

Introduction à Jenkins

Pour aider les développeurs dans leur travail

Par **Alexandre Laurent** 

Date de publication : 28 mai 2019

Dernière mise à jour : 9 septembre 2019

Cet article permet au lecteur de découvrir le logiciel Jenkins et à mieux comprendre son intérêt tout en lui offrant les bases pour bien démarrer avec cet outil.

Commentez

I - Introduction.....	3
II - Fonctionnement.....	3
II-A - Modules.....	3
III - Installation.....	4
IV - Présentation de l'interface.....	4
V - Configuration d'un projet.....	9
V-A - Processus de compilation d'un projet.....	10
V-A-1 - Choisir entre pipeline et freestyle.....	10
V-B - Job freestyle.....	11
V-B-1 - Création d'un job.....	11
V-B-2 - Configuration.....	11
V-C - FreeStyle multiconfiguration.....	11
V-C-1 - Chemin des outils.....	12
V-C-2 - Exécution de commande.....	12
V-C-2-a - Exécution de commande spécifique à la machine.....	12
V-C-3 - Généralisation.....	12
V-C-4 - Remarques.....	12
V-D - Job Pipeline.....	13
V-D-1 - Création d'un job.....	13
V-D-2 - Configuration.....	13
V-D-2-a - Script Pipeline.....	13
V-D-2-b - Restreindre l'exécution d'une tâche.....	14
V-D-2-c - Paralléliser des tâches.....	14
V-D-3 - Écrire son script.....	15
V-D-3-a - Module Blue Ocean.....	15
V-D-3-b - Générateur de code.....	15
V-D-3-c - Générateur déclaratif.....	15
V-D-3-d - Documentation.....	15
V-D-4 - Script ou déclaratif.....	15
V-D-5 - Configuration du projet BouskNet.....	16
V-E - Paramètres.....	16
V-F - Lancer la compilation par requête Web.....	17
V-F-1 - Lancement avec authentification.....	17
VI - Gestion des accès.....	18
VII - Trucs et astuces.....	18
VII-A - Configuration des esclaves.....	18
VII-A-1 - Nombre de fils d'exécution.....	19
VII-A-2 - Étiquettes.....	19
VII-A-3 - Variables d'environnement.....	19
VII-B - Configuration des tâches.....	20
VII-B-1 - Ignorer des commits.....	20
VII-B-2 - Artefacts.....	20
VII-B-3 - Empêcher l'exécution de deux tâches en parallèle.....	20
VIII - Conclusion.....	20
IX - Ressources supplémentaires.....	20
X - Remerciements.....	20

I - Introduction

Le travail nécessaire pour développer un projet informatique est considérable et loin d'être facile. Pour assurer une meilleure réussite du projet, il est nécessaire de mettre en place des outils permettant d'aider l'équipe travaillant sur le projet. Notamment, en simplifiant son travail, l'équipe pourra gagner en sérénité et se concentrer sur les tâches pour lesquelles leur expertise est requise.

Cette sérénité peut être acquise en s'assurant que chaque modification du projet ne casse pas le produit. Ainsi, il est imaginable de mettre en place :

- une vérification de la compilation (est-ce que le projet compile sur toutes les plateformes visées ?) ;
- des tests automatisés du projet (est-ce que la modification provoque des régressions ?) ;
- des tests de déploiement (est-ce que le projet peut toujours être livré ?) ;
- la publication automatisée des nouvelles versions ;
- la vérification d'éléments annexes sur lesquels le projet repose.

Ces tâches peuvent rapidement devenir rébarbatives, prendre beaucoup de temps ou encore, être source d'erreurs. Toutefois, ces tâches ont aussi la particularité de se répéter plus ou moins régulièrement. De par ce fait, et comme la machine est la meilleure pour effectuer des tâches répétitives, nous allons vouloir les automatiser.

Dans les grandes lignes, nous avons besoin d'un outil qui effectuera des actions déclenchées soit par un utilisateur, soit périodiquement, soit après une modification du code. **Jenkins**, un logiciel open source et multiplateforme permet de mettre en place cette infrastructure.

II - Fonctionnement

Jenkins est un serveur Web, multiplateforme, développé en JAVA. Jenkins est construit en suivant un modèle maître/esclave. Ainsi, vous configurez vos projets à travers le serveur Web sur la machine maître et les esclaves (qui sont des instances légères de Jenkins) vont effectuer les actions configurées.



La machine maître peut aussi exécuter des tâches. Plus précisément, chaque machine (maître ou esclave) possède un ou plusieurs exécuteurs et chaque exécuteur peut faire une tâche à la fois.



il est conseillé de ne pas exécuter de tâches sur l'instance maître. En effet :

- *une tâche sur l'instance maître peut accéder aux fichiers de configuration de Jenkins (notamment les modifier, quant bien même le créateur de la tâche n'a habituellement pas de tels droits) ;*
- *les tâches exécutées sur l'instance maître ralentiront l'exécution de Jenkins ;*
- *les outils installés sur la machine maître pour exécuter les tâches augmentent inévitablement la surface d'attaque.*

II-A - Modules

Par défaut, Jenkins est similaire à une coquille vide. Il y a juste le nécessaire pour administrer le serveur, mettre en place les machines esclaves et exécuter des commandes. Tout le reste des fonctionnalités (récupération des sources à partir de Git, envoi d'email lors d'un échec...) est implémenté sous la forme de modules (*plugins*).

Ainsi, vous installez uniquement ce dont vous avez besoin. Les modules ajoutent des fonctionnalités à Jenkins et vont compléter l'interface par de nouvelles options.

Dans ce tutoriel, nous utilisons les modules suivants :

- Pipeline ;
- Mercurial ;
- Blue Ocean.

