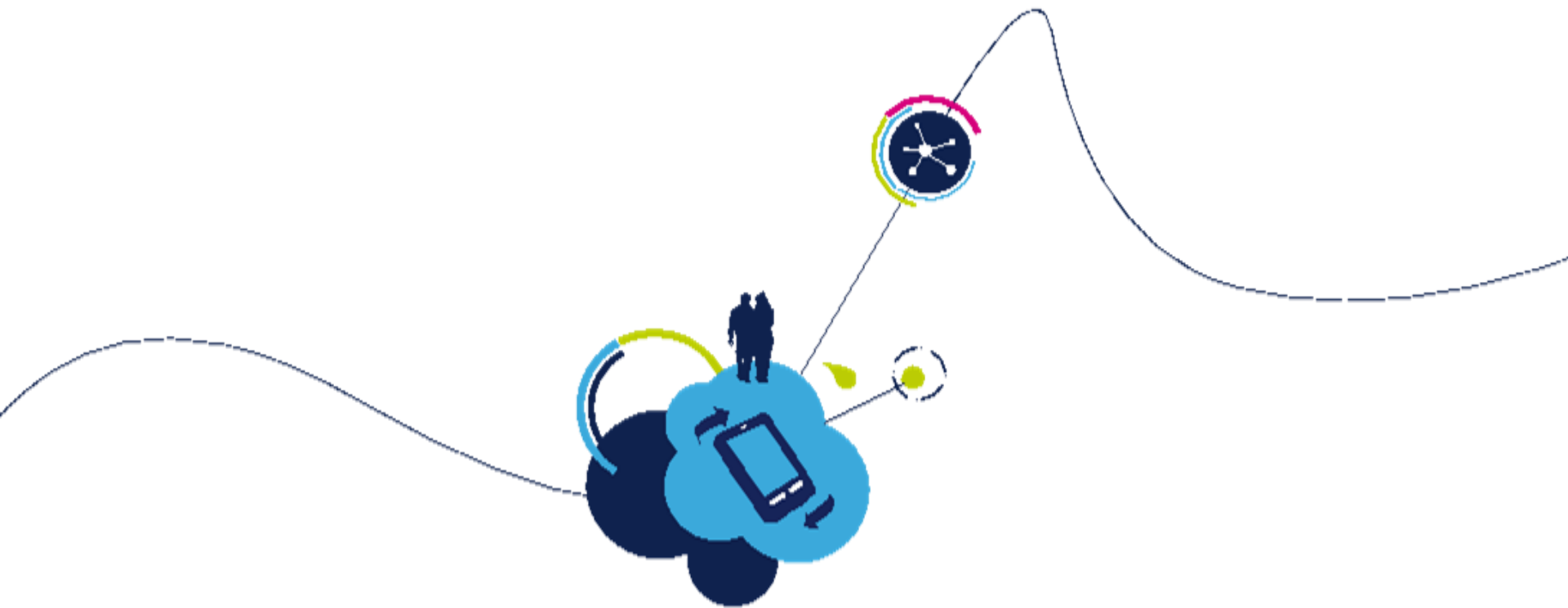


Intégration continue avec Jenkins

Pré requis

Les étudiants doivent se familiariser avec d'autres technologies utilisées dans cette formation:

- Docker
- Git/Github
- Apache Maven
- Apache Groovy



CONSTRUIRE UNE APPLICATION AVEC JENKINS

Sommaire

- Débuter avec Jenkins
- Gérer Jenkins
- Les plugins de Jenkins
- Extension des capacités
- Les 'jobs' Jenkins
- Les 'builds' Jenkins
- Les artefacts dans Jenkins
- Gestion de source code avec Jenkins
- Déclenchement des 'builds'
- Exécution des tests avec Jenkins

VOS DEBUTS AVEC JENKINS

Pour commencer avec Jenkins, il faut:

- Couvrir les exigences
- Obtenir Jenkins
- Déployer Jenkins
- Acheter l'administration de base de Jenkins

EXIGENCES MINIMALES

Jenkins fonctionne sur:

- Unix/BSD
- Linux
- Mac OS

Windows

La version de Java requise :

- 7 est la version minimum.
- **8 recommandé.**

EXIGENCES D'EXÉCUTION: JVM

La Java Virtual Machine (JVM) doit être configurée :

- Taille du tas de mémoire : `-Xms1g -Xmx2g`
- Permgen sur JDK7 uniquement : `-XX:Maxpermsize=256m`
- G1 garbage collector pour le tas > 4Gb : `-XX:+Useg1gc`
- Vérifiez vos documentations de mémoire Java

EXIGENCES D'EXÉCUTION: OS

Attention aux limites du système d'exploitation

- Max fichiers ouverts
- Processus à fourche max
- Réglage des réseaux (taille des paquets, délais TCP)

OBTENIR JENKINS

Jenkins est distribué sur de nombreux canaux:

- Fichier WAR
- Paquets natifs du système d'exploitation (RPM, DEB...)
- Image Docker
- Modèles Cloud (AWS, Azure...)

Commencer par <https://jenkins.io/>

EXECUTER JENKINS: WAR

Lors de l'utilisation de la distribution Web Application Archive (WAR) de Jenkins, il est exploitable:

- Comme application autonome
- Dans un conteneur servlet

EXECUTER JENKINS : STANDALONE WAR

Exécuté à partir de la ligne de commande

- Utilise un serveur d'application intégré (Jetty)
- Il offre des fonctionnalités supplémentaires (redémarrer à partir de l'interface web,...)
- `$ java ${JAVA_OPTS} -jar jenkins.war ${JENKINS_OPTS}`

EXECUTER JENKINS : STANDALONE WAR

Autres options utiles :

--prefix \$PREFIX (par défaut : /)

Exécute Jenkins pour inclure le \$PREFIX à la fin de l'URL

--httpPort \$PORT (par défaut : 8080)

Jenkins écoute sur le port \$PORT.

--httpListenAddress \$HTTP_HOST (par défaut : 0.0.0.0)

Lie Jenkins à l'adresse IP représentée par \$HTTP_HOST.

--logfile \$LOGFILE

écrire dans \$LOGFILE au lieu de stdout

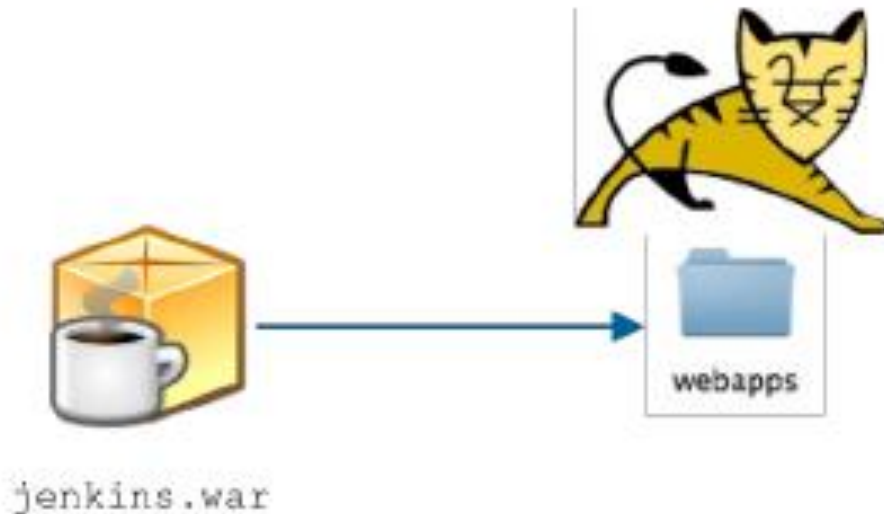
Gestion des signaux :

- Réagit à SIGTERM et SIGINT pour ordonner l'arrêt propre.
- Lors de la réception de SIGALRM, le fichier journal est rouvert.
Autoriser la rotation du journal.

EXÉCUTER JENKINS SUR LE SERVEUR D'APPLICATION

Déployer WAR de Jenkins sur un serveur d'application existant:

- Convenable pour les infrastructures existantes.
- Il suffit de déployer le fichier jenkins.war de la manière habituelle.



INSTALLATION JENKINS : PAQUETS LINUX

Paquets natifs disponibles pour les principales distributions Linux :

- RPM pour les familles Redhat
- Deb pour Debian/Ubuntu

Beaucoup d'autres : Gentoo, Opensuse...

Pattern avec les gestionnaires de paquets :

- Ajouter le dépôt de paquets Jenkins
- Installer et le démarrer

Exemple pour la famille Redhat :

```
# Add the Jenkins Yum Package Repository
$ wget -O /etc/yum.repos.d/cje.repo "http://<...>/rpm/jenkins.repo"
$ rpm --import "http://<...>/rpm/jenkins-ci.org.key"
# Install it
$ yum install jenkins
# Start it
$ service jenkins start
```

EN SAVOIR PLUS SUR LES PAQUETS NATIFS LINUX

- Basé sur jenkins.war (avec Jetty Application Server intégré)
- Avantages:
 - Créer des utilisateurs jenkins
 - Définit les scripts de service (init.d, upstart ou systemd)
 - Les fichiers de configuration suivent les conventions natives.
 - Offre la rotation des logs nativement.
- Où sont les fichiers ?
 - Les paramètres suivent le système de maintenance :
/etc/default/jenkins ou /etc/sysconfig/jenkins pour init. d /upstart
/etc/systemd/system/... pour Systemd
 - \$JENKINS_HOME, le "Data Folder" de Jenkins

INSTALLATION JENKINS : WINDOWS

Exécuter l'une des opérations suivantes :

- setup.exe
- jenkins.msi si . NET 2.0 runtime est déjà disponible

- Installer Jenkins comme un service Windows:

Tous les fichiers vont dans %JENKINS_HOME%

- Le fichier de configuration est en %JENKINS_HOME%
config.xml:

Attention! Il peut être écrasé par Jenkins lui-même.

INSTALLATION JENKINS : DOCKER

- Image Docker officielle sur Dockerhub
- Intégration native avec l'infrastructure Docker
- Versions Jenkins de Security scanné et Long Term Support (LTS)
- Fournit des utilitaires pour l'installation de plugins, la préconfiguration...
- Extensible pour en construire votre propre image Docker
- Basé sur Jenkins WAR autonome
- https://hub.docker.com/_/jenkins/

ACCÈS À LA GESTION JENKINS

- Gérer le lien de la page Jenkins dans le menu de gauche
- Visible uniquement par les utilisateurs autorisés (administrateur)
- Lien direct : `http://JENKINS_URL>/manage`



MESSAGES DE GESTION JENKINS

La page de gestion de Jenkins peut afficher des alertes sur sa configuration comme :

- Problèmes de diagnostic automatique comme une mauvaise configuration du Reverse proxy.
- Problèmes de sécurité
- Mises à jour de Jenkins Core disponibles

Manage Jenkins



The screenshot shows the 'Manage Jenkins' interface with three alerts in a light blue box. The first alert has a red circle icon and states 'It appears that your reverse proxy set up is broken.' with 'More info' and 'Dismiss' buttons. The second alert has a yellow triangle icon and states 'New version of Jenkins (1.542.18.3) is available for [download](#) ([changelog](#)).'. The third alert has a yellow triangle icon and states 'Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.' with 'Setup Security' and 'Dismiss' buttons.

It appears that your reverse proxy set up is broken. [More info](#) [Dismiss](#)

New version of Jenkins (1.542.18.3) is available for [download](#) ([changelog](#)).

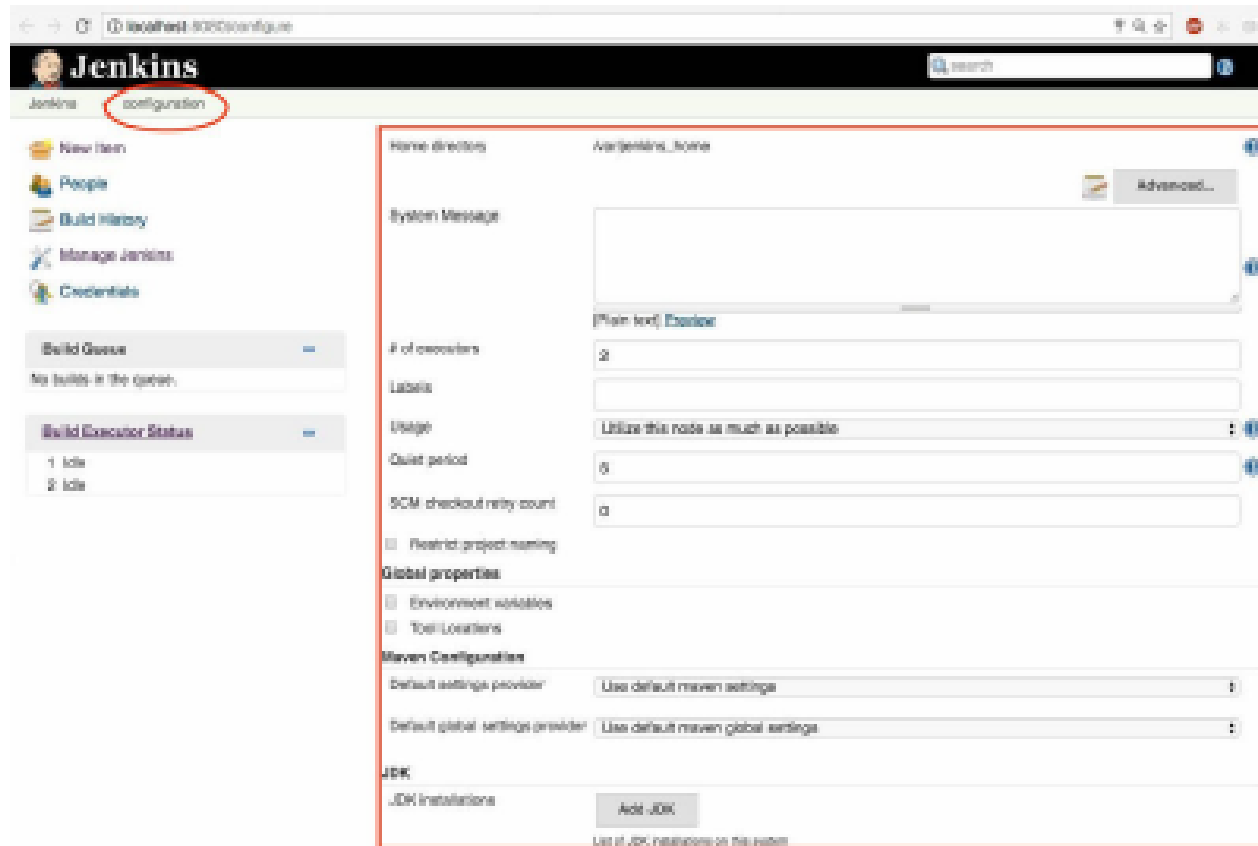
Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse. [Setup Security](#) [Dismiss](#)

GESTION JENKINS

- ✓ Configuration principale :
 - Configurer le système, la console de script, gérer les anciennes données
- ✓ Plugin Management, Jenkins CLI, Manage Nodes
- ✓ Informations principales :
 - Information sur le système, log système, statistiques de chargement, À propos de Jenkins
- ✓ Sections relatives à la sécurité :
 - Configurations de sécurité globales, Gérer les informations d'identification, Approbation du script en cours
- ✓ Cycle de vie du service Jenkins :
 - Configuration de rechargement, préparation à l'arrêt

Configurer le système

- Options de configuration Admin
- Configuration des outils (Jenkins ≥ 2.0 : section dédiée)



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo and a search bar. Below the navigation bar, the 'configuration' link is highlighted with a red circle. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main content area is titled 'Home directory' and 'Advanced...'. It contains several configuration sections: 'System Message' with a text area and a 'Plain text' button; 'Number of executors' set to 2; 'Labels' set to 'Utilize this node as much as possible'; 'Quiet period' set to 0; 'SCM checkout retry count' set to 0; 'Global properties' with checkboxes for 'Environment variables' and 'Tool Locations'; 'Maven Configuration' with dropdowns for 'Default settings provider' and 'Default global settings provider'; and 'JDK' with a table for 'JDK installations' and an 'Add JDK' button. The bottom of the page shows a link to 'List all JDK installations on the system'.

