pas de base dans SonarQube afin d'enrichir l'analyse de code et de s'assurer du respect de ces règles.

Il est aussi possible d'avoir des données plus précises sur chacun des fichiers comme nous pouvons le voir dans la figure précédente. Enfin, je souhaite préciser que la majorité des figures présentées dans cette partie sont issues du projet de fin d'études de Valentin Montmirail.

Bibliographie

[1] *Cahier de spécifications 2013/2014 - Anne Castier.*

[2] *Manuel d'installation de la machine serveur 2013/2014 - Anne Castier.*

[3] *Manuel d'installation de la machine cliente 2013/2014 - Anne Castier.*

[4] *Guide utilisateur 2013/2014 - Anne Castier.*

[5] *Rapport final de projet 2013/2014 - Anne Castier.*

[6] *Cahier de spécifications - Florent Clarret.*

[7] *Manuel d'installation de la machine serveur et des esclaves - Florent Clarret.*

[8] *Manuel d'administration du serveur - Florent Clarret.*

[9] *Guide de démarrage rapide - Florent Clarret.*

[10] *Guide utilisateur complet - Florent Clarret.*

Mise en place d'un serveur d'intégration continue et de qualité de code

Département Informatique
5e année
2014-2015

Rapport final de projet de fin d'études

Résumé : Ce document constitue le rapport final du projet de fin d'études visant à mettre en place un serveur d'intégration continue et de qualité de code au sein de Polytech Tours

Mots clefs : Rapport final, Intégration continue, Qualité de code, Serveur, Jenkins, Sonar

Abstract: This document is the final report of Florent Clarret's project, supervised by Mr Nicolas Ragot, Mr Pascal Meichel and Mr Vincent T'Kindt, during the year 2014-2015 in Polytech Tours.

Keywords: Final report, Continuous integration, Quality of code, Server, Jenkins, Sonar

---

Encadrants
Nicolas Ragot
nicolas.ragot@univ-tours.fr
Pascal Meichel
pascal.meichel@univ-tours.fr
Vincent T'Kindt
tkindt@univ-tours.fr

Université François-Rabelais, Tours

Étudiant
Florent Clarret
florent.clarret@etu.univ-tours.fr

DI5, 2014-2015