Jenkins propose un gestionnaire de modules qui permet de les installer, les supprimer ou les mettre à jour à travers une interface simple. Celle-ci se trouve dans le menu Administrer Jenkins → Gestion des plugins.

i *Par défaut, Jenkins installe de nombreux modules, notamment liés à JAVA. Si vous ne les utilisez pas, vous pouvez les désinstaller.*

III - Installation

L'installation de l'instance maître est plutôt simple. L'installation se fait de manière intuitive : sous Linux, vous pouvez utiliser votre gestionnaire de paquets. Pour Windows, un installateur au format .MSI est proposé. Pour OS X, c'est un fichier pkg. Finalement, si aucune des solutions proposées sur la page de [téléchargement](#) ne vous convient, vous pouvez toujours utiliser le fichier .war.

i *Le serveur est accessible à travers le port 8080. Sur la machine hébergeant Jenkins, vous pourrez accéder au serveur Web à travers un navigateur Web en utilisant l'adresse localhost:8080.*

i *Deux versions sont proposées, une version stable, renouvelée toutes les douze semaines et ayant un support étendu et une version de développement mise à jour toutes les semaines. Toutefois, Jenkins vous proposera les mises à jour directement à travers l'interface Web.*

Une fois l'installation de l'instance maître finalisée, vous pourrez configurer autant d'instances esclaves que vous le souhaitez. Pour cela, il suffit de créer de nouveaux nœuds. Afin que Jenkins puisse utiliser le nœud, il est nécessaire qu'une connexion s'établisse entre eux. Deux approches sont possibles :

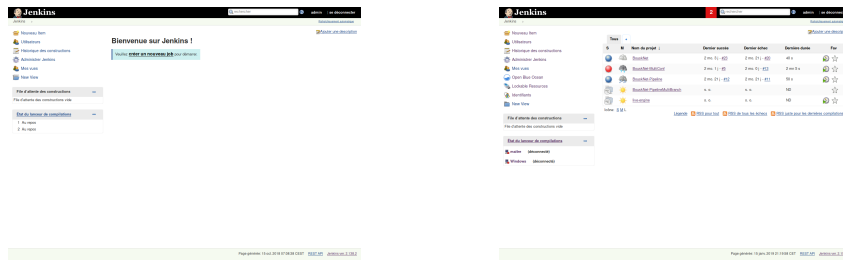
- la machine esclave lance d'elle-même l'agent JAVA pour se connecter à Jenkins ;
- Jenkins exécute une commande pour lancer l'agent JAVA sur la machine esclave. Par exemple, cela peut s'effectuer grâce à une connexion SSH.

Le plus simple est la première solution. En effet, sur la page de configuration du nœud, vous trouverez un bouton pour lancer l'agent JAVA à travers le navigateur (fichier JNLP). Ensuite, il sera possible d'installer ce dernier comme un service pour une exécution automatique.

i *Si vous ne voyez pas la méthode de lancement « Launch agent via Java Web Start », c'est que vous devez donner un port pour JNLP dans la configuration de sécurité globale.*

IV - Présentation de l'interface

L'accueil est un écran récapitulatif permettant de voir l'état des tâches configurées sur Jenkins.

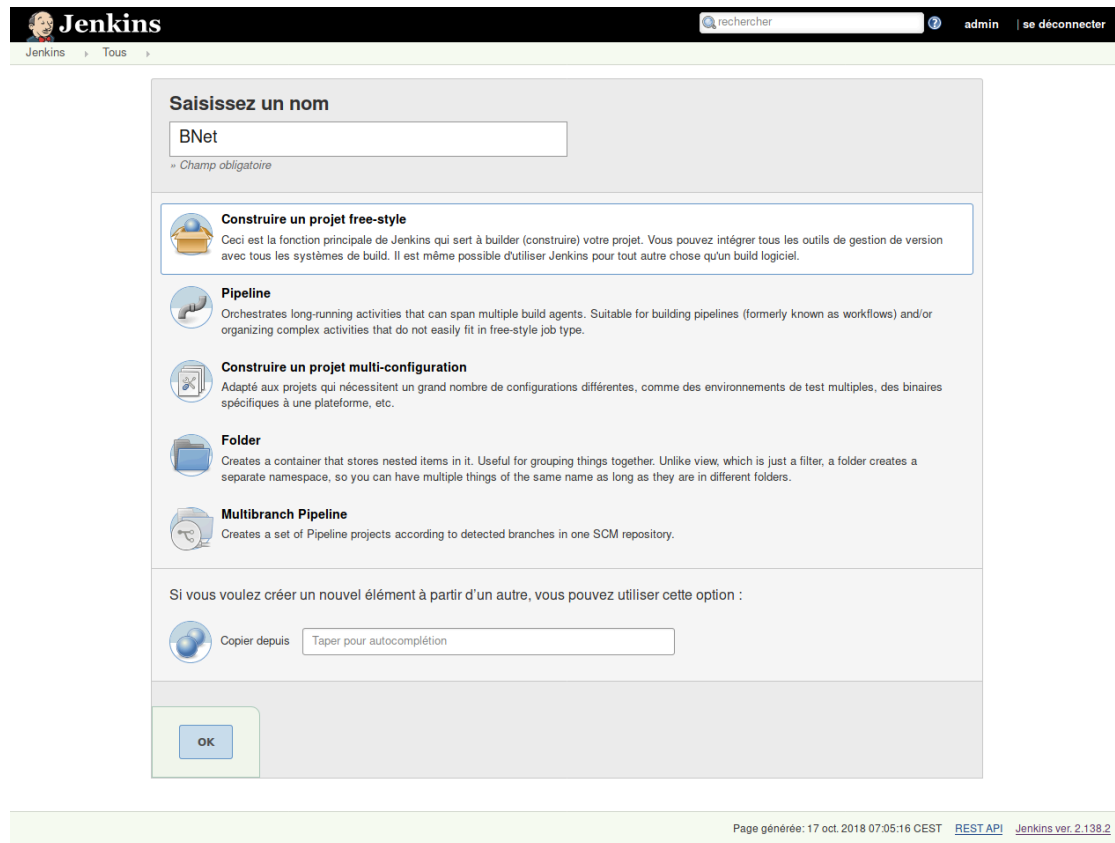


Le tableau de bord juste après l'installation. Le tableau de bord avec quelques projets.


L'interface de Jenkins se décompose en trois parties : un en-tête, une colonne à gauche contenant un menu suivi d'un récapitulatif des tâches en cours et en attente et la partie centrale.

Le tableau de bord permet de facilement voir les tâches réussies et en échec, d'accéder rapidement aux informations sur celles-ci et de lancer manuellement une tâche.