Options d'administration

Principales configurations d'administration :

- Jenkins Options de service : Jenkins URL, Messages,
- Jenkins Master Node (exécuteurs, étiquettes, usages de construction...)
- Variables d'environnement du Jenkins Master



This screenshot shows the 'Configure' page for a Jenkins Master Node. The left sidebar contains the following sections: 'Home directory' (set to /var/jenkins_home), 'System Message' (with a text area and a 'Plain text' button), '# of executors' (set to 2), 'Labels' (empty), 'Usage' (set to 'Utilize this node as much as possible'), 'Quiet period' (set to 5), 'SCM checkout retry count' (set to 0), 'Restrict project naming' (checkbox), and 'Global properties' (checkbox). The main content area shows the 'Advanced...' button and a 'Plain text' button.



This screenshot shows the 'Jenkins Location' configuration page. It includes the following fields: 'Jenkins URL' (set to https://ci.mycompany.org:8080/), 'System Admin e-mail address' (set to admin-jenkins@mycompany.org), 'SSH Server' (checkbox), and 'SSH Port' (set to Fixed: 2200, with options for Random and Disable).

Options d'administration

Principales configurations d'administration :

- Jenkins Options de service : Jenkins URL, Messages,
- Jenkins Master Node (exécuteurs, étiquettes, usages de build...)
- Variables d'environnement du Jenkins Master



This screenshot shows the 'General' configuration tab for a Jenkins instance. The 'Home directory' is set to '/var/jenkins_home'. The 'System Message' field is empty, with a 'Plain text' button and a 'Preview' link below it. The 'Advanced...' button is visible. The 'Number of executors' is set to 2. The 'Labels' field is empty. The 'Usage' dropdown is set to 'Utilize this node as much as possible'. The 'Quiet period' is set to 5 minutes. The 'SCM checkout retry count' is set to 0. There are checkboxes for 'Restrict project naming' and 'Global properties', with 'Environment variables' listed below.

Home directory: /var/jenkins_home

System Message: [Empty text area]

of executors: 2

Labels: [Empty text field]

Usage: Utilize this node as much as possible

Quiet period: 5

SCM checkout retry count: 0

☐ Restrict project naming

Global properties

☐ Environment variables



This screenshot shows the 'Location' configuration tab for a Jenkins instance. The 'Jenkins URL' is set to 'https://ci.mycompany.org:8080/'. The 'System Admin e-mail address' is set to 'admin-jenkins@mycompany.org'. The 'SSH Server' section shows the 'SSH Port' set to 'Fixed: 2200', with options for 'Random' and 'Disable' also visible.

Jenkins Location

Jenkins URL: https://ci.mycompany.org:8080/

System Admin e-mail address: admin-jenkins@mycompany.org

SSH Server

SSH Port: Fixed: 2200

OPTIONS D'OUTILS

Configurations d'outils :

- Dépend des plugins installés
- Emplacements des outils (JDK, Mavens, etc.)
- Configuration des outils (par défaut) pour les outils comme SSHD, Scms

The screenshot displays the 'Options' dialog box for tool configurations, organized into three main sections: JDK, Ant, and Maven.

JDK Section:

- JDK Installations:** A list of installed JDKs. Two entries are visible:
 - JDK 10:** Name: jdk10, JAVA_HOME: /usr/local/java/jdk10. It has an 'Install automatically' checkbox (checked) and a 'Delete JDK' button.
 - JDK 7:** Name: jdk7, JAVA_HOME: /usr/local/java/jdk7. It also has an 'Install automatically' checkbox (checked) and a 'Delete JDK' button.
- Add JDK:** A button to add new JDK installations.
- List of JDK installations on this system:** A label indicating the source of the listed installations.

Ant Section:


- Ant Installations:** A section with an 'Add Ant' button and a label 'List of Ant installations on this system'.

Maven Section:

- Maven Installations:** A list of installed Maven versions. One entry is visible:
 - Maven 3.3:** Name: maven-3.3, MAVEN_HOME: /usr/local/apache-maven/maven-3.3.9. It has an 'Install automatically' checkbox (checked) and a 'Delete Maven' button.

GESTION : CONSOLE DE SCRIPT

- Utiliser des scripts Apache Groovy dans Jenkins JVM
- Gestion en tant que code : Configurez programmatiquement Jenkins
- Modèle Jenkins complet que Groovy manipule
- Javadoc : <http://javadoc.jenkins-ci.org/>
- Utilisez Script Console pour exécuter vos scripts :



Script Console

Type in an arbitrary [Groovy script](#) and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

```
1 // Adding a Maven Installation
2
3 mavenTools=Jenkins.instance.getExtensionList(hudson.tasks.Maven.DescriptorImpl.class)[0];
4
5 mavenInstallations=(mavenTools.installations as List);
6
7 mavenInstallations.add(new hudson.tasks.Maven
8     .MavenInstallation("MAVEN1", "/opt/apache-maven-1", []));
9
10 mavenTools.installations=mavenInstallations;
11
12 mavenTools.save();
```

Run

GESTION : INFORMATION SUR LE SYSTÈME

- URL directe : `http://<JENKINS_URL>/systemInfo`
- Fournit un aperçu graphique du système :
- Propriétés du système Java (`java.io.tmpdir`, `user.dir`...)
- Variables d'environnement du système (`$PATH`, `$HOME`...)
- Présentation de l'état du plugin installé
- Accès Threaddump monitor () pour la JVM Jenkins

`http://JENKINS_URL>/systemInfo`

`http://JENKINS_URL>/threadDump`

System Properties

Name ↓	Value
awt.toolkit	sun.awt.X11.XToolkit
executable-war	/usr/share/jenkins/jenkins.war
file.encoding	UTF-8
file.encoding.pkg	sun.io
file.separator	/
hudson.diyChunking	true
java.awt.graphicsenv	sun.awt.X11GraphicsEnvironment
java.awt.headless	true
java.awt.printerjob	sun.print.PSPrinterJob
java.class.path	/usr/share/jenkins/jenkins.war
java.class.version	52.0
java.endorsed.dirs	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/endorsed
java.ext.dirs	/usr/lib/jvm/java-8-openjdk-amd64/jre/lib/ext:/usr/java/packages/libext
java.home	/usr/lib/jvm/java-8-openjdk-amd64/jre
java.io.tmpdir	/tmp
java.library.path	/usr/java/packages/lib/amd64:/usr/lib/x86_64-linux-gnu:/lib/x86_64-linux-gnu:/usr/lib/x86_64-linux-gnu:/usr/lib:/lib:/usr/lib

GESTION : INFORMATION SUR LE SYSTÈME

- URL directe : `http://<JENKINS_URL>/log/`
- Offre l'interface graphique pour accéder aux différents journaux
- Utilisez-le pour trouver des messages d'erreur pour votre système lorsque l'interface utilisateur graphique est accessible.
- Un nouveau log provider peut être ajouté



GESTION : CYCLE DE VIE DES SERVICES

- Gestion des options utilisées pour démarrer/arrêter/recharger le service Jenkins
- Recharger la configuration à partir du disque :
 - Lire à nouveau la configuration de \$JENKINS_HOME
 - Puis recharger Jenkins Service
- Préparation à l'arrêt :

Gère l'arrêt gracieux (fin de travail propre)

PLUGINS JENKINS : EXTENSION DES CAPACITÉS

- Jenkins peut être étendu par des plugins
- Beaucoup de fonctions Jenkins "classiques" sont des plugins
- Plusieurs plugins déjà disponibles (Consultez le site Web de Jenkins)
- Exemples de plugins:
 - Source code management tools
 - Build Tools
 - Reporting tools
 - Code coverage, static code analysis
 - Online source code browsers
 - Issue tracker
 - Notification tools
 - Views and UI customizations
 - Distributed builds

QUE SONT LES PLUGINS ?

- Un plugin est au format . Hpi
- Il s'agit d'un format JAR avec quelques conventions spéciales (par ex. pas de web.xml)
- Maven sait comment gérer le format hpi
- Certains plugin peuvent être trouvés en utilisant jpi
- Les plugins sont des artefacts versionnés :
 - Vous pouvez mettre à jour et le downgrade.
 - Ils peuvent avoir des dépendances (obligatoires ou facultatives)
 - Les plugins sont situés dans `${JENKINS_HOME}/plugins:`
Fichiers hpi et versions non archivées

INSTALLATION DE PLUGINS

The screenshot shows the Jenkins web interface. On the left sidebar, the 'Manage Jenkins' link is circled in red. The main content area is titled 'Manage Jenkins' and contains a list of administrative tasks. The 'Manage Plugins' item is also circled in red. Below it, the 'System Information' item is highlighted with a yellow background.

Manage Jenkins

- Configure System**
Configure global settings and paths.
- Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
- Configure Analytics**
Configure the storage and reporting of Analytics.
- Reload Configuration from Disk**
Reload all the loaded data in memory and reload everything from the system. Useful when you modified config files directly on disk.
- Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**
- System Information**
Displays various environmental information to assist trouble-shooting.
- System Log**
System log captures output from Java, ch11, Logging output related to Jenkins.
- Load Statistics**
Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**
Access/manage Jenkins from your shell, or from your script.
- Script Console**
Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- High Availability Status**
Monitor the status of the Jenkins Enterprise HA cluster.
- Plugin Usage**
Summarizes usage of installed plugins insofar as that can be determined. Also status, report bugs and suggestions to [CloudBees Support](#).

Build Queue
No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle

QUELS SONT LES PLUGINS DISPONIBLES ?

Site principal : <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>
Pour vos Jenkins :




The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a breadcrumb trail: "Jenkins > Plugin Manager". Below this, there are two links: "Back to Dashboard" with a green arrow icon and "Manage Jenkins" with a wrench icon. On the right, there's a "Filter:" input field. The main content area has four tabs: "Updates", "Available", "Installed", and "Advanced". The "Available" tab is selected and highlighted in yellow. A yellow callout bubble points to this tab with the text "List of available plugins". Below the tabs, there's a table of available plugins. The table has two columns: "Name" and "Version". The first row is for "CCM Plugin" with version "3.1". The second row is for "ExCoo Runner plugin" with version "1.1". The third row is for "MSBuild Plugin" with version "1.20". The fourth row is for "MSTest plugin" with version "0.10". Each row has a checkbox in the "Name" column and a description in the "Version" column.

Name	Version
CCM Plugin This plug-in generates reports on cyclomatic complexity for .NET code.	3.1
ExCoo Runner plugin ExCooCond.exe support plugin.	1.1
MSBuild Plugin This plugin allows you to use MSBuild to build .NET projects.	1.20
MSTest plugin This plugin converts MSTest TFX test reports into JUnit XML reports as it can be integrated with Jenkins's JUnit testcases. You can use MSTestRunner plugin or VsTestRunner plugin to run the test and use this plugin to process the results.	0.10

QUELS SONT LES PLUGINS INSTALLES ?

Filter:

Updates	Available	Installed	Advanced		
Enabled	Name ↓	Version	Previously installed version	Pinned	Uninstall
	Active Directory plugin This plugin enables authentication through Active Directory on Windows environment.	1.42	Downgrade to 1.41	Unpin 	Uninstall
	Amazon Web Services SDK This plugin provides AWS SDK for Java for other plugins.	1.10.26			Uninstall
	Ant Plugin This plugin adds Apache Ant support to Jenkins.	1.2			Uninstall
	Async Http Client This plugin provides a shared dependency on the async-http-client library so that other plugins can co-operate when using this library.	1.7.24			Uninstall

- Lorsque vous mettez à jour un bundle plugin, il est épinglé
- Les plugins épinglés ne sont pas écrasés automatiquement
- Sur le disque, vous verrez un fichier nommé foo.hpi.pinned

INSTALLATION MANUELLE DES PLUGINS : GUI

Updates

Available

Installed

Advanced

Advanced options

HTTP Proxy Configuration

Server

Port

User name

Password

No Proxy Host

Advanced...

Submit

Upload Plugin

You can upload a .jar file to install a plugin from outside the central plugin repository.

File:

Choisissez un fichier

Ajouter fichier existant

Upload

Update Site

URL

Submit

INSTALLATION MANUELLE DES PLUGINS : MISES EN GARDE

- Les noms des plugins peuvent changer : Git Plugin est nommé git
- Les plugins peuvent avoir des dépendances sur d'autres plugins :
Git Plugin a besoin de git-client
- Soyez averti au sujet de la dérogation potentielle avec des plugins épinglés
- La bonne pratique est de mettre à jour fréquemment!
- Les systèmes qui ont besoin de modèles "Immutable" Jenkins fournit des outils:
L'image Docker de Jenkins a un script shell pour télécharger des plugins et dépendances.

LES JOBS JENKINS

- QU'EST-CE QU'UN JOB/PROJET JENKINS
- Les jobs sont au cœur du processus de construction de Jenkins
- Un job Jenkins est un ensemble de tâches (ou « étapes ») définies par l'utilisateur
- Une application peut nécessiter plusieurs jobs
- Un seul emploi peut comporter plusieurs étapes
- Chaque fois qu'un Job/Project est exécuté, il est nommé un Build
- Ainsi, un job a un historique de construction avec des horodatages et des journaux qui vont avec.

