

# Serveur d'intégration continue Jenkins et d'analyse de code Sonar couplés à la forge logiciel SourceSup

**Sébastien MEDARD**

GIP RENATER

263 avenue du Général Leclerc

CS 74205 – 35042 Rennes Cedex

## Résumé

*L'intégration continue est devenue indispensable au bon déroulement d'un projet de développement. Elle permet d'automatiser les builds, le lancement des tests unitaires, les déploiements ainsi que l'analyse du code de manière périodique ou ponctuelle afin de repérer rapidement les problèmes. Un serveur d'analyse de code permet, grâce à un ensemble de règles, de scruter le code et de déterminer quelles parties ne respectent pas les bonnes pratiques établies.*

*La forge SourceSup (accessible via la fédération d'identité de l'enseignement supérieur et la recherche) offre maintenant l'accès à un serveur Jenkins (intégration continue) ainsi qu'à un serveur Sonar (analyse de code). Ces serveurs, grâce à des plugins, vont pouvoir récupérer les informations d'authentification que reçoit SourceSup et ainsi identifier à leur tour l'utilisateur connecté. L'utilisateur n'aura donc pas à recréer de compte sur ces serveurs pour y accéder.*

*Sur le serveur Jenkins un utilisateur pourra créer ses jobs, les lier aux dépôts de ses projets SourceSup et aussi lancer des analyses Sonar. Le créateur d'un job en devient l'administrateur et seul propriétaire, libre à lui ensuite d'en modifier la configuration pour accepter les autres utilisateurs de son projet. Il est possible de paramétrer un job Jenkins pour qu'il lance une analyse de code Sonar sur le code du projet. Par cette action, le serveur d'intégration continue va créer au niveau du serveur Sonar une nouvelle analyse. L'exécuteur du job va alors être créé automatiquement au niveau de Sonar et il se retrouvera propriétaire de l'analyse du code.*

Mots-clefs

*Intégration continue, Jenkins, analyse de code, Sonar, SourceSup, fédération d'identité, plugin*

## 1 Principe de l'intégration continue

L'intégration continue est une pratique logicielle rattachée à l'« Extreme Programming » dont le but est d'accélérer le développement des logiciels en automatisant l'intégration du code. Auparavant les projets rencontraient certains problèmes lorsque différentes parties du code, développées indépendamment les unes des autres, étaient intégrées en même temps. Des soucis apparaissaient fréquemment, notamment lors de commits partiels de code (un développeur voulant transmettre au gestionnaire de sources une évolution, oublie de commiter un fichier) ou lors d'un commit de fichiers de configuration liés au poste du développeur. L'objectif de l'intégration continue est de vérifier qu'à chaque modification de code, le build de l'application reste possible et qu'il n'y a pas de régressions provoquées. Pour y parvenir, plusieurs règles sont définies dans les méthodes agiles, comme l'hébergement des sources sur un dépôt unique, l'obligation de commiter régulièrement les modifications, l'automatisation des builds du projet grâce à des outils comme MAVEN ou ANT, ou l'exécution de tests unitaires lancés de façon automatique.

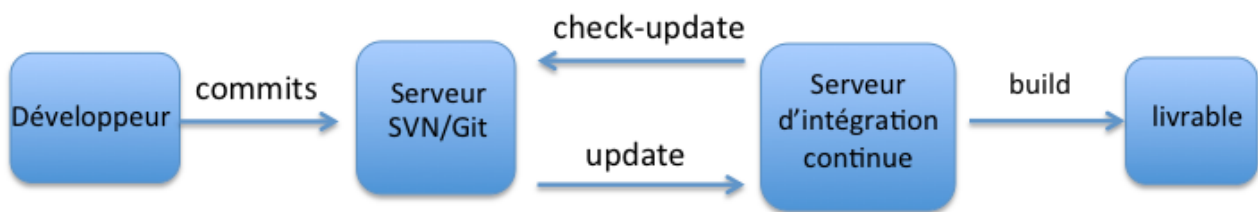


Figure 1 - Workflow d'intégration continue

## 2 La qualité de code

Dans le développement d'un logiciel, il est devenu important d'écrire du code source de qualité, respectant certaines règles de codage. En effet avoir du code lisible, bien documenté et conforme aux normes du langage utilisé est important pour réduire les possibilités de bogues ainsi que faciliter le transfert du code à d'autres développeurs. Il est donc important d'avoir une idée de la qualité de son code. La qualité d'un code source peut dépendre de plusieurs métriques, comme le nombre de lignes de code dans un fichier/classe, le nombre de lignes de code par méthode, le nombre d'instanciations d'objets, la présence de commentaires dans le code, la présence de code dupliqué dans plusieurs fichiers au lieu d'être factorisé, etc. Le serveur Sonar permet d'afficher des résultats de l'analyse du code d'un projet par différents outils comme checkstyle, PMD, findbugs et d'autres, et de montrer dans quels fichiers les erreurs sont situées ainsi que faire un bilan global sur la qualité du projet.