Dans le menu à gauche, la première entrée « Nouveau Item » permet de créer une nouvelle tâche :



Une fois la création validée, vous êtes redirigé sur la page de configuration du projet :

 Jenkins

rechercher

admin | se déconnecter

Jenkins

BouskNet

General

Gestion de code source

Ce qui déclenche le build

Environnements de Build

Build

Actions à la suite du build

Description

[Plain text] [Prévisualisation]

☐ Ce build a des paramètres

☐ This build requires lockable resources

☐ Supprimer les anciens builds

☐ Throttle builds

☐ Désactiver le projet

☐ Exécuter des builds simultanément si nécessaire

Avancé...

Gestion de code source

☒ Aucune

☐ Git

Ce qui déclenche le build

☐ Déclencher les builds à distance (Par exemple, à partir de scripts)

☐ Construire après le build sur d'autres projets

☐ Construire périodiquement

Sauver

Apply

Version de version

Environnements de Build

☐ Use secret text(s) or file(s)

Build

Ajouter une étape au build

Actions à la suite du build


Ajouter une action après le build

Page générée: 17 oct. 2018 07:06:09 CEST

[REST API](#)

[Jenkins ver. 2.138.2](#)

En validant cette page, vous arrivez sur la page du projet. Elle permet de lancer une compilation, d'avoir l'historique des builds ou encore d'accéder à l'espace de travail permettant de visualiser les fichiers du projet :


Jenkins

2

[admin](#) | [se déconnecter](#)

[Jenkins](#) > [BouskNet](#) >
 [Rafraîchissement automatique](#)

- [Retour au tableau de bord](#)
- [État](#)
- [Modifications](#)
- [Répertoire de travail](#)
- [Lancer un build](#)
- [Supprimer Projet](#)
- [Configurer](#)
- [Favoris](#)
- [Open Blue Ocean](#)
- [Rename](#)

Projet BouskNet

Voici une petite description de test

[modifier la description](#)

[Désactiver le projet](#)

[Espace de travail](#)
























[Changements récents](#)

Liens permanents

- [Dernier build \(#23\), il y a 2 mo. 5 j](#)
- [Dernier build stable \(#23\), il y a 2 mo. 5 j](#)
- [Dernier build avec succès \(#23\), il y a 2 mo. 5 j](#)
- [Dernier build en échec \(#20\), il y a 2 mo. 23 j](#)
- [Dernier build non réussi \(#20\), il y a 2 mo. 23 j](#)
- [Last completed build \(#23\), il y a 2 mo. 5 j](#)

Historique des builds

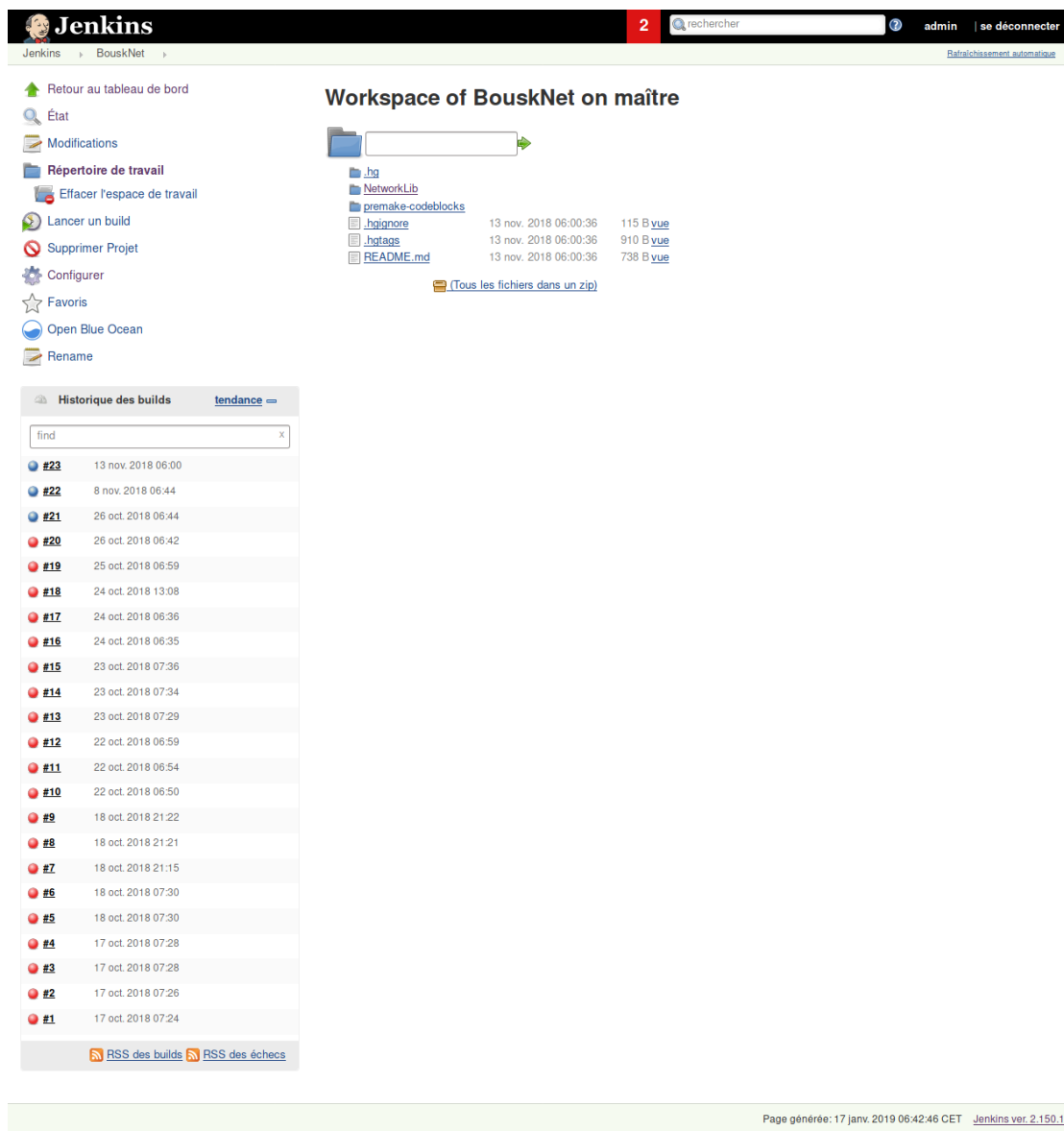
tendance ▾

	#23	13 nov. 2018 06:00
	#22	8 nov. 2018 06:44
	#21	26 oct. 2018 06:44
	#20	26 oct. 2018 06:42
	#19	25 oct. 2018 06:59
	#18	24 oct. 2018 13:08
	#17	24 oct. 2018 06:36
	#16	24 oct. 2018 06:35
	#15	23 oct. 2018 07:36
	#14	23 oct. 2018 07:34
	#13	23 oct. 2018 07:29
	#12	22 oct. 2018 06:59
	#11	22 oct. 2018 06:54
	#10	22 oct. 2018 06:50
	#9	18 oct. 2018 21:22
	#8	18 oct. 2018 21:21
	#7	18 oct. 2018 21:15
	#6	18 oct. 2018 07:30
	#5	18 oct. 2018 07:30
	#4	17 oct. 2018 07:28
	#3	17 oct. 2018 07:28
	#2	17 oct. 2018 07:26
	#1	17 oct. 2018 07:24