ANATOMIE DES PROJETS

Tous les projets partagent ces propriétés communes :

- Un type (voir la diapositive suivante)
- Options de projets globaux :
 - Meta Information : Nom, Description.
 - Gestion de l'historique de construction : conservation des journaux, élimination des anciennes constructions...
 - Gestion de l'orchestration : Attendre entre les compilations, désactiver, attendre...
- Gestion du code source : Utilisons-nous un SCM ?
- Build Triggers : Quand voulez-vous lancer une compilation ?
Manuellement, Périodiquement (type cron), SCM basé sur les événements, Autre basé sur les événements...

Chaque configuration de projet est conservée sous forme de fichier XML config.xml

- Accessible en HTTP sur `http://${JENKINS_URL}/job/${PROJECT_NAME}/config.xml`
- Il est persisté dans le dossier `${JENKINS_HOME}/jobs/${PROJECT_NAME}`

TYPES DE PROJETS JENKINS

- Freestyle Project : type de projet principal
- Maven Project : nécessite un plugin (Avertissement : presque obsolète)
- Pipeline et Multi-Branch Pipeline : nécessite un plugin
- Job externe
- Projet multi-configuration : nécessite un plugin

PROJETS JENKINS: FREESTYLE

- C'est la caractéristique centrale (et la plus flexible) de Jenkins.
- Peut construire n'importe quel type de projet (Ant, Maven, Makefile, script Shell...)
- Composé de "Steps"; qui est extensible par plugins
 - Défini et ordonné par l'utilisateur
 - Exemple : script Shell, script Windows, versions Maven ou Ant...
- Peut avoir des actions «Post-build»
 - Ne fait pas partie du build
 - Exécuter en fonction de l'état du build principal

PROJETS JENKINS : MAVEN

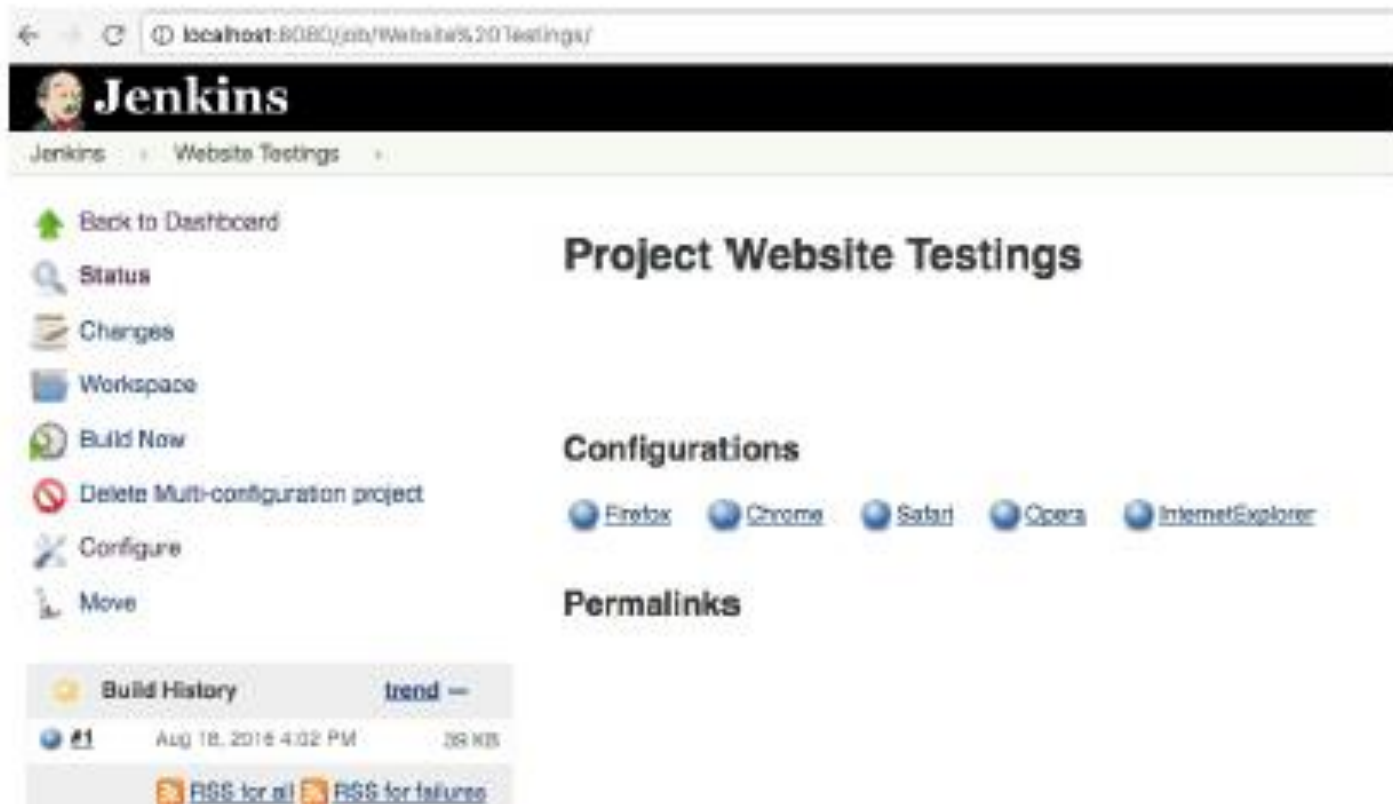
- Optimisé pour les projets Maven 2/3 :
- Jenkins lit le fichier pom.xml pour avoir connaissance du projet Maven
 - Dépendances de Maven
 - Projets multi-modules
- Basé sur un seul Goal Maven
 - Étapes avant et après la construction, qui font partie du build lui-même
 - Configurable (Goals, options Maven)
- Comme pour les jobs freestyle, peut avoir des actions «Post-construction»
 - Ne fait pas partie du build
 - Exécuté en fonction de l'état du build principal

PROJET JENKINS : PIPELINES

- Le standard actuellement avec Jenkins
- Utilise Apache Groovy pour coder le pipeline de build de votre projet
- Configuration textuelle (Groovy) : Partageable et « versionnable »
- Pipeline Groovy Script de :
 - SCM : Jenkinsfile
 - Écriture en ligne, en utilisant le Snippet Generator
- Pas de concept d'étapes de construction, ni d'actions pré/post. Le script définit tous cela.

PROJET JENKINS : MULTI-CONFIGURATION

- Basé sur des projets Freestyle
- Définition d'un ensemble d'axes : ensemble de valeurs défini par l'utilisateur
- Le nom de l'axe injectera la variable d'environnement pour les compilations
Exemple : mise à l'essai de l'application contre tous les JDK
- Les compilations sont exécutées dans différents « sous-projets » (exécution parallélisée par défaut)
- L'axe peut être filtré :
Exemple : "Ne pas tester JDK 6 sur Windows"



BUILD JENKINS : TABLEAU DE BORD

The screenshot shows the Jenkins dashboard interface. On the left, a sidebar contains navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, Support, Credentials, and Pooled Virtual Machines. Below these are three expandable sections: Build Queue (showing 'No builds in the queue'), Build Executor Status (showing 1 idle and 2 busy executors), and Master Metrics (showing 0 min idle because jobs didn't have enough executors). The main area is titled 'Build jobs' and contains a table of build jobs. A callout 'Create a new build job' points to the 'New Item' link. Another callout 'Configure Jenkins' points to the 'Manage Jenkins' link. A third callout 'Scheduled jobs' points to the 'Build Queue' section. A fourth callout 'Start a build job' points to the 'Build jobs' table. The table has columns for Name, Last Success, Last Failure, Last Duration, and Deployed On. It lists five jobs: jenkins-default, jenkins-bootstrap, jenkins-configuration-test, jenkins-metrics, and jenkins-test-test. The footer contains links for Documentation, Support, and the CloudBees Jenkins Platform logo.

Create a new build job

Configure Jenkins

Scheduled jobs

Build jobs

Start a build job

Name	Last Success	Last Failure	Last Duration	Deployed On
jenkins-default	N/A	14 min - 11	62 ms	
jenkins-bootstrap	2 min 43 sec - 62	5 min 23 sec - 62	1 min 25 sec	
jenkins-configuration-test	2 min 18 sec - 62	5 min 18 sec - 62	24 sec	N/A
jenkins-metrics	2 min 18 sec - 62	5 min 0 sec - 62	2 min 18 sec	N/A
jenkins-test-test	5 min 40 sec - 62	N/A	1.2 sec	N/A

Legend: PGG for all, PGG for Jenkins, PGG for Jenkins build

ETAT DU BUILD

Un build peut avoir l'un de ces statuts :

- Exécution (couleur clignotant bleu/gris) : Le build est actuellement en cours d'exécution
- Annulé (gris) : La compilation a été annulée par l'utilisateur ou l'opération administrative
- Success (Bleu) : Le build exécuté sans erreur
- Echech (Rouge) : Le build a échoué avec des erreurs irrécupérables
- Unstable (jaune) : Le build réussi mais certains tests ont échoué

SUIVI DES JOBS DE BUILD

CloudBees Jenkins Enterprise

Search

Jenkins

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Wasted Minutes

0 ms were wasted because you didn't have enough executors.

Quick status on each project

Name	Last Success	Last Failure	Last Duration	Deployed On
connectio-default	N/A	14 min - #1	62 ms	N/A
connectio-fis			1 min 25 sec	N/A
connectio-iss			24 sec	N/A
connectio-mas			2 min 16 sec	N/A
connectio-ver			1.2 sec	N/A

Legend: RSS for all, RSS for failures, RSS for just failed builds

Documentation Support

CloudBees Jenkins Platform

Help us localize this page

Page generated: Mar 19, 2016 11:17:18 AM

REST API Jenkins ver. 1.642.2.2 (CloudBees Jenkins Enterprise 1.5.11)

www.cloudbees.com

SUIVI DES JOBS DE BUILD

[add description](#)

Tous +

S	W	Name ↓	Last Success	Last Failure	Last Duration	Deployed On
		gameoflife-default			62 ms	N/A
		gameoflife-freestyle			1 min 25 sec	N/A
		gameoflife-integration-tests			24 sec	N/A
		gameoflife-metrics			2 min 16 sec	N/A
		gameoflife-web-tests			1,2 sec	N/A

Icon: [S](#) [M](#) [L](#)

Build unstable

[RSS for failures](#) [RSS for just latest builds](#)

Jenkins [DISABLE AUTO REFRESH](#)

[New Item](#) [People](#) [Build History](#) [Project Relationship](#) [Check File Fingerprint](#) [Manage Jenkins](#) [Support](#) [Credentials](#) [Posted Virtual Machines](#)

[add description](#)

Tous +

S	W	Name ↓	Last Success	Last Failure	Last Duration	Deployed On
		gameoflife-default	N/A	14 min - #1	62 ms	N/A
		gameoflife-freestyle	3 min 43 sec - #2	8 min 23 sec - #1	1 min 25 sec	N/A
		gameoflife-integration-tests	2 min 19 sec - #2	8 min 19 sec - #1	24 sec	N/A
		gameoflife-metrics	2 min 16 sec - #2	8 min 0 sec - #1	2 min 16 sec	N/A
		gameoflife-web-tests	8 min 42 sec - #1	N/A	1,2 sec	N/A

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Build Queue [see](#)

No builds in the queue.

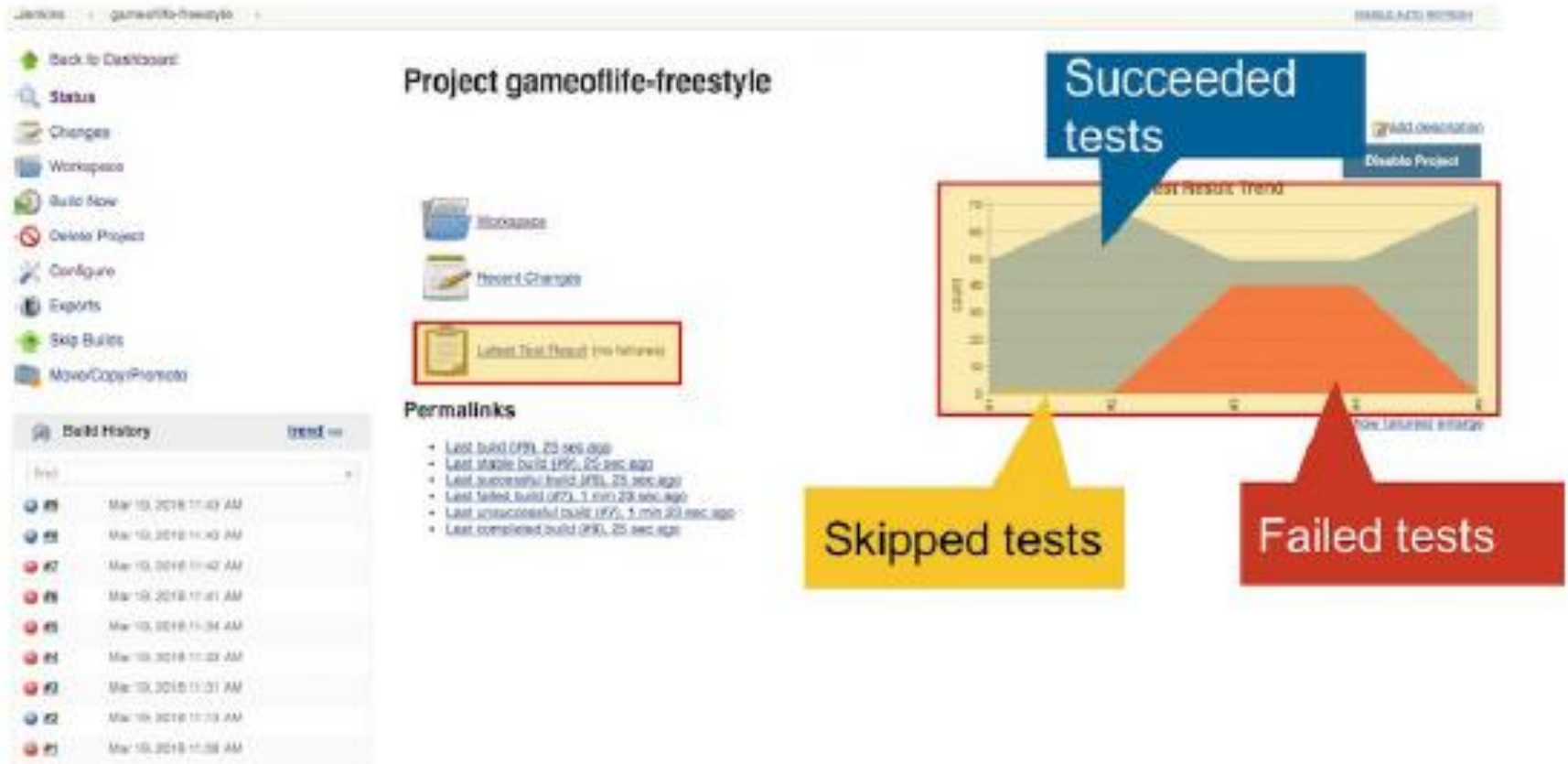
Build Executor Status [see](#)

1. [gameoflife-freestyle](#) #10

2. N/A

Click on progress bar to jump to the console

DETAILS DE BUILD - TESTS



DETAILS DE BUILD - CHANGEMENTS

The screenshot displays the Jenkins Enterprise web interface. On the left, a sidebar contains navigation links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, Exports, Skip Builds, and Move/Copy/Promote. The main content area is titled 'Project gameoflife-freeside'. It features a 'Workspace' section with a 'Recent Changes' button highlighted by a red box and a blue arrow pointing to the 'Changes' panel on the right. Below the workspace is a 'Latest Test Result (no failures)' section. The 'Build History' table on the left shows a list of builds with columns for build number, status, and time. The 'Changes' panel on the right lists recent changes, including updates to web tests, checkstyle, findbugs support, and release preparation for gameoflife-0.9.13-SNAPSHOT.