## 3 Intérêt pour SourceSup et ses utilisateurs

SourceSup est une plateforme de développement informatique. Le but des projets qu'elle héberge est de fournir un livrable pouvant être installé sur une machine ou déployé sur un serveur. Toutefois pour pouvoir tester régulièrement les modifications du code, il est important d'être capable de produire un livrable fonctionnel rapidement et de manière automatique. C'est justement l'un des objectifs d'une plateforme d'intégration continue comme Jenkins : lancer de manière régulière et automatique les opérations de création de build d'un projet. Cette action s'inscrit dans le workflow d'un projet et il est donc important que SourceSup puisse proposer ce service à ses utilisateurs.

De même, l'analyse du code par des outils est aussi très important, cela permet de voir si l'ensemble du code du projet respecte les mêmes règles, ce qui n'est pas toujours le cas avec un groupe de développeurs n'ayant pas forcément les mêmes manières d'écrire le code. Sonar s'inscrit dans les outils utiles pour le bon déroulement des projets, et a donc sa place sur SourceSup.

Mais ces serveurs peuvent être déployés sans trop de difficulté au sein d'un établissement, donc quel est intérêt de les installer sur SourceSup ? Tout d'abord les utilisateurs d'un établissement n'auront pas besoin d'installer ces serveurs en interne. Ensuite, il n'y a pas de maintenance à faire, notamment si un projet produit des mises à jours très régulières. Certaines mises à jours incluent des modifications de bases de données pour Sonar, ou des changements de format de données pour Jenkins, pouvant parfois entraîner des complications. Autre avantage non négligeable : la plateforme SourceSup est accessible depuis tout l'Internet, donc les utilisateurs peuvent s'y connecter de partout, les serveurs ne sont pas cloisonnés sur un réseau interne. Il est aussi possible de gérer les droits sur les jobs Jenkins et les analyses Sonar, il est donc aisé de montrer ses résultats ou ses configurations à des personnes d'un autre projet en leur offrant l'accès.

## 4 Couplage

### 4.1 Plugin FusionForge

SourceSup est un service basé sur la forge logicielle FusionForge. Cette application fonctionne avec un système de plugins intéressant. Chaque plugin peut être activé ou non dans un projet, selon les besoins des utilisateurs. Cela permet de ne pas perdre les développeurs car ceux-ci ne voient à l'écran que ce dont ils ont besoin. L'accès aux serveurs Jenkins et Sonar a été réalisé via un plugin FusionForge. Il est donc possible d'activer ou non cet accès au niveau de l'administration du projet. Ce plugin est simple, il présente les interfaces des serveurs Jenkins et Sonar dans une iframe. L'utilisateur n'a donc pas besoin de retenir l'URL de ces serveurs, ils sont présents dans SourceSup.

### 4.2 Authentification

SourceSup est un service déployé par RENATER, à destination de la communauté « enseignement supérieur et recherche », qui utilise l'authentification mise en place par RENATER pour tous ses services et qui est accessible à tous les membres de cette communauté, à travers la fédération d'identité. Le principe consiste à permettre à un utilisateur de s'authentifier sur un service avec la plateforme d'authentification de son établissement. Celle-ci transmettra certaines informations de l'utilisateur à SourceSup afin de l'identifier. L'un des principaux intérêts de cette architecture est de permettre à l'utilisateur de n'avoir qu'un seul couple login/mot de passe à gérer pour l'ensemble des services fédérés. Il est donc intéressant de transposer ce système déjà en place aux serveurs d'intégration continue et d'analyse de qualité de code, afin que l'utilisateur n'ait pas besoin de se créer un compte sur ces serveurs et qu'il soit automatiquement authentifié lors de l'accès à ces plateformes.

Pour le serveur Jenkins, différents modes d'authentification sont possibles, comme la connexion à un LDAP, une base de données, etc. Un plugin permet toutefois de déléguer l'authentification sur le serveur à un proxy. Ce proxy, un serveur Apache, reçoit déjà les informations d'authentification pour SourceSup, et les garde en session. Il est alors aisé, par une configuration Apache, de demander au proxy de transmettre ces informations à Jenkins pour qu'il puisse à son tour les exploiter.