[RSS des builds](#)
[RSS des échecs](#)

Page générée: 17 janv. 2019 06:44:51 CET
 [REST API](#)
[Jenkins ver. 2.150.1](#)

Page du projet



Jenkins 2 admin | se déconnecter

Jenkins > BouskNet > [Rafraîchissement automatique](#)

Retour au tableau de bord
État
Modifications
Répertoire de travail
Effacer l'espace de travail
Lancer un build
Supprimer Projet
Configurer
Favoris
Open Blue Ocean
Rename

Workspace of BouskNet on maître

Fichier	Date	Statut	Statut
.hg	13 nov. 2018 06:00:36	115 B	vue
NetworkLib	13 nov. 2018 06:00:36	910 B	vue
premake-codeblocks	13 nov. 2018 06:00:36	738 B	vue
.hgignore	13 nov. 2018 06:00:36		
.hgtags	13 nov. 2018 06:00:36		
README.md	13 nov. 2018 06:00:36		

(Tous les fichiers dans un zip)

Historique des builds [tendance](#)

find

#	Date	Statut
#23	13 nov. 2018 06:00	✓
#22	8 nov. 2018 06:44	✓
#21	26 oct. 2018 06:44	✓
#20	26 oct. 2018 06:42	✓
#19	25 oct. 2018 06:59	✓
#18	24 oct. 2018 13:08	✓
#17	24 oct. 2018 06:36	✓
#16	24 oct. 2018 06:35	✓
#15	23 oct. 2018 07:36	✓
#14	23 oct. 2018 07:34	✓
#13	23 oct. 2018 07:29	✓
#12	22 oct. 2018 06:59	✓
#11	22 oct. 2018 06:54	✓
#10	22 oct. 2018 06:50	✓
#9	18 oct. 2018 21:22	✓
#8	18 oct. 2018 21:21	✓
#7	18 oct. 2018 21:15	✓
#6	18 oct. 2018 07:30	✓
#5	18 oct. 2018 07:30	✓
#4	17 oct. 2018 07:28	✓
#3	17 oct. 2018 07:28	✓
#2	17 oct. 2018 07:26	✓
#1	17 oct. 2018 07:24	✓

[RSS des builds](#)
[RSS des échecs](#)

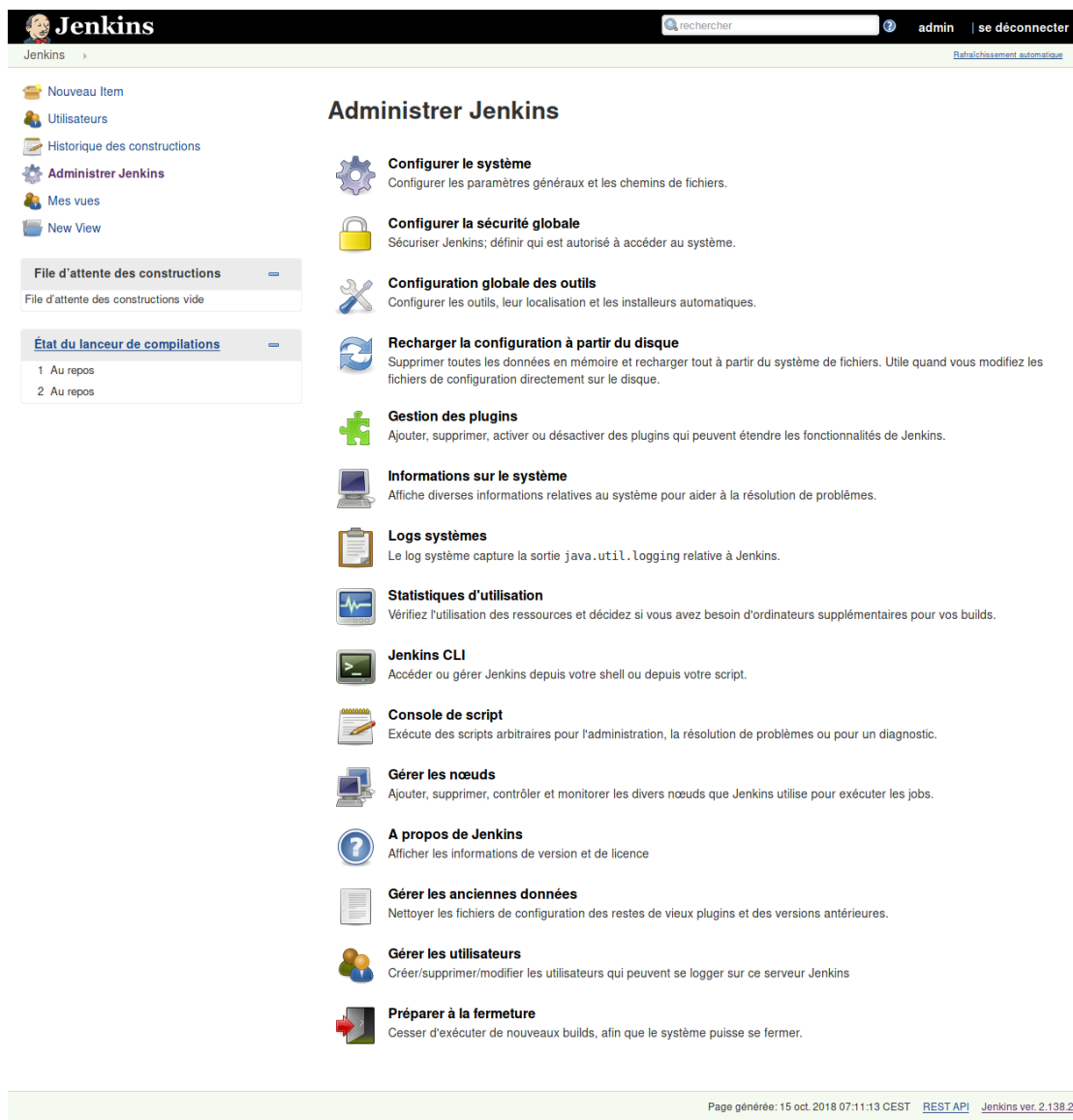
Page générée: 17 janv. 2019 06:42:46 CET [Jenkins ver. 2.150.1](#)

Explorateur de fichiers du projet



Le lien « tendance » dans l'encadré de l'historique des builds permet d'obtenir un historique des temps de construction du projet.