CloudBees Jenkins Enterprise

jenkins - gameoflife-freeside

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Exports
Skip Builds
Move/Copy/Promote

Build History

Build	Time
#2	Mar 13, 2015 11:43 AM
#3	Mar 13, 2015 11:43 AM
#4	Mar 13, 2015 11:43 AM
#5	Mar 13, 2015 11:43 AM
#6	Mar 13, 2015 11:43 AM
#7	Mar 13, 2015 11:43 AM
#8	Mar 13, 2015 11:43 AM
#9	Mar 13, 2015 11:43 AM
#10	Mar 13, 2015 11:43 AM
#11	Mar 13, 2015 11:43 AM

Workspace

Recent Changes

Latest Test Result (no failures)

Permalinks

- Last build (11:43 AM)
- Last stable build (11:43 AM)
- Last successful build (11:43 AM)
- Last failed build (11:43 AM)
- Last successful build (11:43 AM)
- Last completed build (11:43 AM)

Changes


#11 (Jun 19, 2015 9:33:16 AM)

1. Change how cells are represented — [adrien.lecharpentier / detail](#)

#10 (Jun 19, 2015 9:24:08 AM)

1. Fine-tuned the web tests — [john.smart / detail](#)
2. Updated checkstyle — [john.smart / detail](#)
3. Added findbugs support — [john.smart / detail](#)
4. [maven-release-plugin] prepare release gameoflife-0.9.13-SNAPSHOT — [john.smart / detail](#)
5. [maven-release-plugin] prepare for next development iteration — [john.smart / detail](#)
6. Findbugs needs XML output — [john.smart / detail](#)
7. Updated versions — [john.smart / detail](#)
8. Update versions — [john.smart / detail](#)
9. Updated thucydides — [john.smart / detail](#)
10. Renamed webtests module to acceptance-tests modules — [john.smart / detail](#)
11. Tidied up code — [john.smart / detail](#)
12. Improved metrics — [john.smart / detail](#)
13. Moved metrics into a separate profile — [john.smart / detail](#)
14. Reintegrated the automated acceptance tests into part of the normal build — [john.smart / detail](#)
15. Pointed the CloudBees instances to the correct URLs — [john.smart / detail](#)
16. Added MySQL for thucydides integration — [john.smart / detail](#)
17. [maven-release-plugin] prepare release gameoflife-0.9.13-SNAPSHOT — [jenkins / detail](#)
18. [maven-release-plugin] prepare for next development iteration — [jenkins / detail](#)
19. Deploying to artifactory — [john.smart / detail](#)
20. [maven-release-plugin] prepare release gameoflife-0.9.14-SNAPSHOT — [jenkins / detail](#)
21. [maven-release-plugin] prepare for next development iteration — [jenkins / detail](#)
22. [maven-release-plugin] prepare release gameoflife-0.9.15-SNAPSHOT — [jenkins / detail](#)
23. [maven-release-plugin] prepare for next development iteration — [jenkins / detail](#)
24. [maven-release-plugin] prepare release gameoflife-0.9.16-SNAPSHOT — [jenkins / detail](#)
25. [maven-release-plugin] prepare for next development iteration — [jenkins / detail](#)

DETAILS DE BUILD - CHANGEMENTS

 CloudBees Jenkins Enterprise

Jenkins » gameoflife-webstyle

[Back to Dashboard](#)

Commit 12b6d807ed7543e667e333869d5ef8f4ee3b5401b by [john.smart](#)
Fine-tuned the web tests

gameoflife-webtests/src/test/java/com/wakaleo/gameoflife/webtests/WhenTheUserEntersAnInitialGrid.java

gameoflife-webtests/src/test/java/com/wakaleo/gameoflife/webtests/WhenTheUserSpawnsTheNextGeneration.java

gameoflife-webtests/src/test/java/com/wakaleo/gameoflife/webtests/WhenTheUserGoesToTheHomePage.java

gameoflife-webtests/src/main

Reports

Stop Builds

Move/Copy/Promote

Build History

View —

#9

May 19, 2016 11:03 AM

#9

May 19, 2016 11:40 AM

#7

May 19, 2016 11:42 AM

#6

May 19, 2016 11:01 AM

#5

May 19, 2016 11:34 AM

#5

May 19, 2016 11:33 AM

#9

May 19, 2016 11:33 AM

#2

May 19, 2016 11:13 AM

#1

May 19, 2016 11:08 AM

Build for all

Build for release

Changes

#11 (Jun 19, 2015 9:33:16 AM)

Builds are represented — [add new test reporter](#) / [detail](#)

2015 9:24:08 AM)

the web tests — [john.smart](#) / [detail](#)

ockstyle — [john.smart](#) / [detail](#)

ugs support — [john.smart](#) / [detail](#)

ase-plugin] prepare release gameoflife-0.9.3-SNAPSHOT — [john.smart](#) / [detail](#)

5. [maven-release-plugin] prepare for next development iteration — [john.smart](#) / [detail](#)

6. Findbugs needs XML output — [john.smart](#) / [detail](#)

7. Updated versions — [john.smart](#) / [detail](#)

8. Update versions — [john.smart](#) / [detail](#)

9. Updated thucydides — [john.smart](#) / [detail](#)

10. Renamed webtests module to acceptance-tests module — [john.smart](#) / [detail](#)

11. Tidied up code — [john.smart](#) / [detail](#)

12. Improved metrics — [john.smart](#) / [detail](#)

13. Moved metrics into a separate profile — [john.smart](#) / [detail](#)

14. Reintegrated the automated acceptance tests into part of the normal build — [john.smart](#) / [detail](#)

15. Pointed the Cloudbees instances to the correct URLs — [john.smart](#) / [detail](#)

16. Added MYSQL for thucydides integration — [john.smart](#) / [detail](#)

17. [maven-release-plugin] prepare release gameoflife-0.9.13-SNAPSHOT — [john.smart](#) / [detail](#)

18. [maven-release-plugin] prepare for next development iteration — [john.smart](#) / [detail](#)

19. Deploying to artifactory — [john.smart](#) / [detail](#)

20. [maven-release-plugin] prepare release gameoflife-0.9.14-SNAPSHOT — [john.smart](#) / [detail](#)

21. [maven-release-plugin] prepare for next development iteration — [john.smart](#) / [detail](#)

22. [maven-release-plugin] prepare release gameoflife-0.9.15-SNAPSHOT — [john.smart](#) / [detail](#)

23. [maven-release-plugin] prepare for next development iteration — [john.smart](#) / [detail](#)

24. [maven-release-plugin] prepare release gameoflife-0.9.16-SNAPSHOT — [john.smart](#) / [detail](#)

25. [maven-release-plugin] prepare for next development iteration — [john.smart](#) / [detail](#)

Latest Test Result (40 Minutes)

Permalinks

Last build failed 23 sec ago

Last build failed 1PM 20 sec ago

Last successful build 1PM 20 sec ago

Last failed build 1PM 1 min 20 sec ago

Last successful build 1PM 1 min 20 sec ago

Last completed build 1PM 23 sec ago

DETAILS DE BUILD – ARTEFACTS

Jenkins » gameoflife-default » #3 » gameoflife-web

 [Back to Project](#)


 [Status](#)

 [Changes](#)

 [Console Output](#)

 [Edit Build Information](#)

 [Delete Build](#)

 [Executed Mojos](#)

 [Deploy Now](#)



 [Test Result](#)

 [See Fingerprints](#)

 [Redeploy Artifacts](#)

 [Previous Build](#)

Build gameoflife-web (Mar 19, 2016 11:53:31 AM)

Build Artifacts		
	gameoflife-web-0.0.82.pom	6.44 KB view
	gameoflife-web-0.0.55.war	3.27 MB view



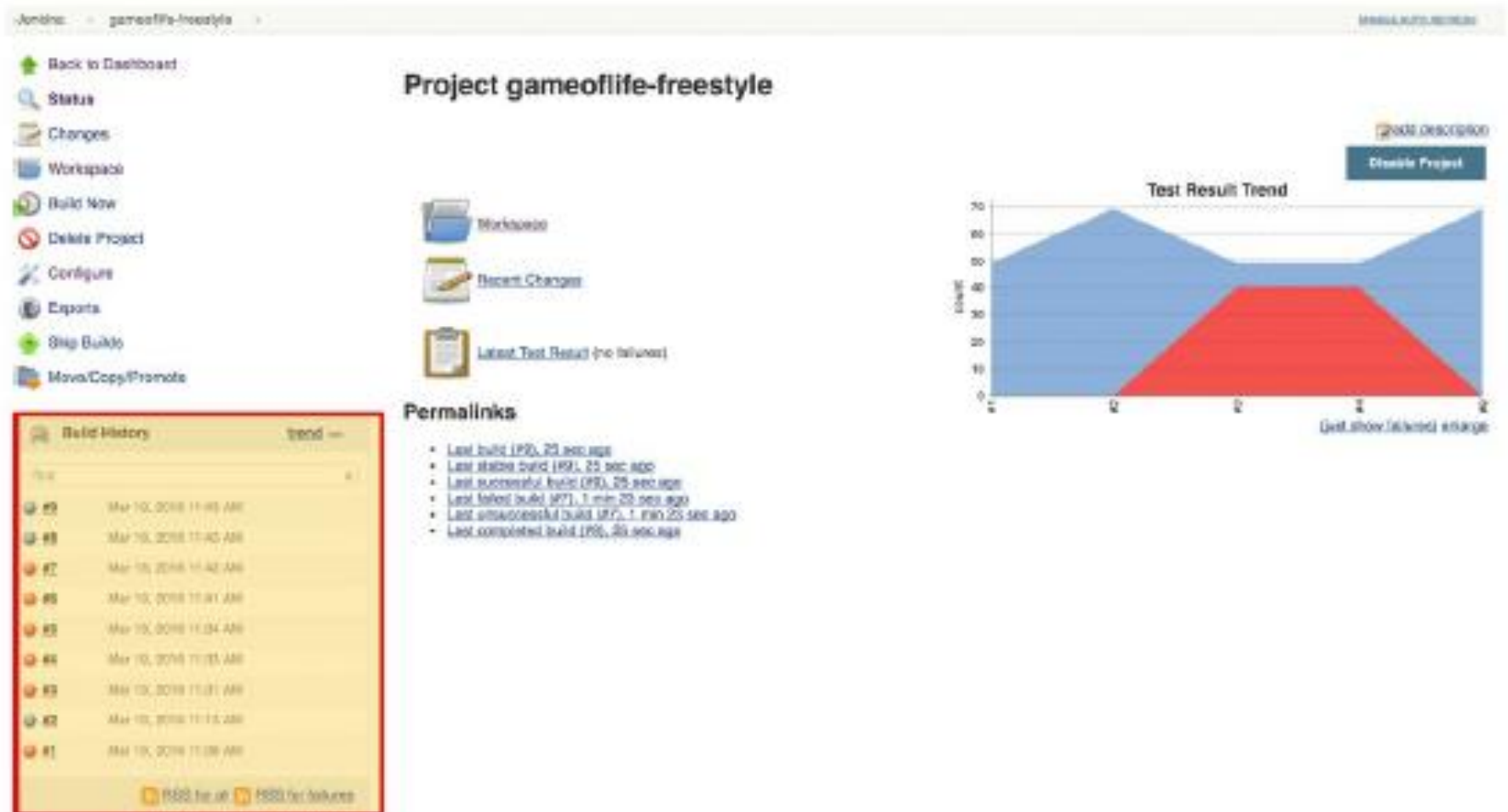
~~No changes~~ ~~Changes in dependency~~

1. [gameoflife-core](#)  #2 →  #3 [\(detail\)](#)
2. [gameoflife-webservice](#)  #2 →  #3 [\(detail\)](#)



[Test Result](#) (no failures)

DETAILS DE BUILD – HISTORIQUE



DETAILS DE BUILD

The screenshot shows the Jenkins interface for a build. The top bar indicates 'CloudBees Jenkins Enterprise' and the project path 'Jenkins > gameoflife-default > #3'. The main heading is 'Build #3 (Mar 19, 2016 11:53:26 AM)'. On the left, a sidebar contains links: 'Back to Project', 'Status', 'Can build information', 'Delete Build', 'Test Result', 'See Fingerprints', and 'Previous Build'. The main content area includes: 'Revision: 48 Changes' with a link to '1. Rewriting Dead Cell code (detail/Greentox2.x)'; 'Started by anonymous user'; 'This run spent:' with a list of timings; 'Test Result (no failures)' with a link to 'Test Result'; 'Module Builds' with a table of module build times; and 'Downstream Builds' with links to 'gameoflife-integration-tests' and 'gameoflife-metrics'. Yellow callout boxes with arrows point to specific elements: 'Build number and date' points to the build title; 'Who / what triggered this build' points to the 'Started by' information; 'The changes of this build' points to the 'Changes' section; 'Build timings' points to the 'This run spent:' section; 'Test results' points to the 'Test Result' link; and 'Module dependencies (Maven)' points to the 'Module Builds' table.