Le serveur Sonar propose lui aussi différents modes d'authentification, mais ne possède pas l'équivalent du plugin « reverse proxy » de Jenkins. Nous avons comblé ce manque en développant un plugin Sonar permettant de déléguer l'authentification au serveur à un proxy. Toutefois Sonar nécessite une base de données pour fonctionner, et les utilisateurs doivent y être stockés. La table des utilisateurs en base n'est pas importante pour l'authentification, car celle-ci est récupérée via le proxy, mais elle est utile pour la gestion des droits. Elle doit donc être renseignée avec les utilisateurs possibles de SourceSup, sinon une personne connectée à Sonar n'aurait aucun droit sur rien.

### 4.3 Gestion des droits

SourceSup héberge des projets qui peuvent être privés ou publics. Les projets privés ne sont accessibles qu'à certains utilisateurs et ne doit pas être visibles des autres. Il faut pouvoir retrouver ces aspects sur Jenkins afin de ne pas contourner les droits des utilisateurs dans SourceSup. Le serveur d'intégration continue Jenkins est basé sur un système de jobs. Il existe plusieurs modes de gestion de droits sur les jobs dans Jenkins, comme offrir tous les droits sur tous les jobs aux utilisateurs connectés, gérer les droits grâce à une matrice globale à Jenkins, ou bien ne pas avoir de droits du tout. Ces solutions ne répondant pas au besoin de SourceSup, il fallait passer par un plugin permettant de gérer les droits au niveau des jobs. Ce plugin existe et permet à la création du job, de lui allouer une matrice de droits qui ne contient que le créateur du job. Cela signifie que seul le créateur du job a les droits sur son job, celui-ci n'étant pas visible des autres utilisateurs. Cette configuration permet de créer des jobs privés, mais aussi de les rendre publics si besoin, en autorisant dans la matrice l'utilisateur global « authenticate ». L'ajout des autres utilisateurs du projet SourceSup est laissé à la charge du propriétaire du job, qui peut facilement remplir la matrice des droits.

La même problématique d'aspect public/privé existe sur le serveur d'analyse de qualité de code Sonar. En effet une analyse d'un projet privé ne doit pas être publique, car elle permet de naviguer dans les différents fichiers, de visualiser les problèmes trouvés par Sonar, ainsi que d'afficher le bilan qualité du projet et l'historique des analyses. Sonar fonctionne avec une base de données, dans laquelle sont enregistrés tous les utilisateurs de SourceSup. Lors de la

création d'un utilisateur sur SourceSup, une entrée est automatiquement créée dans la base de données Sonar. Si celui-ci désire utiliser Sonar pour son projet, la base sera déjà configurée. Cela est obligatoire car du côté de Sonar, les droits sont gérés dans la base de données. Sur la plateforme SourceSup, il est prévu que la création d'une analyse Sonar ne soit possible que via Jenkins. Les deux serveurs étant fortement liés grâce au plugin « Sonar » de Jenkins. Celui-ci offre la possibilité de lancer l'analyse du code concerné par le job. Le job va donc demander la création d'une analyse de code sur Sonar, mais ce plugin ne permet pas de lier directement cette nouvelle analyse à un utilisateur. Si le code en question est sensé être privé, cela devient problématique. Pour combler ce défaut, le plugin « Sonar » de Jenkins a du être modifié afin de lui permettre d'indiquer directement dans la base Sonar à qui appartient l'analyse créée. L'utilisateur lançant l'exécution du job Jenkins va alors se retrouver propriétaire et administrateur de l'analyse sur Sonar. Il sera seul à pouvoir y avoir accès mais libre à lui de modifier la configuration et d'ajouter les autres personnes de son projet.