L'administration de Jenkins est séparée en plusieurs catégories :



Dans la configuration système, on pourra retrouver la configuration de la machine serveur en tant que machine pouvant exécuter des tâches ainsi que la configuration de paramètres exposés par les modules.

Dans la configuration de la sécurité globale, on pourra gérer comment les utilisateurs et les esclaves se connectent au serveur et comment le serveur doit se connecter à Internet.

La configuration globale des outils permet d'indiquer l'emplacement des outils et si besoin est d'activer des options spécifiques à ceux-ci. Par exemple, c'est ici que l'on peut configurer quel JDK utiliser et s'il n'est pas trouvé automatiquement par Jenkins, de spécifier son chemin.

La page « Gestion des plugins » amène à une interface permettant d'installer, mettre à jour et supprimer des modules. Finalement, vous pouvez accéder à la configuration des utilisateurs Jenkins et des nœuds au travers de leur page de gestion respective.



Cette présentation n'est pas exhaustive : le lecteur est amené à explorer les différents menus et à découvrir de lui-même ce qu'offre Jenkins.

V - Configuration d'un projet

Pour montrer le fonctionnement de Jenkins, nous allons utiliser le projet C++ de **Bousk** : **BouskNet**, une bibliothèque pour les applications réseau, développée pour les besoins du **cours C++ sur le réseau**.



Même si Jenkins est développé en JAVA et que les fonctionnalités pour les projets JAVA sont avancées, il est parfaitement possible de gérer des projets dans d'autres langages.

V-A - Processus de compilation d'un projet

Globalement, le processus permettant de compiler un projet est le suivant :

- récupération du projet à partir du serveur de contrôle de version ;
- création des fichiers de compilation ;
- compilation ;
- application des tests automatiques ;
- préparation de la version à distribuer ;
- mise en place de la version à distribuer.

Ce processus doit donc être configuré dans Jenkins. Deux méthodes de configuration s'offrent à vous :

- par le biais d'un pipeline, qui définit la configuration d'un projet grâce à du code ;
- la méthode historique (*freestyle*) où la configuration du projet se réalise à travers un formulaire à remplir. Dans ce cas, certaines étapes sont évidentes, d'autres se font grâce à l'exécution de scripts et ne sont donc pas guidées.

Pour cet article, nous nous limitons au plus simple : on ne teste que la compilation du projet. Cela veut dire qu'il suffit de configurer la récupération des sources, générer le Makefile avec `premake` et lancer la compilation avec `make`. Ainsi, nous vérifions si le projet compile, sur chaque plateforme, pour chaque changement de code. Ajouter l'exécution des tests ou des étapes supplémentaires est simple une fois le projet en place.

V-A-1 - Choisir entre pipeline et freestyle

Les tâches *freestyle* (méthode historique) permettent de configurer le projet en remplissant un formulaire détaillant les étapes de construction du projet. Toutefois, pour les étapes de compilation, vous serez rapidement amené à écrire des lignes de commande BATCH ou Bash pour lancer vos outils de compilation.

De plus, dès lors que votre projet est un peu plus complexe (gestion de plusieurs plateformes, de plusieurs variantes...), vous allez rapidement devoir ajouter de la logique à vos scripts afin de gérer tous les cas. On remarque alors que la configuration des tâches se rapproche de l'écriture d'un programme.

Pipeline donne une solution à cette problématique en offrant un nouveau langage, spécifique, pour écrire la configuration de vos tâches. Grâce à ce langage (basé sur Groovy), vous pourrez rendre la configuration plus simple et lisible, et ce, tout en gérant des cas de plus en plus complexes.

Aussi, le Pipeline peut être sauvegardé dans un fichier, celui-ci pouvant être versionné. Ainsi, vous pouvez lier une configuration spécifique à une version spécifique de votre projet. Plus précisément, la configuration de votre projet sera toujours en accord avec votre projet, et donc toujours compilable, et ce, même si vous devez compiler une ancienne version.

En bref, Pipeline est mieux et plus avancé. À l'heure actuelle, il n'y a aucune raison de commencer un projet « *freestyle* », et ce, même si les projets *freestyle* sont globalement (et à première vue) plus simples à mettre en place. Même si Pipeline nécessite d'apprendre un nouveau langage, ce dernier est simple et Jenkins offre de nombreux outils pour une mise en place rapide.



Pipeline offre aussi les avantages de permettre de paralléliser les étapes du projet et offre une meilleure interface pour lire les logs.

V-B - Job freestyle

L'ancien modèle de configuration des tâches se fait intégralement au travers de l'interface utilisateur de Jenkins.

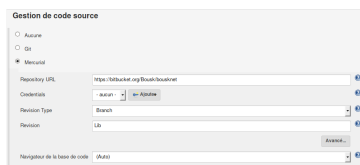
V-B-1 - Création d'un job

Il suffit de cliquer sur « Nouveau Item », donner un nom puis de sélectionner le type de job (ici, projet free-style). De là, vous serez redirigé vers la page de configuration de votre nouveau job.

V-B-2 - Configuration

La première partie de la page de configuration est négligeable. Elle permet d'indiquer une description du projet (affichée sur la page du projet), de choisir si la tâche a des **paramètres**, ou encore de gérer comment le projet est compilé ou est-ce que les anciennes compilations doivent être conservées.

Ensuite arrive la gestion du code source, ou comment nous allons récupérer le code source du projet. Les différents outils disponibles ici dépendent des modules installés. Ayant installé les modules Mercurial et Git, nous ne retrouvons donc que Mercurial et Git.