Build number and date

Who / what triggered this build

The changes of this build

Build timings

Test results

Module dependencies (Maven)

Build Details:

- Build #3 (Mar 19, 2016 11:53:26 AM)
- Revision: 48 Changes
- 1. Rewriting Dead Cell code (detail/Greentox2.x)
- Started by anonymous user
- This run spent:
 - 30 ms waiting in the queue;
 - 31 sec building on an executor;
 - 31 sec total from scheduled to completion.
- Test Result (no failures)
- Module Builds
- Downstream Builds

Module	Build Time
gameoflife	0,81 sec
gameoflife-build	1,4 sec
gameoflife-ci	0,34 sec
gameoflife-core	3,1 sec
gameoflife-web	15 sec
gameoflife-webserver	0,36 sec

gameoflife-integration-tests #3
gameoflife-metrics (none)

DETAILS DE BUILD – SORTIE DE LA CONSOLE



The screenshot shows the Jenkins CloudBees Enterprise interface. The top navigation bar includes the Jenkins logo, the text "CloudBees Jenkins Enterprise", and a search bar. Below the navigation bar, the breadcrumb "Jenkins > gameoflife-freestyle > #2" is visible. On the left sidebar, there are several links: "Back to Project", "Status", "Changes", "Console Output" (highlighted with a red box), "View as plain text", "Edit Build Information", "Delete Build", "Tag this build", "Test Result", "Previous Build", and "Next Build". The main content area is titled "Console Output" and displays the build log. The log starts with "Edonné par l'utilisateur anonymous" and "building in workspace C:\CloudBees\tools\jenkins-data\workspace\gameoflife-freestyle". It then shows the update of the workspace to revision 16. The log continues with a message about no changes for the workspace since the previous build. The build process is then initiated with the command "cmd.exe /C \"C:\CloudBees\tools\apache-maven-3.0.2\bin\mvn.bat -Djetty.port=9998 -Dm2repository.port=9991 -Dm2repository.url=https://maven.apache.org/m2-wget/\"". The log shows the Maven process starting and then encountering several warnings. The first warning is about the missing 'build.plugins.plugin.version' for 'org.apache.maven.plugins:maven-project-info-reports-plugin'. The second warning is about the missing 'build.plugins.plugin.version' for 'org.apache.maven.plugins:maven-war-plugin'. The log then shows the Maven process completing and the build finishing successfully. The final output is a list of build artifacts: "gameoflife", "gameoflife-build", "gameoflife-code", "gameoflife-web-service", "gameoflife-web", and "gameoflife-ctl".

CloudBees Jenkins Enterprise

Jenkins > gameoflife-freestyle > #2

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete Build

Tag this build

Test Result

Previous Build

Next Build

Console Output

```
Edonné par l'utilisateur anonymous
building in workspace C:\CloudBees\tools\jenkins-data\workspace\gameoflife-freestyle
updating workspace C:\CloudBees\tools\jenkins-data\workspace\gameoflife-freestyle at revision '2016-03-19T11:13:06.661 +0100'
At revision 16

No changes for workspace C:\CloudBees\tools\jenkins-data\workspace\gameoflife-freestyle since the previous build
[gameoflife-freestyle] $ cmd.exe /C "C:\CloudBees\tools\apache-maven-3.0.2\bin\mvn.bat -Djetty.port=9998 -
Dm2repository.port=9991 -Dm2repository.url=https://maven.apache.org/m2-wget/"
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.wakaleo.gameoflife:gameoflife-
code:jar:0.0.00
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-project-info-reports-plugin is missing. @ line 17, column 11
[WARNING] 'reporting.plugins.plugin.version' for org.apache.maven.plugins:maven-project-info-reports-plugin is missing.
@ line 25, column 11
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.wakaleo.gameoflife:gameoflife-
web-service:war:0.0.00
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-war-plugin is missing. @ line 103, column
11
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this session, Maven will continue to ignore these problems, and will not attempt to fix them.
[INFO]
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] gameoflife
[INFO] gameoflife-build
[INFO] gameoflife-code
[INFO] gameoflife-web-service
[INFO] gameoflife-web
[INFO] gameoflife-ctl
[INFO]
```

ARTEFACTS DANS JENKINS

- Un artefact est un fichier produit à la suite d'une construction Jenkins.
- Ce nom vient de la convention de nommage de Maven
- Une seule construction Jenkins peut produire de nombreux artefacts
- Par défaut, ils sont stockés là où ils sont créés :
 - Dans l'espace de travail du chantier
 - Ils sont donc supprimés lorsque l'espace de travail est effacé à moins qu'ils soient archivés

COMMENT ARCHIVER LES ARTEFACTS ?

- Les artefacts peuvent être archivés, avec l'action Post-build d'un job :

Nécessite un motif pour savoir quels artefacts archiver:

- my-app.zip: Le fichier my-app.zip, à la racine de l'espace de travail image
- /*. png: Tous les fichiers avec l'extension . png dans le dossier images à la racine de l'espace de travail
- target/**/* . jar: Tous les fichiers avec l'extension . jar, récursivement dans le dossier cible, à la racine de l'espace de travail
- L'archivage conserve ces fichiers dans \${JENKINS_HOME} pour toujours
 - Exhauste l'espace disque du système de fichiers
 - Sauvegardes?

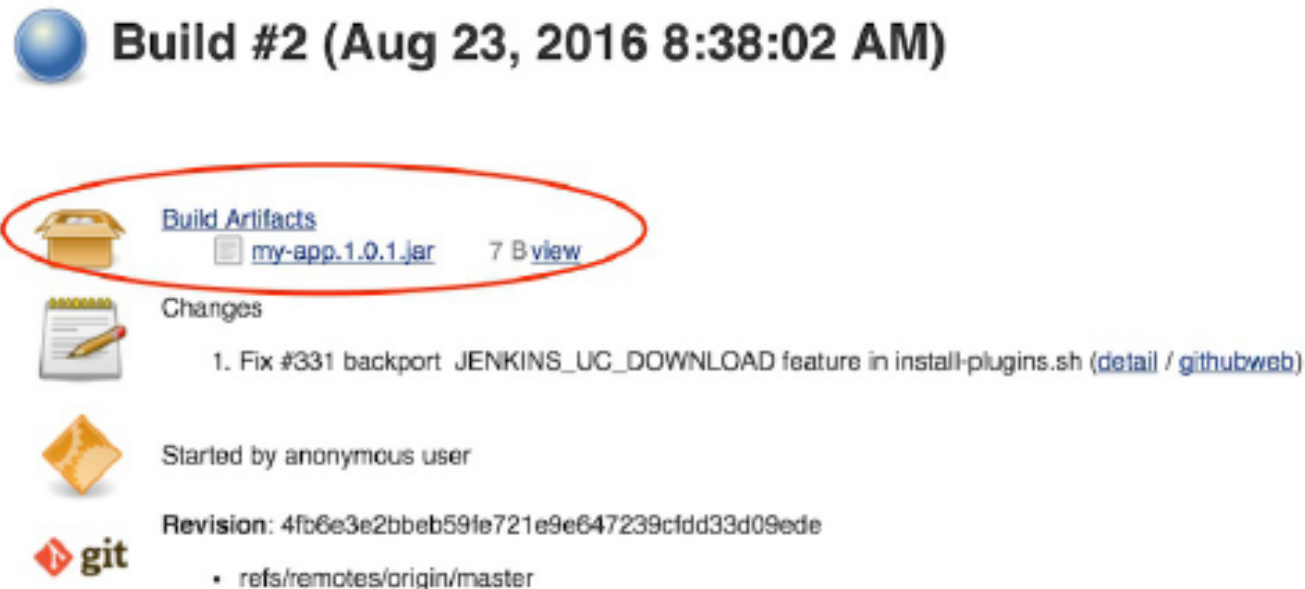
ACCÈS AUX ARTEFACTS ARCHIVÉS

Une fois archivé, attaché à la construction qui l'a produit:


- Tous les artefacts d'une construction :


`http://${JENKINS_URL}/job/${YOUR_JOB}/${BUILD_NUMBER}/artifact`

- Visible sur la page principale de build :



Build #2 (Aug 23, 2016 8:38:02 AM)

 [Build Artifacts](#)
 [my-app.1.0.1.jar](#) 7 B [view](#)

 **Changes**
1. Fix #331 backport JENKINS_UC_DOWNLOAD feature in install-plugins.sh ([detail](#) / [githubweb](#))

 Started by anonymous user

 **Revision:** 4fb6e3e2bbeb59fe721e9e647239cidd33d09ede
• refs/remotes/origin/master

POLITIQUE DE CONSERVATION DES ARTEFACTS

- Couplé à la politique de rétention de build
- La suppression d'un build supprime les artefacts attachés
- Bonne pratique : jeter les anciens builds et nettoyer:
 - Selon l'âge : Nombre de jours pour conserver un build
 - Selon le nombre : Nombre maximum de builds à conserver
- Important: un build peut être gardé pour toujours
 - marqué comme **Kept Forever**
 - Utile pour les versions de release ou "Promotions"



The screenshot shows the 'Discard Old Builds' configuration page in Jenkins. It includes a checkbox for 'Discard Old Builds' which is checked. Below it, the 'Strategy' is set to 'Log Rotation'. There are two input fields: 'Days to keep builds' with a value of 7, and 'Max # of builds to keep' with a value of 20. Both fields have small text below them explaining their function. An 'Advanced...' button is located at the bottom right.

☒ Discard Old Builds

Strategy: Log Rotation

Days to keep builds: 7
If not empty, build records are only kept up to this number of days

Max # of builds to keep: 20
If not empty, only up to this number of build records are kept

Advanced...

GESTION DU CODE SOURCE AVEC JENKINS

- Utiliser SCM comme source de build pour un projet Jenkins donné :
- Le déclenchement s'appuie sur les événements du CMP
- Intégration avec le navigateur SCM pour voir les changements
- Intégration avec l'autorisation SCM pour offrir une meilleure expérience utilisateur



GESTION DU CODE SOURCE AVEC JENKINS

QU'EST-CE QUE LA SCM AVEC LES PROJETS JENKINS ?

- Intégration basée sur des plugins
- Plusieurs sont supportés : Git, Mercurial, Subversion, CVS, TFS...
- Les fonctionnalités de Sidekicks sont également des plugins : AAA (cf. Sécurité) : Github Oauth, Bitbucket Auth...

- Intégrations du navigateur :
 - Voir les détails des modifications apportées aux builds
 - Partager des pointeurs avec d'autres utilisateurs
 - Jenkins intègre avec les navigateurs de nombreux SCM modernes
 - Subversion : Trac, Viewsvn, Websvn, Sventon, Fisheye, Collabnet...
 - Git : gitweb, github,...
 - Mercurial : bitbucket, googlecode, hgweb,...

COMMENT INTÉGRER LE SCM AVEC JENKINS ?

1. Commun pour tous les projets :

- Section Gestion du code source dans la configuration des jobs
- Sélectionner le type de SCM
- Configurer l'URL, les identifiants, les options spécifiques SCM

2. Ajouter éventuellement un navigateur SCM et un déclencheur de build basé sur SCM

3. Puis lancer un build et voir SCM en action

INTEGRER UN SELF-HOSTED GIT

The screenshot shows a GitHub repository page for 'jenkins / demoapp'. The repository has 1 watch, 0 stars, and 0 forks. The 'Code' tab is selected, showing the file list. The repository is owned by 'Damien DUPORTAL' and has a single branch named 'demoapp'. The file list includes a 'docker' directory, a 'src' directory, a 'sshkeys' directory, and several files like '.dockerignore', '.gitignore', 'Dockerfile', 'Jenkinsfile', 'README.md', 'docker-compose.yml', 'hello-world.yml', and 'pom.xml'. Each file entry shows a commit hash and a description of the commit.

Home Explore Help Register Sign in

jenkins / demoapp Watch 1 Star 0 Fork 0

Code Issues 0 Pull Requests 0 Commits 18 Releases 0 Wiki

No Description

Branch: master demoapp HTTP SSH http://gitserver:3000/jenkins

File	Commit Hash	Commit Message	Time
Damien DUPORTAL	5be65c8bc5	README	2 months ago
docker	c1754738aa	Making it works offline	2 months ago
src	23837a398e	Making a GUI for app	2 months ago
sshkeys	8987225697	Fully Docker infra with 2 Jenkins slaves	2 months ago
.dockerignore	c1754738aa	Making it works offline	2 months ago
.gitignore	23837a398e	Making a GUI for app	2 months ago
Dockerfile	56c97cb468	Building the app in a Docker container	2 months ago
Jenkinsfile	5c91366d18	Jenkinsfile: adding Docker build and push parts	2 months ago
README.md	5be65c8bc5	README	2 months ago
docker-compose.yml	c1754738aa	Making it works offline	2 months ago
hello-world.yml	23837a398e	Making a GUI for app	2 months ago
pom.xml	23837a398e	Making a GUI for app	2 months ago

CONFIGURATION DU SCM DU PROJET JENKINS

- Git est sélectionné comme SCM
- URL et identifiants configurés :

Source Code Management

☐ None
☐ CVS
☐ CVS Projectset
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

CONFIGURATION DU SCM DU PROJET JENKINS

- Git est sélectionné comme SCM
- URL et identifiants configurés :

Source Code Management

☐ None
☐ CVS
☐ CVS Projectset
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

- Configuration du navigateur SCM pointant vers Gogs :

Repository browser

URL

JENKINS BUILD CHANGELOG

- Chaque build suit les changements depuis le précédent :

Build #6 (Aug 23, 2016 10:05:25 AM)



Build Artifacts



[my-app.1.0.1.jar](#)

7 B [view](#)



Changes

1. Updating README to reflect latest changes ([detail](#) / [gogs](#))
2. Updating main web page of application ([detail](#) / [gogs](#))



Started by anonymous user



Revision: 4e37423f2be396d0c3e22145243f1e6788dcb89e

- origin/gogs-jenkins-browse-bug

JENKINS SCM BROWSER

À partir des pages Modifications ou Modifications détaillées, vous pouvez accéder à :

- Fichiers actuels dans le navigateur Web SCM.
- Diff, également dans le navigateur Web SCM.

The image shows a Jenkins SCM Browser interface. On the left, a 'Changes' summary lists two commits. The first commit, 3bd0aecd451d7895e50945db0cd3c57a37d98a5, is by damien.duportal and updates the README. The second commit, 4e37423f2be395d3c3e22145243f1e6788dcb09e, is also by damien.duportal and updates the main web page. A red arrow points from the 'README.md (diff)' link in the first commit to a detailed diff view on the right. A purple arrow points from the 'src/main/resources/assets/index.html (diff)' link in the second commit to the same detailed diff view. The detailed diff view on the right shows the changes to the index.html file, with a green background indicating the added content.