#### 4.4 Architecture

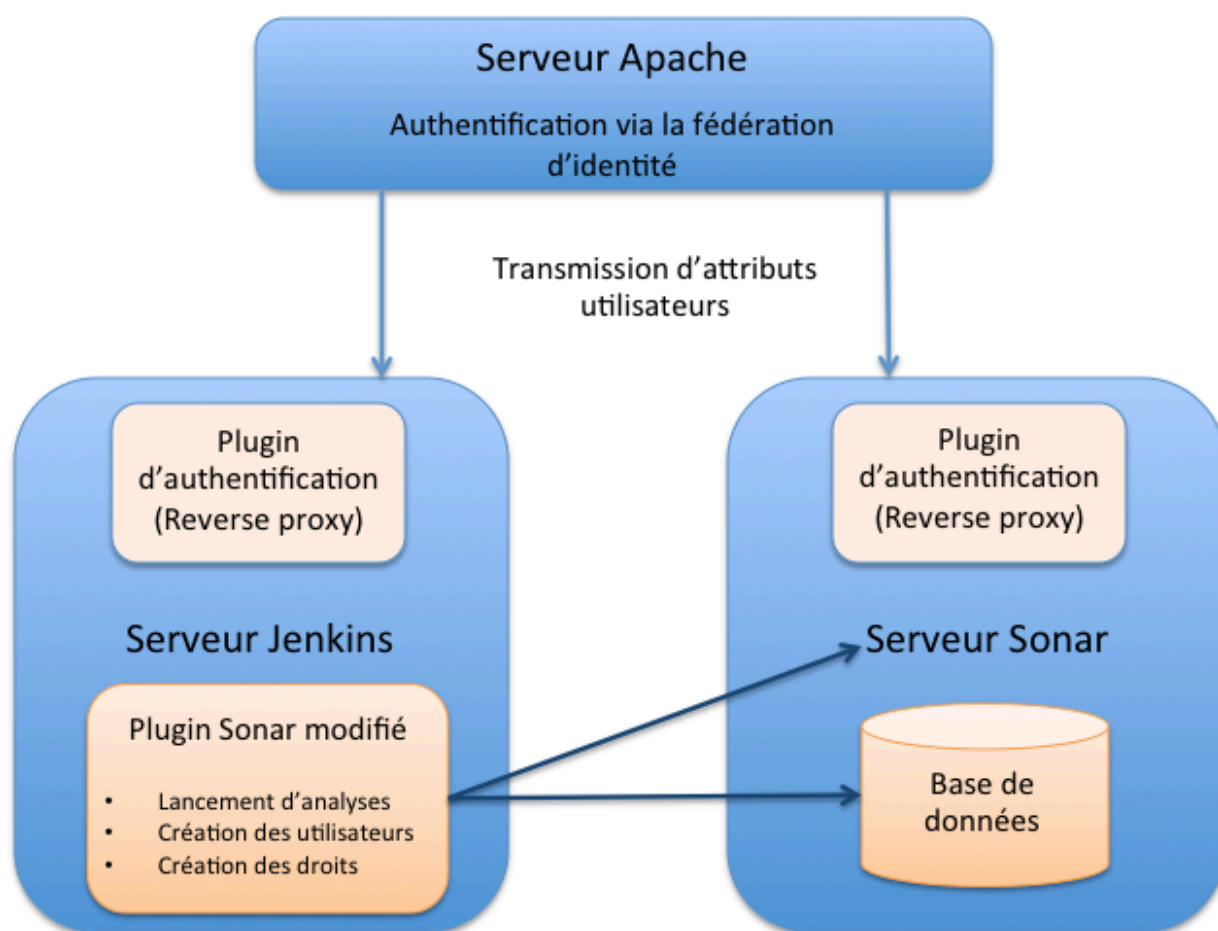


Figure 2 - Architecture du service SourceSup couplé aux serveurs Jenkins et Sonar

## 5 Conclusion

Le couplage entre SourceSup et les serveurs Jenkins et Sonar s'opère bien. Les fonctionnalités qu'apportent ces serveurs sont utiles aux utilisateurs de la forge pour mener à bien leurs développements. Plusieurs projets ont déjà pu améliorer la qualité et la robustesse de leur code.

Toutefois des améliorations restent possibles, comme instaurer un parallélisme entre plusieurs serveurs Jenkins, ou si on veut voir plus loin, avec le développement du cloud, déployer automatiquement des machines virtuelles avec des serveurs Jenkins et Sonar et permettre l'accès aux utilisateurs d'un projet.

Le prochain ajout prévu sera la mise en place d'un serveur Nexus au sein de cette architecture. Ce serveur permet de mieux gérer les snapshots générés, de récupérer sur le net qu'une seule fois les différentes bibliothèques, de déposer les snapshots générés par les builds, etc.

## 6 Bibliographie

<http://hudson-ci.org/>

<http://linsolas.developpez.com/articles/hudson/>

<http://www-igm.univ-mlv.fr/~dr/XPOSE2011/IntegrationContinue/index.html>

**Fabian PIAU** : L'INTÉGRATION CONTINUE : Améliorer la qualité des logiciels et réduire les risques :  
[http://fabianpiou.com/wp-content/uploads/post/00003/Dossier\\_CI.pdf](http://fabianpiou.com/wp-content/uploads/post/00003/Dossier_CI.pdf)

<http://www.sonarqube.org/>

<http://akrambenaissi.wordpress.com/2011/01/10/qualite-de-code-avec-sonar/>

[http://linsolas.developpez.com/articles/java/qualite/sonar/?page=page\\_2#LII](http://linsolas.developpez.com/articles/java/qualite/sonar/?page=page_2#LII)

<http://docs.codehaus.org/display/SONAR/Running+SonarQube+behind+a+Proxy>