Il est possible de configurer plusieurs dépôts à récupérer avant de lancer la compilation, de spécifier une branche à l'aide d'un paramètre ou encore, de spécifier un commit spécifique avec la syntaxe @XXX (où XXX est le numéro du commit).

Après la configuration de la récupération du code, vous pouvez configurer la périodicité de cette récupération, ou autrement dit, qu'est-ce qui déclenche l'exécution de cette tâche. Cela peut être un déclenchement manuel, à la suite d'autres tâches, périodique ou simplement parce que le code a changé (vérification périodique du code sur le serveur de gestion des versions).

Enfin arrive l'étape de compilation. Vous pouvez la configurer en ajoutant des scripts shell ou batch afin de lancer les commandes de compilation du projet. Les commandes sont évidemment exécutées sur la machine esclave. Donc, si votre esclave est une machine Linux, utilisez les scripts shell, sinon, les scripts batch.

Finalement arrivent les étapes suivant la réussite de la tâche. C'est ici que l'on pourra, par exemple, notifier par courriel de l'échec de la tâche ou faire un rapport.

V-C - FreeStyle multiconfiguration

Le problème avec la configuration précédente est que si vous souhaitez compiler le projet sur plusieurs plateformes, vous serez bloqué, car les commandes sont spécifiques à une plateforme.

Pour pallier cela, Jenkins fournit une tâche de type « multiconfiguration ». Celle-ci permet de définir un ou plusieurs ensembles de paramètres (appelés axes) qui définiront des variables ayant le nom de l'axe dans les étapes du projet.

Pour une tâche dont le but est d'exécuter un ensemble de tests, ou de compiler un ensemble de produits sur un même type de machine, cela peut être simple à mettre en place pour peu que vous puissiez indiquer un paramètre déterminant ce qu'il faut faire.

Par contre, si vous souhaitez configurer des tâches qui sont sensiblement différentes suivant les machines sur lesquelles vous les exécutez (par exemple, la compilation sur Windows et une compilation sur Linux), vous devrez recourir à quelques astuces.

V-C-1 - Chemin des outils

La première difficulté sera de retrouver vos outils suivant les machines sur lesquelles vous exécutez votre tâche. Pour résoudre cela, vous pouvez configurer des variables d'environnement spécifiques à votre machine dans la page de configuration de la machine.

V-C-2 - Exécution de commande

Sous Linux, vous n'avez pas d'interpréteur Batch. Sous Windows, vous n'avez, par défaut, pas d'interpréteur Shell. Il faudra en installer un, tel que **Git for Windows**, et de rendre disponible l'exécutable sh.exe (en l'ajoutant dans la variable d'environnement PATH). Ainsi, même sous Windows, vous pouvez exécuter les scripts shell de Jenkins.

V-C-2-a - Exécution de commande spécifique à la machine

Le nom que vous donnez à vos axes de configuration de votre tâche multiconfiguration se transcrit en variables définies et accessibles dans vos scripts.

Par conséquent, vous pouvez très bien écrire un script tel que :

```
if [ "${AXIS_NAME}" = "Windows" ]
then
    # Commande Windows
else
    # Commande autre
fi
```

V-C-3 - Généralisation

Évidemment, différencier la plateforme dans les scripts ce n'est pas la meilleure solution, car vous obtiendrez une répétition du code (code de compilation Windows et code de compilation Linux). En jouant sur la configuration des chemins des outils et avec un axe approprié, vous pouvez obtenir un résultat plus générique et aussi, plus simple.



Pour restreindre l'exécution des tâches à certaines valeurs de vos axes, il suffit d'ajouter des labels à vos esclaves. Par exemple, si vous souhaitez faire une compilation vs2015 (Visual Studio) uniquement pour Windows, créez un axe contenant vs2015 et assignez à votre machine Windows l'étiquette vs2015.



*Si cette méthode ne vous convient pas, vous pouvez toujours opter pour la création d'un projet par plateforme. Toutefois, cela peut rendre la gestion des projets plus compliquée (non-centralisation des informations, nécessité de modifier à plusieurs endroits). Une autre solution à mi-chemin existe grâce au module **Multijob**.*

V-C-4 - Remarques

Même si un job freestyle permet une configuration rapide, les tâches complexes demandent de mettre en place une logique avancée ressemblant au déroulement d'un algorithme.

Aussi, au cours de la vie du projet, l'architecture de ce dernier peut évoluer et conjointement, les tâches liées à celui-ci devront évoluer. S'il arrive que vous deviez réexécuter vos tâches sur une ancienne version, la tâche ayant été modifiée, celle-ci échouera certainement.

C'est là qu'entre en jeu Pipeline. Celui-ci offre un langage de programmation simple, mais complet pouvant être sauvegardé sur le logiciel de contrôle de version et donc, synchronisé avec les différentes versions de votre projet.

V-D - Job Pipeline

Pipeline est un module de plus en plus populaire permettant de configurer les tâches au travers d'un code. Ce dernier peut être sauvegardé avec le reste du code source de votre projet, sur votre serveur de contrôle de version.

V-D-1 - Création d'un job

Encore une fois, cela passe par « Nouveau Item ». Évidemment, le type de tâche à prendre sera « Pipeline ».

V-D-2 - Configuration

La page de configuration reprend des éléments de celle pour les tâches free-style tout en offrant une version épurée. En effet, mise à part la configuration des paramètres, des options globales de la tâche et de la répétitivité de la tâche, on arrive tout de suite à la configuration du pipeline et cela, à travers un simple éditeur de script.

V-D-2-a - Script Pipeline

Les pipelines reposent sur le langage de script **Groovy**. La documentation dédiée à Jenkins est disponible sur [cette page](#).