Changes

Summary

1. Updating README to reflect latest changes ([details](#))
2. Updating main web page of application ([details](#))

Commit 3bd0aecd451d7895e50945db0cd3c57a37d98a5 by damien.duportal
Updating README to reflect latest changes
Signed-off-by: Damien DUPORTAL, <damien.duportal@gmail.com>

[README.md \(diff\)](#)

Commit 4e37423f2be395d3c3e22145243f1e6788dcb09e by damien.duportal
Updating main web page of application
Signed-off-by: Damien DUPORTAL, <damien.duportal@gmail.com>

[src/main/resources/assets/index.html \(diff\)](#)

[Direct Access to Gogs](#)

MISES À JOUR PROGRESSIVES VS CLEAN CHECKOUT

- Comportement de build par défaut de Jenkins :
 - Essayer de réduire le temps de build au fil du temps
 - Essayez de construire là où des constructions réussies ont déjà eu lieu
 - Code source déjà cloné dans l'espace de travail du projet : mise à jour mineure, quelques éléments à télécharger
 - Bon pour les constructions exécuter fréquemment
- Comportement par défaut non acceptable pour une version : besoin de construire à partir de zéro
 - Nettoyer l'espace de travail avant la compilation est une option
 - Ainsi SCM "checkout" commence à partir d'un espace de travail vide
 - La politique de contrôle SCM peut également être ajustée pour cela (voir la diapositive suivante)

CAPACITÉS SCM AVANCÉES

- Le bouton « avancé » pour la configuration SCM fournit de nouveaux comportements, basé sur le type SCM:
- Branches, tags, références pour les SCM décentralisés
- Sous-modules pour Git
- Comportement de clone avancé (timeout, clean avant clone)
- Génération Changelog

AUTRES DÉCLENCHEURS

L'application des concepts de CI/CD vous donne envie de lancer un build dès que possible.

- En fonction de vos besoins, quand voulez-vous lancer des constructions ?

- Base régulière : "Toutes les nuits", "Tous les lundis", "Toutes les heures«
- Changement de code : pour chaque commit
- Mélange des deux : "Vérification des changements toutes les 5 minutes"

QUE SONT LES DÉCLENCHEURS DE BUILD ?

- Build Trigger est le mécanisme qui lance la construction de projets/jobs
- Il s'agit d'un ensemble de configuration au niveau du projet
- Le déclencheur par défaut est vous : Lancer manuellement une compilation
- Certains déclencheurs sont propres à certains types de projets
- Les projets Maven peuvent être construits lorsqu'une dépendance a été construite au sein de Jenkins
- La liste de déclenchement peut être étendue par certains plugins
- Le « Promotion plugin » ajoute « Trigger quand un autre emploi promu »

DÉCLENCHEURS COMMUNS : CONSTRUIRE PÉRIODIQUEMENT

- Tous les projets peuvent le faire
- Utiliser une syntaxe similaire à celles du Cron
- Assure que des projets sont construits à des dates ou à des fréquences prévues
- "Construire toutes les 30 minutes" :

☒ Build periodically

Schedule

H/30 * * * *

Would last have run at Thursday, August 25, 2016 3:48:12 PM UTC; would next run at Thursday, August 25, 2016 4:18:12 PM UTC.

DÉCLENCHEURS COURANTS : SCRUTATION DU SCM

- Vérifier périodiquement les changements dans le SCM:
 - Au lieu de lancer build périodiquement
 - Même syntaxe que les triggers « Build périodiquement »
 - Pas de changements, pas de build déclenché:
 - "Vérifier les changements de SCM toutes les 5 minutes" :

☒ Poll SCM

Schedule

H/5 * * * *

Would last have run at Thursday, August 25, 2016 4:05:31 PM UTC; would next run at Thursday, August 25, 2016 4:05:31 PM UTC.

Ignore post-commit hooks ☒

DÉCLENCHEURS COURANTS : APRÈS UN AUTRE PROJET

- Vous pouvez construire après qu'un autre projet a été construit
- Selon le statut du projet parent : stable, instable, cassé ?

☒ Build after other projects are built

Projects to watch

maven

- ☒ Trigger only if build is stable
- ☐ Trigger even if the build is unstable
- ☐ Trigger even if the build fails

ARRETER LA SCRUTATION

- Le scrutation est inefficace car il ajoute un délai entre la validation et le démarrage du build
- Préoccupations de l'administrateur : Les scrutation d'SCM consomment les réseaux et le système de fichiers.
- Besoin réel : Construire dès que je change de code
- Solution : Trigger build en dehors de Jenkins
- Les systèmes externes (comme les SCM) détecte les changements
- Exemple : Les systèmes SCM ont des Hooks :
Configuration d'une URL pour appeler lorsque des événements se produisent

DÉCLENCHEURS EXTERNES FONDÉS SUR DES ÉVÉNEMENTS

- Jenkins fournit des entrées pour lancer des compilations :
- Les plugins SCM (comme git-plugin) fournissent des endpoints d'API pour déclencher les compilations :

`http://JENKINS_URL>/git/notifyCommit? url=GIT_REPO_URL`

- Trigger build à distance : par l'URL du projet, avec un TOKEN à fournir.
Ensuite, pour lancer build, il suffit d'appeler:

`http://JENKINS_URL>/job/your-job>/build? TOKEN=YOURTOKEN`

☒ Trigger builds remotely (e.g., from scripts)

Authentication Token

SuchVerySecretTokenI'mSureNoOneWillNeverFindThisIn2016!

Use the following URL to trigger build remotely: `JENKINS_URL/job/free/build?token=TOKEN_NAME` or `/buildWithParameters?`

`token=TOKEN_NAME`

Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

AVANTAGES DES TESTS AVEC JENKINS

- Jenkins construit vos applications aussi souvent que possible
- Automatiser les tests au même endroit :
- Limite les coûts
- Détecte les erreurs tôt : par des tests continus de changement
- Jenkins est un serveur d'automatisation : le bon outil pour ce genre de tâche !
- Jenkins soutient la publication de rapports de tests dans le cadre de projets
- L'extensibilité avec les plugins permet l'intégration de n'importe quel framework de test

AVANTAGES DES TESTS AVEC JENKINS

- Jenkins soutient les frameworks de tests avec... plugins
 - Généralement un rapport agrégateur pour le type d'analyse de test (statique, dynamique...)
 - Un plugin pour chaque outils d'automatisation de tests (Checkstyle, etc.)
 - Il suffit d'installer des plugins, et Build Steps ou Post-build Actions apparaîtra
 - Les tests sont exécutés dans le cadre de la compilation :
- Exemple : mvn install implique mvn test
- Build Summary intègre également des informations sur les outils .



CASSER LES BUILDS


Le statut global des tests implique le statut de construction :

Compilation "Blue" : compilation, essai et paquet

Compilation jaune : compiler mais lancer des avertissements


Compilation rouge : ... les tests échouent ou pire !


Exemple avec Junit échoue à la construction sauf si on lui a dit :

☐ Publish JUnit test result report 


Test report XMLs


Please [includes](#) setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

☐ Retain long standard output/error 

Health report amplification factor 

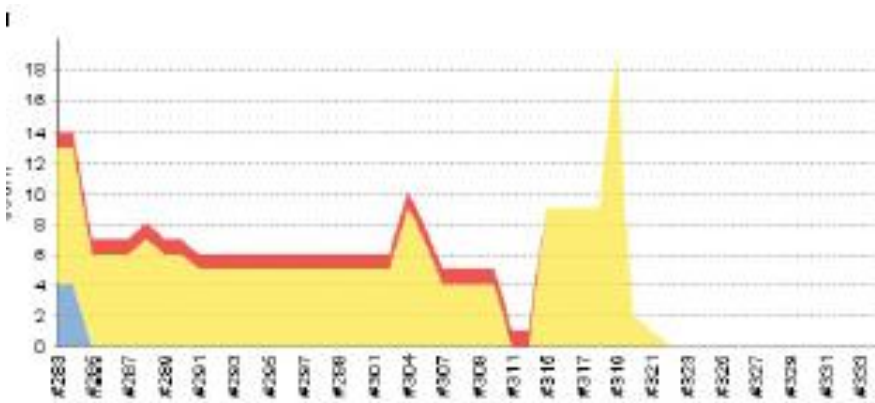
1% failing tests scores as 99% health. 5% failing tests scores as 95% health

☒ Allow empty results ☐ Do not fail the build on empty test results 



AFFICHAGE DES RÉSULTATS DES TESTS DANS JENKINS

- Jenkins supporte la publication des résultats des tests.
 - Les rapports de test sont considérés comme des artefacts liés à la construction
 - Archivé automatiquement : contenu léger
- La publication se fait généralement sous la forme d'une action postérieure à la construction, selon l'état de la construction
- Des plugins spécifiques permettent à Jenkins de servir ce contenu statique comme un rapport visuel :
 - Les plugins de reporting fournissent des tendances sur les tests
 - Beaucoup de vues : avertissements, cumulatifs, évolutions de test...



ANALYSE STATIQUE

- Les tests et leurs rapports sur une application
- Jamais nécessaire pour lancer l'application (pas de smoke tests)
- Peut fonctionner facilement dans Jenkins
- Test unitaire
- Code Lint
- Outillage installé ou configuré par Jenkins ou le Projet

ANALYSE DYNAMIQUE

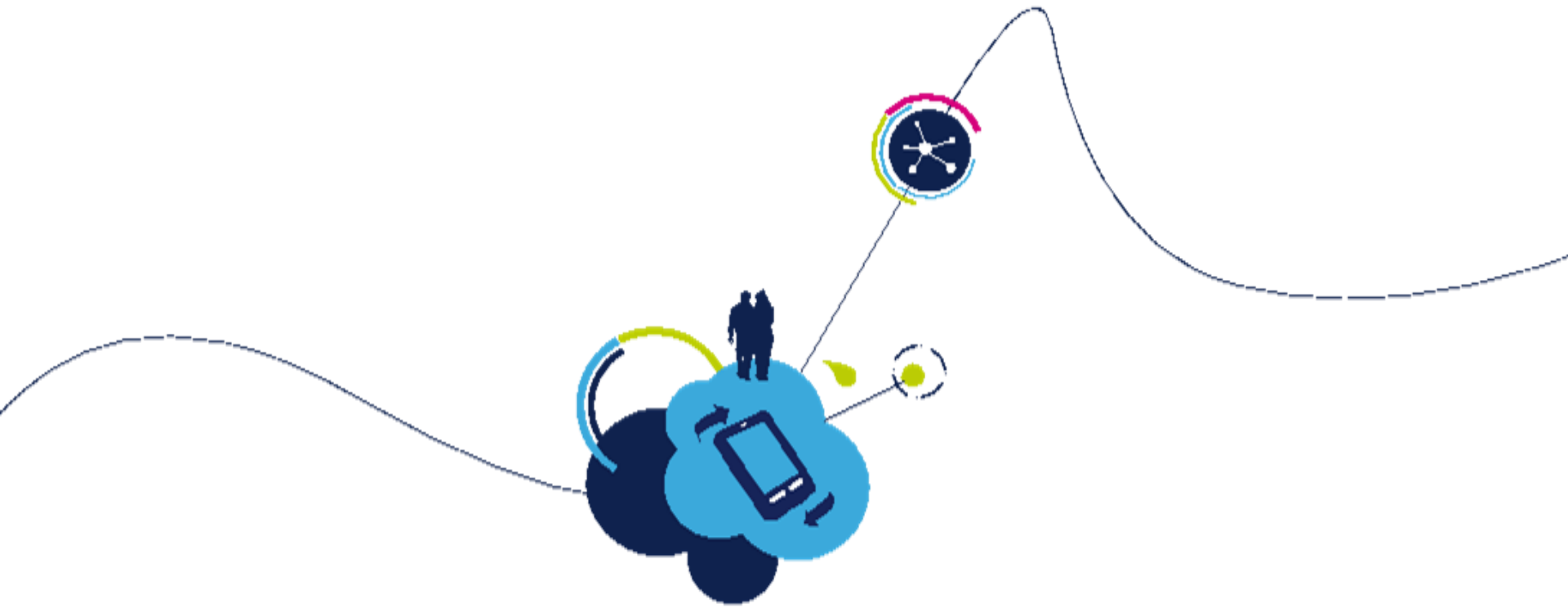
- C'est aussi les tests et leurs rapports sur une application
- Exige le lancement de l'application , même dans un environnement non productif
- Le smoke test de base est implicite
- Exige la gestion des ressources au niveau Jenkins :
 - Port pour écouter ?
 - Ressource du système de fichiers ?
 - Accès au réseau ?
- Cycle de vie de l'application : s'assurer qu'il sera arrêté ou nettoyé dans tous les cas

INTÉGRATION AVEC LES OUTILS D'AUTOMATISATION DE TESTS

- Les tests et les analyses peuvent être effectués à l'extérieur de Jenkins:

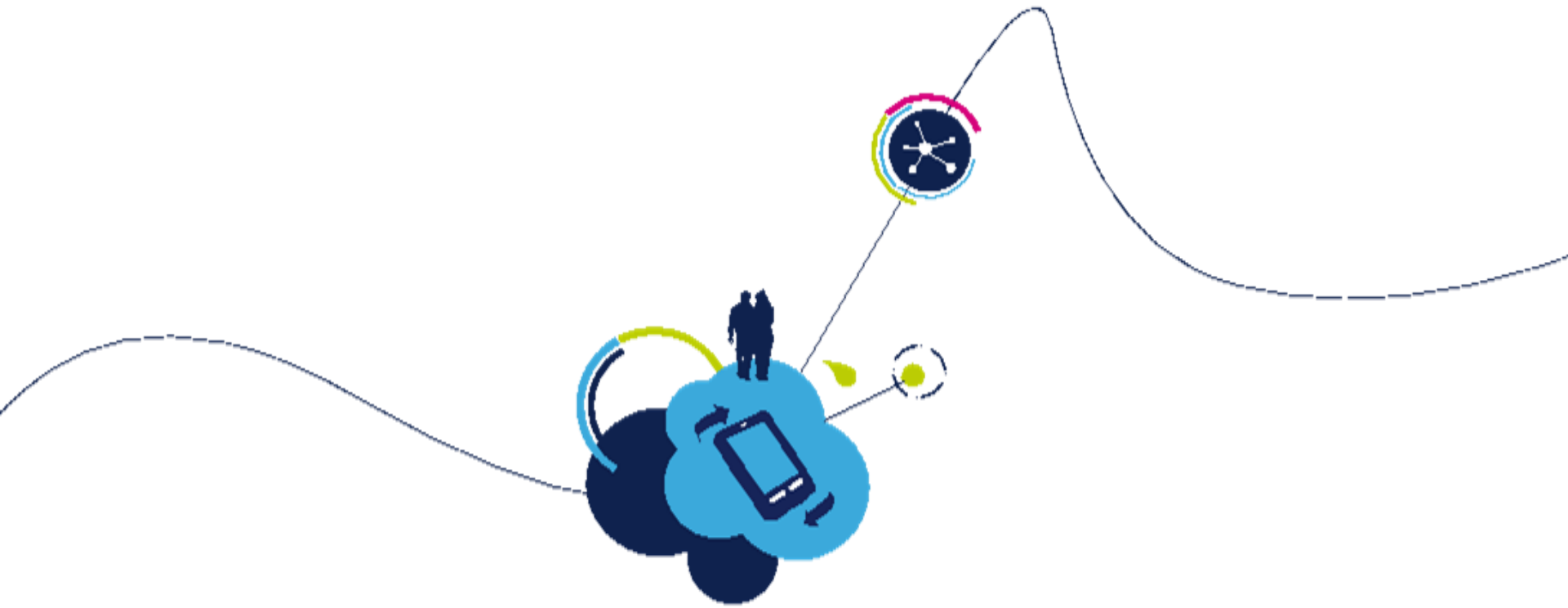
Sonarqube est un excellent exemple

- L'intégration se fait avec plus de plugins
 - Un plugin par outil d'automatisation de test tiers fonctionnant à l'extérieur de Jenkins (Sonar...)
 - Automatiquement connecté à l'agrégateur pour le reporting ou le résumé
- Exige la gestion centrale de Jenkins :
 - Localisation du service externe
 - Paramètres d'authentification
- Paramétrage dans la configuration du projet comme pour les tests : étapes et actions
- Les rapports sont généralement disponibles via les permaliens vers le service distant



Lab n°1:

Anatomie d'une application, Vue d'ensemble.



