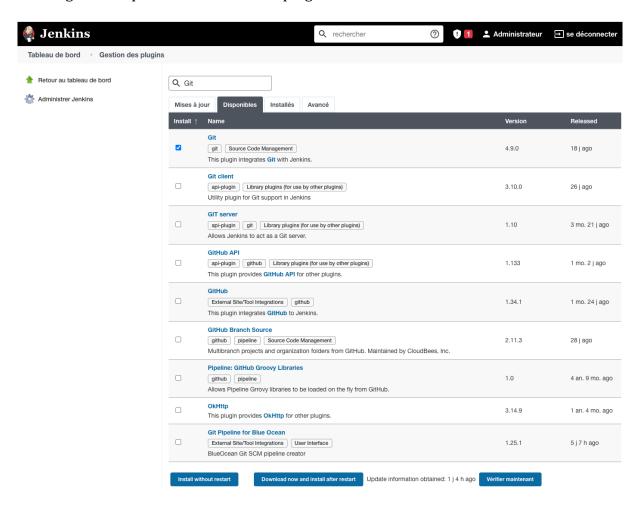
Jenkins et Git

Vous avez vu précédemment comment définir des jobs de build en utilisant les codes sources localement. Comme vous le savez, dans un projet informatique, il est indispensable d'utiliser un outil de versionning, comme Git. Il est donc essentiel d'associer l'outil d'intégration utilisé avec le dépôt correspondant. Le but de cette partie est donc de vous montrer comment associer Git à Jenkins.

1. Git et Jenkins

Vous allez ajouter d'abord le plugin Git à Jenkins. Allez dans Administrer Jenkins > Gestion des Plugins > Disponibles. Cherchez le plugin Git :

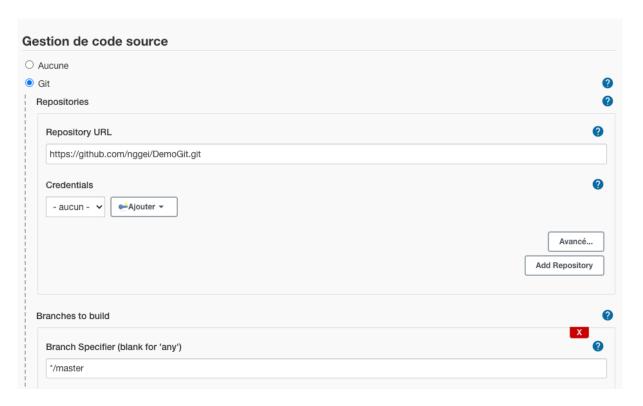


Cliquez sur « Installer without restart ». L'installation s'effectue. Cochez « Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours ».

Une fois l'ajout du plugin Git à Jenkins est effectué, créez un Job (projet Maven) comme fait précédemment sur Jenkins. Ici on l'appelle SimpleMavenGitWebProject.

Vous allez configurer Jenkins avec Git. Autrement dit, jenkins va considérer les codes sources disponibles sur votre repository et non pas sur votre machine. Nous allons utiliser ici les projet Web précédemment développé et mis sur le repository GitHub.

Dans Gestion de code source, choisir Git, puis renseignez l'URL de votre repository github :



Puis plus bas dans « *Ce qui déclence le build* », cochez « *Scrutation de l'outil de gestion de version* ». Dans le champ « Planning », il faut écrire l'expression permettant à Jenkins de savoir comment il va vérifier les changements dans Git. Vous pouvez cliquer sur « ? » de planning pour voir comment écrire une expression. Ici par exemple, on veut que ça soit chaque minute (*****):



TD PDLA 4IR INSA de Toulouse guermouc@insa-toulouse.fr

Plus bas dans « Build », on veut à chaque fois qu'il y a un changement dans Git, compiler le projet que Jenkins aurait cherché de Git.

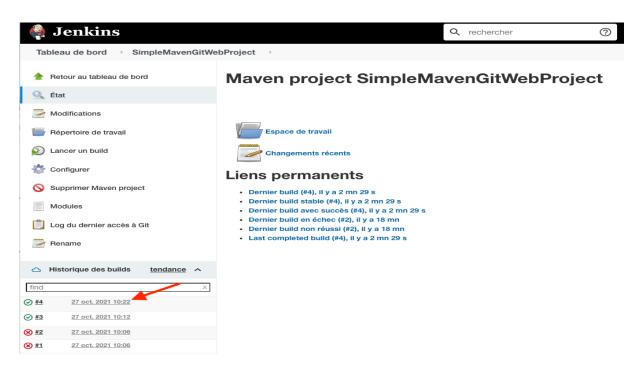
Remarque:

Ici, on ne met plus le chemin absolu du fichier pom.xml de votre projet en local comme vous l'aviez fait précédemment. En effet, Jenkins va récupérer le projet disponible sur GitHub et donc le fichier pom.xml va être présent dans le répertoire demoWebApp (nom du projet présent dans le repository GitHub) dans le workspace de Jenkins. Sauvegardez.



Jenkins va scruter votre dépôt toute les minutes. Ne lancez pas manuellement le projet Jenkins.

Faites des modifications à votre projet (par exemple dans votre servlet ajoutez un message) et poussez-les sur votre dépôt. Retournez à Jenkins et cliquez sur votre job « SimpleMavenGitWebProject » et ne faites rien. Juste observez le lancement d'un build automatiquement, suite à un changement dans votre repository Git. Vérifier l'historique des builds:



TD PDLA 4IR INSA de Toulouse guermouc@insa-toulouse.fr

Regardez le log. Vous pouvez voir les changements qui ont été poussés (le fichier concerné et le commit réalisé avec son auteur) en cliquant sur *details*.



Ajouter une description



Changes

1. modification de la servlet (details / githubweb)



Lancé par un changement dans la base de code



Revision: 320d20fcc8cc3737acdadba88a8d4f7367db1ee5 Repository: https://github.com/nggei/DemoGit.git

· refs/remotes/origin/master

Construction du module

demoWebApp Maven Webapp 1,6 s