Les scripts Pipeline apportent les grandes notions suivantes :

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

Les scripts peuvent s'écrire de deux manières différentes, soit en utilisant une syntaxe de script, soit une syntaxe déclarative. Voici l'architecture d'un Pipeline scripté :

```
node {  
    stage('Build') {  
        //  
    }  
    stage('Test') {  
        //  
    }  
    stage('Deploy') {  
        //  
    }  
}
```

Et la même chose, avec la syntaxe déclarative :

```
pipeline {  
    agent any  
    stages {
```

```

    stage('Build') {
        steps {
            //
        }
    }
    stage('Test') {
        steps {
            //
        }
    }
    stage('Deploy') {
        steps {
            //
        }
    }
}

```

V-D-2-b - Restreindre l'exécution d'une tâche

Tout comme avec les jobs freestyle, vous avez la possibilité de restreindre l'exécution d'une tâche à une machine (ou un lot de machines) précise grâce aux labels :

- syntaxe script :

```

node (label: 'slave') {
    ...
}

```

- syntaxe déclarative :

```

pipeline {
    agent {label 'slave'}
    stages {
        ...
    }
}

```

V-D-2-c - Paralléliser des tâches

La force du pipeline est aussi de permettre la parallélisation des tâches. Pour cela, le mot clé `parallel` a été implémenté :

- syntaxe script :

```

parallel (
    "stream 1" : {
        node ("windows") { ...
    }
    "stream 2" : {
        node ("linux") ...
    }
}

```

- syntaxe déclarative :


```

pipeline {
    agent none
    stages {
        stage('Run Tests') {
            parallel {
                stage('Test On Windows') {
                    agent {
                        label "windows"
                    }
                    steps {
                        bat "run-tests.bat"
                    }
                }
            }
        }
    }
}

```

V-D-3 - Écrire son script

V-D-3-a - Module Blue Ocean

 *Blue Ocean n'est utilisable qu'avec certains logiciels de contrôle de version. Toutefois, même si votre projet n'a pas été créé avec Blue Ocean, vous pouvez accéder à l'interface de ce dernier à tout moment.*

V-D-3-b - Générateur de code

V-D-3-c - Générateur déclaratif

V-D-3-d - Documentation

V-D-4 - Script ou déclaratif

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright © 2019 Alexandre Laurent. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

<https://alexandre-laurent.developpez.com/tutoriels/initiation-ci-jenkins/>

V-D-5 - Configuration du projet BouskNet

Ainsi, la compilation du projet BouskNet peut s'effectuer avec ce script :

```
node {
  stage('Checkout') { // for display purposes
    cleanWs()

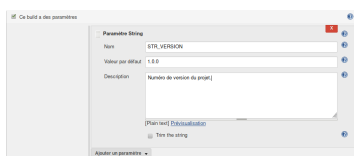
    checkout([$class: 'MercurialSCM',
              credentialsId: '',
              revision: 'Lib',
              source: 'https://bitbucket.org/Bousk/bousknet'])
  }
  stage('Premake') {
    dir('./NetworkLib') {
      sh "${JENKINS_HOME}/Tools/premake/premake5 gmake"
      // Hack pour compiler sur une plateforme ARM
      sh "sed -i -e 's/m64/marm/g' ./Projects/gmake/Network.make"
      sh "sed -i -e 's/m64/marm/g' ./Projects/gmake/Samples_TCP_Client.make"
      sh "sed -i -e 's/m64/marm/g' ./Projects/gmake/Samples_TCP_Server.make"
      sh "sed -i -e 's/m64/marm/g' ./Projects/gmake/Samples_UDP_HelloWorld.make"
    }
  }
  stage('Build') {
    dir('./NetworkLib/Projects/gmake/') {
      sh "make"
    }
  }
}
```



Avant de commencer à configurer un projet dans Jenkins, assurez-vous de savoir le compiler en ligne de commande. Ainsi, cela vous permettra de mieux maîtriser le processus, mais aussi de vérifier que tout est prêt pour compiler ce projet.

V-E - Paramètres

Les paramètres permettent d'ajouter des paramètres à votre tâche qui seront déterminés au moment de lancer la compilation.



Configuration d'un paramètre



Spécification d'un paramètre au lancement d'une tâche

Dans votre tâche, les paramètres s'utilisent comme des variables. Du coup, il suffit d'écrire `${MON_PARAM}` ou `%MON_PARAM%` suivant que vous utilisiez les scripts Shell ou les scripts Batch. Il est aussi possible d'utiliser les paramètres dans les informations de checkout, notamment pour utiliser une branche spécifique définie par un paramètre.

Si vous lancez votre tâche à travers une URL, vous pouvez spécifier les paramètres comme des arguments :

```
http://localhost:8080/job/myJob/buildWithParameters?param1=param1value
```

Plusieurs types de paramètres s'offrent à vous et déterminent l'interface affichée à l'utilisateur lors du lancement du build :

- mot de passe ;
- chaînes de caractères ;
- booléen ;
- choix ;
- exécution ;
- identifiants ;
- fichier ;
- texte

V-F - Lancer la compilation par requête Web

Il est possible de lancer la compilation sans passer par l'interface graphique. Voici l'URL permettant de lancer une tâche :

```
http://serveur:8080/job/NomDuJob/build
```

Vous pouvez aussi ajouter une attente avec le paramètre delay :

```
http://serveur:8080/job/NomDuJob/build?delay=30sec
```

La gestion des paramètres de compilation nécessite de passer par la page buildWithParameters :

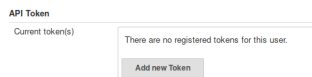
```
http://serveur:8080/job/NomDuJob/buildWithParameters?param1=param1value
```



Le passage de paramètres doit être réalisé en POST.

V-F-1 - Lancement avec authentification

Si votre serveur requiert une authentification, vous devez obtenir un jeton pour l'utilisateur souhaitant scripter ses demandes d'exécution de tâches. Celui-ci peut être obtenu dans la page de configuration de l'utilisateur.



Une fois obtenu, vous devez obtenir l'accès au serveur avec :

```
curl "http://USER:TOKEN@SERVER/crumbIssuer/api/xml?xpath=concat(//crumbRequestField,%22:%22,//crumb)"
```

Cette requête devrait vous permettre d'obtenir un jeton de sécurité à inclure dans vos prochaines requêtes :

```
curl -X POST -H CRUMB http://USER@SERVER/job/TACHE/build
```

- CRUMB : le résultat de la première requête ;
- USER : le nom de l'utilisateur dans Jenkins ;
- TOKEN : le jeton correspondant à l'utilisateur ;
- SERVER : le nom du serveur ;
- TACHE : le nom de la tâche à lancer.