UTILISATION AVANCÉE DE JENKINS

Sommaire

- Les dossiers et les vues
- La rétroaction et les notifications
- Les builds distribués

POURQUOI ORGANISER VOS PROJETS ?

- Comme votre nombre de projets commence à augmenter, vous aurez besoin d'une certaine organisation
- De nombreux jobs par application pour différentes étapes
- De nombreuses applications et équipes
- Jenkins fournit 2 outils pour vous aider dans cette tâche :
 - Sortir de la boîte : Vues
 - Avec un plugin : Dossiers

UTILISATION DES VUES

Views appear as tabs

Create a new view

Test List		Tous	+
S	W	Name ↓	Last Success
		gameoflife-default	4 min 40 sec - #2
		gameoflife-freestyle	21 min - #2
		gameoflife-integration-tests	2 min 41 sec - #3
		gameoflife-metrics	3 min 48 sec - #3
		gameoflife-web-tests	26 min - #1

Icon: [S](#) [M](#) [L](#)

VUE DE LIST

- Contains a list of projects:

View name

- ☐ **Dashboard**
Customizable view that contains various portlets containing information about your job(s)
- ☐ **Groovy Script View**
This view renders HTML produced by a Groovy script.
- ☐ **List View**
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

OK

CREATION DE VUE

Name

Description

[Plain text] [Preview](#)

Filter build queue ☐

Filter build executors ☐

Job Filters

Status Filter

Recurse in subfolders ☐

Jobs

- ☐ gameoflife-default
- ☐ gameoflife-freestyle
- ☐ gameoflife-integration-tests
- ☐ gameoflife-metrics
- ☐ gameoflife-web-tests

☐ Use a regular expression to include jobs into the view

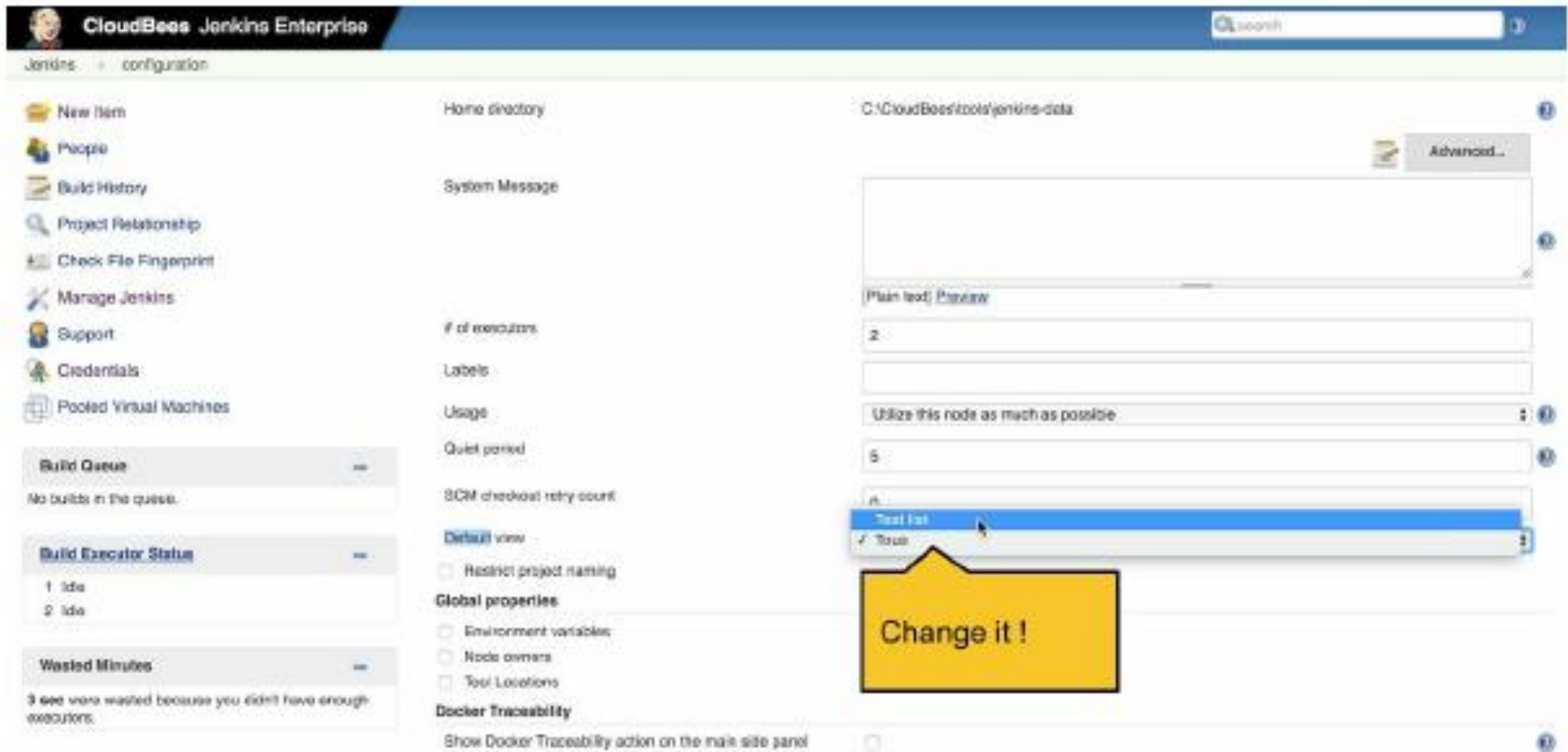
Add Job Filter ▼

Columns

⋮ Status	Delete
⋮ Weather	Delete
⋮ Name	Delete

VUE PAR DEFAUT

La vue 'ALL' est celle par défaut, mais si d'autres vues existent, elles peuvent être modifiées dans \$JENKINS_URL/configure :

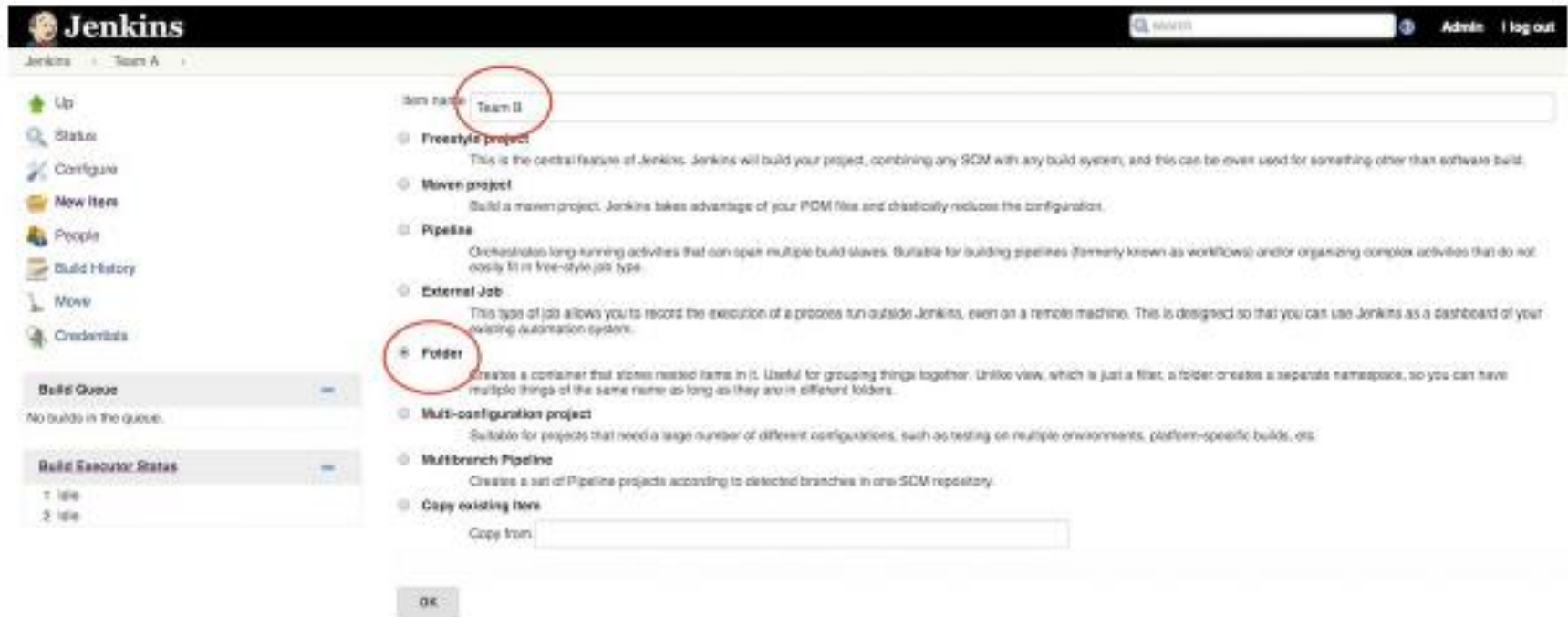


The screenshot displays the Jenkins Enterprise configuration interface. The left sidebar contains navigation links such as 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'Support', 'Credentials', and 'Pooled Virtual Machines'. Below these are sections for 'Build Queue' (showing 'No builds in the queue'), 'Build Executor Status' (showing 1 idle and 2 busy executors), and 'Wasted Minutes' (showing 3 see were wasted). The main configuration area is titled 'configuration' and includes fields for 'Home directory' (C:\CloudBees\tools\jenkins-data), 'System Message', '# of executors' (2), 'Labels', 'Usage' (Utilize this node as much as possible), 'Quiet period' (5), and 'SCM checkout retry count' (1). A dropdown menu for 'Default view' is open, showing 'Text list' and 'True'. A yellow callout box with the text 'Change it !' points to this dropdown. The bottom of the page shows 'Global properties' (Environment variables, Node names, Tool Locations) and 'Docker Traceability' (Show Docker Traceability action on the main side panel).

POURQUOI UTILISER DES DOSSIERS ?

- Vous aider à modéliser des taxonomies personnalisées
- Crée un espace de noms:

Le job « build » dans le dossier A est différent du job « build » dans le dossier B



CONFIGURER UN DOSSIER

The screenshot displays the Jenkins web interface. At the top, the Jenkins logo and name are on the left, and a search bar, 'Admin' link, and 'log out' link are on the right. Below the top bar, a breadcrumb navigation shows 'Jenkins' > 'Team A'. On the left sidebar, there is a list of navigation links: 'Up', 'Status', 'Configure', 'New Item', 'Delete Folder', 'People', 'Build History', 'Move', and 'Credentials'. Below these links are two summary boxes: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 idle' and '2 idle'). The main content area is titled 'Team A' and contains a configuration form. The form includes fields for 'Name' (set to 'Team A'), 'Display Name', 'Description' (with a '[Plain text] Choose' link), 'Views Tab Bar' (set to 'Default Views TabBar'), and 'Icon' (set to 'Default icon'). Below these fields is a 'Health metrics' section with a 'Health metrics...' button. At the bottom of the form are 'Save' and 'Apply' buttons. A red arrow points from the 'Team A' breadcrumb in the top navigation bar to the configuration form.

Jenkins

search Admin log out

Jenkins > Team A >

Up
Status
Configure
New Item
Delete Folder
People
Build History
Move
Credentials

Build Queue —
No builds in the queue.

Build Executor Status —
1 idle
2 idle

Name: Team A

Display Name

Description: [Plain text] Choose

Views Tab Bar: Default Views TabBar

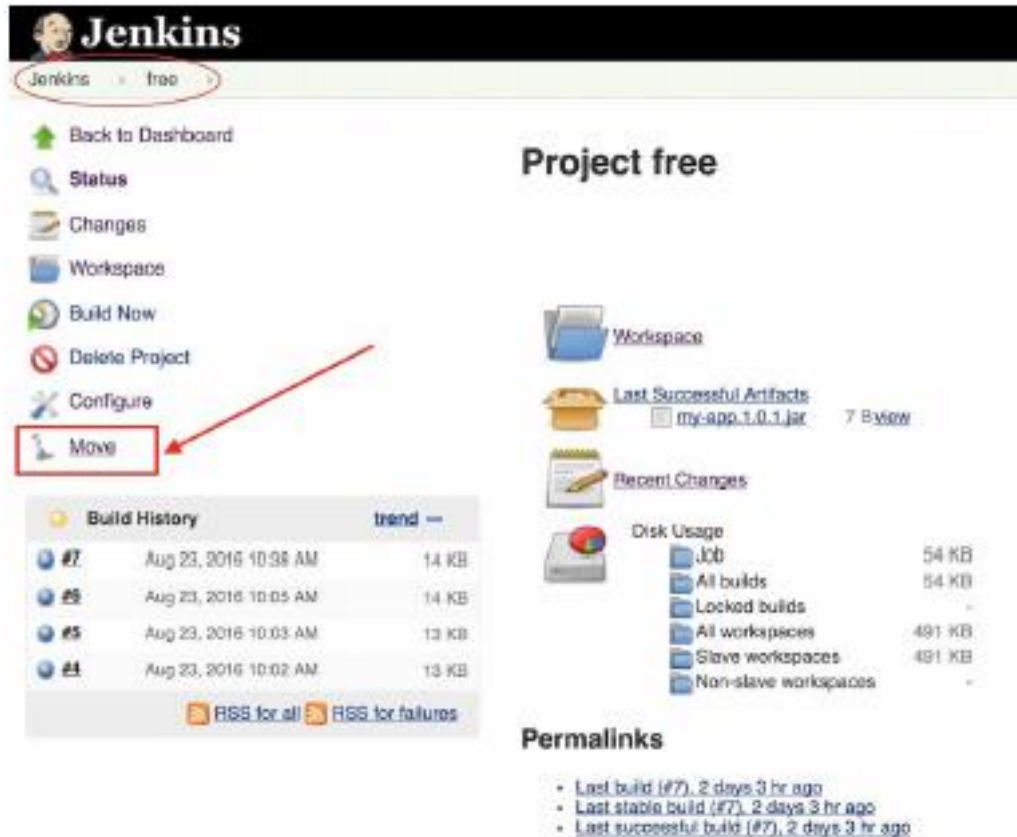
Icon: Default icon

Health metrics

Health metrics...