VI - Gestion des accès

Comme vous l'avez compris, Jenkins permet de paramétrer ce que va exécuter 1 à N machines sur votre réseau. Il est donc nécessaire d'être précautionneux sur qui aura accès à l'administration de Jenkins et de ses tâches. Comme souvent lors de la mise en place d'un serveur, il faut configurer un compte administrateur avec un mot de passe fort et limiter, voire désactiver l'accès libre. En effet, il ne faut pas donner la possibilité à n'importe qui de pouvoir configurer des tâches, car ces dernières pourraient être utilisées pour effectuer des actions embarrassantes.

Aussi, Jenkins intègre un système d'identifiants qui peut être lié à un LDAP, un active directory ou d'autres solutions d'authentification unique. Vous pouvez retrouver ces différentes possibilités au travers du gestionnaire de plugins.

Vous pourrez donc configurer comment les utilisateurs accèdent à Jenkins à travers la page de sécurité globale. Les identifiants de chacun (et leurs droits) se configurent dans la page « Identifiants ». Le plugin « **Matrix Authorization Strategy** » permet d'étendre les possibilités dans les droits d'accès d'un utilisateur.

*La sécurité ne doit jamais être prise à la légère et Jenkins peut permettre l'intrusion dans votre système et votre réseau, comme le prouve **cette actualité**.*



Ne permettez pas d'accéder à Jenkins depuis Internet si vous n'en avez pas besoin.

VII - Trucs et astuces

VII-A - Configuration des esclaves

Les esclaves sont les entités qui permettent aux tâches de s'exécuter. Vous pouvez les configurer individuellement en cliquant sur le nom de l'esclave sur la page principale, puis sur « Configurer » dans la colonne de gauche :

The screenshot shows the Jenkins 'Configure' page for a node named 'Windows'. The page is divided into several sections:

- General:** Nom (Windows), Description (ERROR), Nb d'exécuteurs (1), Répertoire de travail du système distant (C:\Jenkins), Étiquettes (vs2015), Utilisation (Utiliser cet esclave autant que possible), Méthode de lancement (Launch agent by connecting it to the master).
- Disponibilité:** Keep this agent online as much as possible.
- Propriétés du nœud:**
 - ☐ Disable deferred wipeout on this node
 - ☐ Emplacement des outils
 - ☒ Variables d'environnement
 - Liste des paires clé-valeur:
 - nom: MAKE_EXE
 - nom: MAKE_EXE_FAIL, valeur: C:\Program Files\Microsoft Visual Studio 14.0\VC\bin\nmake
 - nom: PREMAKE_EXE, valeur: .\premake5.exe

Buttons: Enregistrer, Supprimer, Ajouter.

Page générée: 21 jul 2019 16:58:48 CEST REST API Jenkins ver. 2.178.2

Vous pourrez y définir le nom, ou encore la politique de disponibilité de l'esclave.

VII-A-1 - Nombre de fils d'exécution

Vous pouvez changer le nombre d'exécuteurs (c'est-à-dire, le nombre de tâches pouvant être exécutées sur une machine en parallèle) pour correspondre au mieux à la puissance de la machine esclave.

VII-A-2 - Étiquettes

Des étiquettes peuvent être assignées aux esclaves. Comme vu dans la section de mise en place de **tâches multiconfigurations**, elles permettent de préciser quelle machine peut exécuter quelles tâches.



Lorsque vous précisez les étiquettes sur lesquelles les tâches peuvent être exécutées, vous pouvez utiliser des opérateurs logiques tels que && et || pour le ET logique et le OU logique, respectivement.

VII-A-3 - Variables d'environnement

Il est possible de définir/redéfinir des variables d'environnement spécifiques à la machine. Ainsi, les tâches pourront s'exécuter même si les outils sont installés différemment des autres machines de votre parc.

VII-B - Configuration des tâches

VII-B-1 - Ignorer des commits

Certains modules récupérant les sources de vos projets permettent de ne pas déclencher la tâche si un message particulier est écrit dans le message de commit. Vous pouvez retrouver cette option dans la section « gestion de code source », bouton « avancé » de la configuration d'une tâche. Jenkins vous laissera ainsi renseigner une expression régulière, qui lorsque vérifiée, fera que Jenkins ignore ce commit.

De la même façon, vous pouvez aussi ignorer les commits de certains utilisateurs.

VII-B-2 - Artefacts

Les artefacts sont les fichiers générés lors de l'exécution d'une tâche. En effet, par défaut (c'est-à-dire sans artefact) Jenkins ne sauvegardera que les journaux de l'exécution de la tâche et cela, seulement si vous ne supprimez pas les anciennes exécutions.

Par conséquent, vous n'avez accès qu'à un nombre limité de journaux et seul le dernier espace de travail (workspace) sera accessible.

En configurant l'archivage des artefacts, vous pouvez conserver les fichiers générés (tels que les exécutables) ou encore, les journaux. Cette option se retrouve dans les étapes postexécutions de la tâche.

VII-B-3 - Empêcher l'exécution de deux tâches en parallèle

Avec le nombre grandissant de tâches sur votre serveur Jenkins, il arrivera un moment où vous souhaiterez bloquer l'exécution d'une tâche si une autre est en cours d'exécution. En effet, une tâche particulière pourrait impacter le bon déroulement d'une autre.

Pour ce faire, vous devez installer le module « **Build Blocker** ». Ce dernier ajoute une entrée dans la configuration des tâches permettant de définir si l'exécution de cette tâche est bloquée par une autre tâche.

La liste des tâches bloquantes peut être définie grâce à une expression régulière.

VIII - Conclusion

Jenkins est un outil puissant et essentiel dans un projet, peu importe la taille de ce dernier. Au cours de cet article, nous avons vu comment le configurer et appris à mettre en place notre premier projet multiplateforme.

On remarque aussi que la migration de la politique du projet Jenkins en considérant la configuration des tâches comme un code permet de simplifier la gestion du projet ainsi que son administration. Ainsi, aujourd'hui, je conseille à tous ceux qui démarrent Jenkins d'utiliser Pipeline et non pas un projet « free style ».

IX - Ressources supplémentaires

- **Le serveur d'intégration continue** du projet Jenkins, fonctionnant grâce à Jenkins ;
- **Récapitulatif des bonnes pratiques** Jenkins ;
- **Syntaxe des scripts Pipeline** ;
- **Liste des modules**.

X - Remerciements

Je tiens à remercier **Marco46** et Franck Talbart, ainsi que **6BerYeti** pour leurs relecture et apports à cet article. De plus, je remercie **ClaudeLELOUP** pour sa relecture orthographique.