Save Apply

DEPLACER UN JOB DANS UN DOSSIER



The screenshot shows the Jenkins interface for a project named 'free'. The 'Jenkins' logo and the project name 'free' are circled in red. A red arrow points to the 'Move' button in the left sidebar. The main content area displays project details for 'Project free', including a workspace, last successful artifacts (my-app-1.0.1.jar), recent changes, and disk usage. A build history table is also present.

Build History

#	Time	Size
#7	Aug 23, 2016 10:38 AM	14 KB
#6	Aug 23, 2016 10:05 AM	14 KB
#5	Aug 23, 2016 10:03 AM	13 KB
#4	Aug 23, 2016 10:02 AM	13 KB

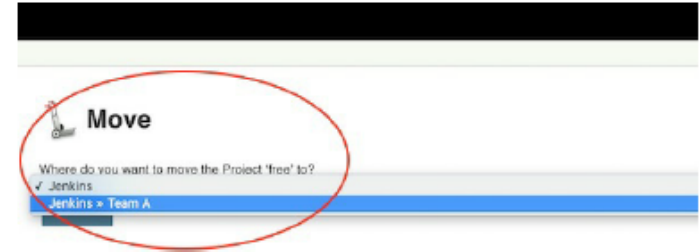
[RSS for all](#) [RSS for failures](#)

Disk Usage

Category	Size
Job	54 KB
All builds	54 KB
Locked builds	-
All workspaces	491 KB
Slave workspaces	491 KB
Non-slave workspaces	-

Permalinks

- Last build (#7), 2 days 3 hr ago
- Last stable build (#7), 2 days 3 hr ago
- Last successful build (#7), 2 days 3 hr ago



The screenshot shows the 'Move' dialog box in Jenkins. The dialog asks 'Where do you want to move the Project 'free' to?'. The 'Jenkins' option is selected, and 'Jenkins > Team A' is highlighted in blue. The dialog is circled in red.

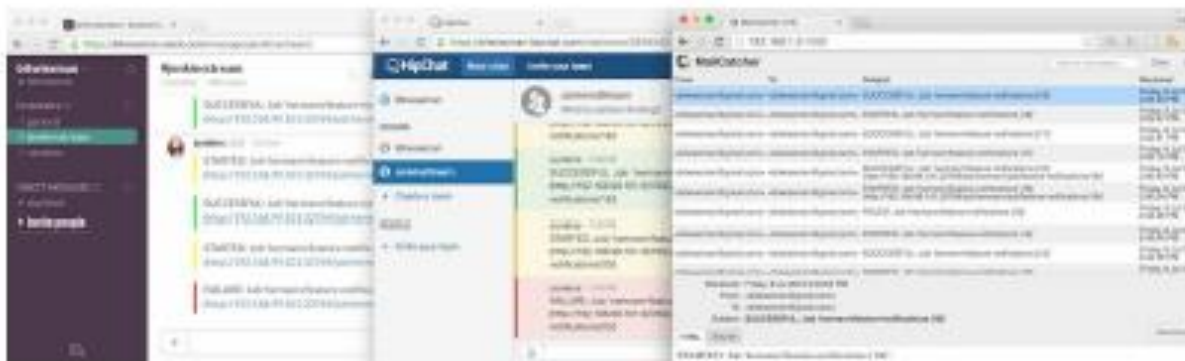
POURQUOI DES NOTIFICATIONS AVEC JENKINS ?

- Le modèle Jenkins fournit un reporting de données :
 - État de build
 - Mesures de l'heure (durée, date et heure de construction...)
 - Rapports de tests
 - Rapports d'analyse

- La mise en œuvre d'une boucle de rétroaction vous oblige à :
 - Savoir quand quelque chose se produit (échec de build, avertissement de test..)
 - Réagir en conséquence au bon message

TYPES DE NOTIFICATIONS DANS JENKINS

- Jenkins considère les notifications à propos des builds comme des 'Post Build Actions'
 - Exemple : 'Envoyer un mail si la compilation échoue'
- Les notifications sont envoyées par les canaux suivants:
 - Email
 - IMs (Skype, IRC, Jabber, Hipchat, Slack, etc.)
 - Autres (SMS, téléavertisseurs...)
- Support pour les notifications par e-mail et les flux RSS sont intégrés dans Jenkins; notifications sur d'autres canaux nécessitent généralement l'installation de plugins pour ces canaux.



CONFIGURATION ET UTILISATION

1. Installez les plugins requis
2. Configurer globalement dans Jenkins Management :
 - Service de notification (IRC, Jabber, Skype..)
 - Identifiants (généralement un token API)
 - Autres options spécifiques au fournisseur (rooms pour Jabber, intégrations de robots...)

3. Par projet, préciser :

- Quand voulez-vous déclencher un message ?

Changement de statut, avertissement, autre événement...

- Destinataires des messages ?

Peut dépendre du déclencheur (informer seulement le développeur qui a cassé le build)

- Qu'est ce que cela veut dire ?

- Il s'agit du contenu du message.

- Utiliser des modèles et des variables d'environnement pour avoir l'information « appropriée »; par exemple, on peut utiliser la variable d'environnement Jenkins BUILD_NUMBER pour inclure le numéro de compilation dans le sujet et le corps du message;

CONFIGURATION ET UTILISATION

1. Installez les plugins requis
2. Configurer globalement dans Jenkins Management :
 - Service de notification (IRC, Jabber, Skype..)
 - Identifiants (généralement un token API)
 - Autres options spécifiques au fournisseur (rooms pour Jabber, intégrations de robots...)

3. Par projet, préciser :

- Quand voulez-vous déclencher un message ?
Changement de statut, avertissement, autre événement...

- Destinataires des messages ?

Peut dépendre du déclencheur (informer seulement le développeur qui a cassé le build)

- Qu'est ce que cela veut dire ?
 - Il s'agit du contenu du message.
 - Utiliser des modèles et des variables d'environnement pour avoir l'information « appropriée »; par exemple, on peut utiliser la variable d'environnement Jenkins BUILD_NUMBER pour inclure le numéro de compilation dans le sujet et le corps du message;

POURQUOI UTILISER LES BUILDS DISTRIBUÉS ?

1. Les versions distribuées doivent être utilisées :
 - Si votre instance Jenkins n'a pas assez de ressources
 - Si vous voulez protéger votre `${JENKINS_HOME}` contre les compilations malveillantes
2. Lorsque les compilations doivent être effectuées sur des nœuds spécialisés (différents systèmes d'exploitation, archs CPU...)
 - Lorsque la charge de construction doit être distribuée pour une meilleure utilisation des ressources

C'EST QUOI UN BUILD DISTRIBUTÉ ?

- Les compilations distribuées sont des compilations exécutées sur des nœuds autres que le nœud principal.

Maître

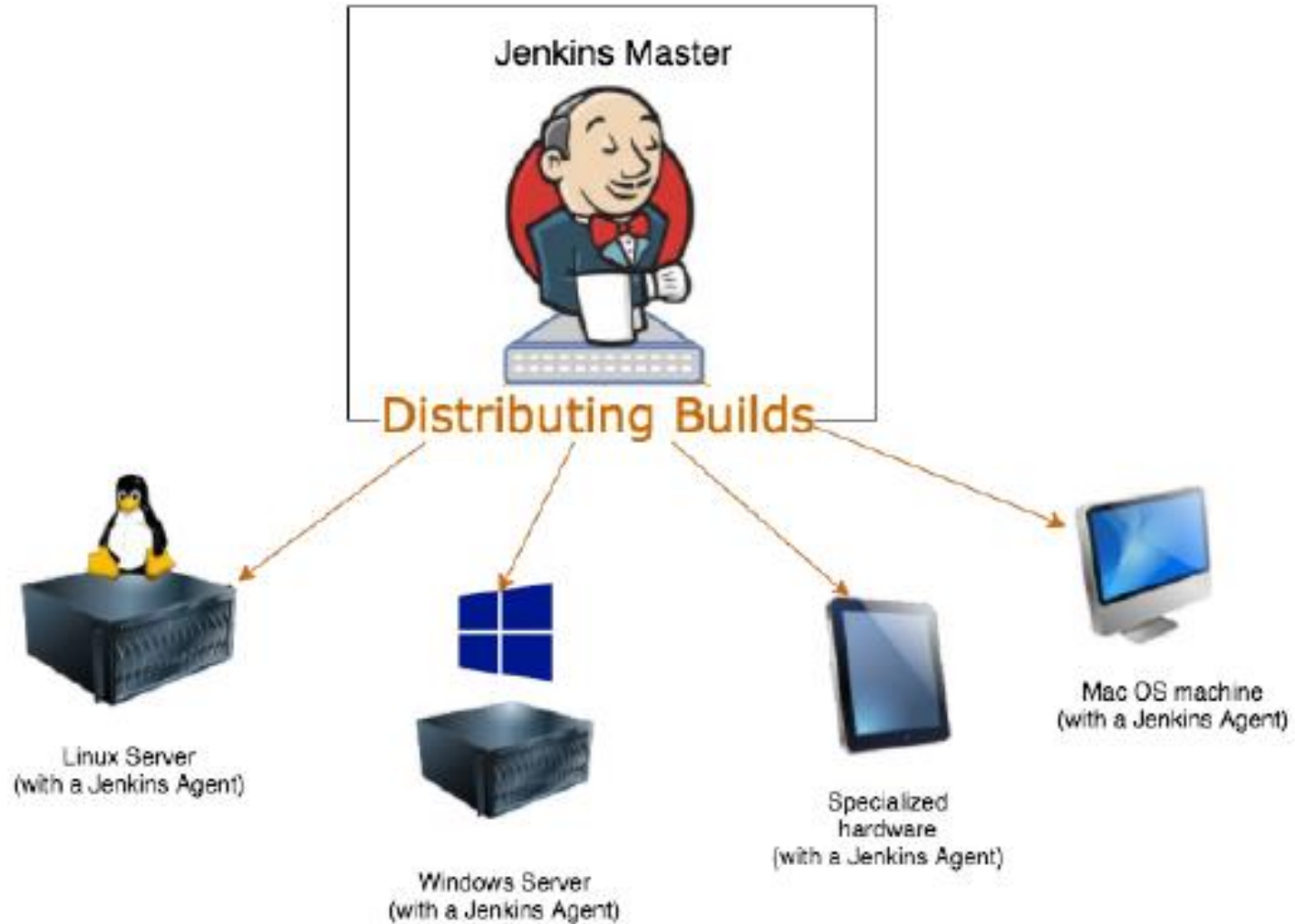
- Sert les requêtes HTTP
- Stocke toutes les informations importantes

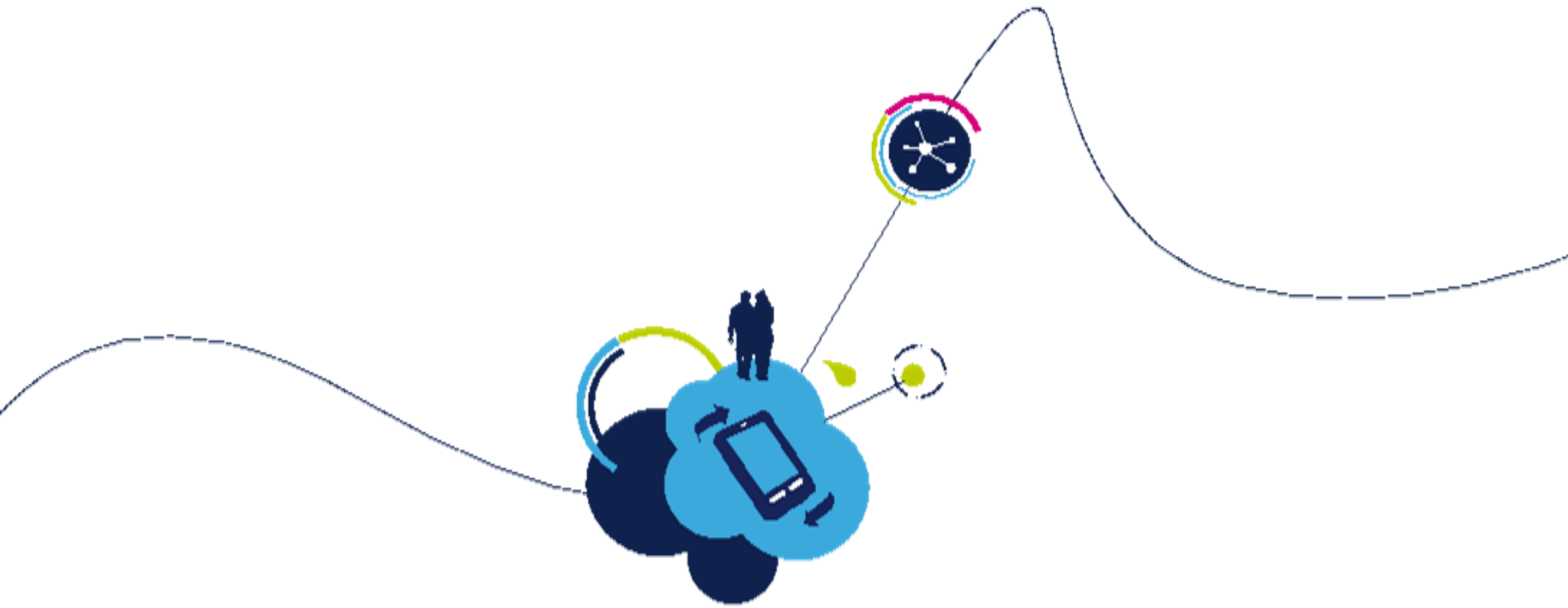
Agents

- Un jar de 170KB
- Présumé non fiable
- Passe à l'échelle d'au moins 100 agents

L'architecture de construction distribuée est une pratique exemplaire de Jenkins

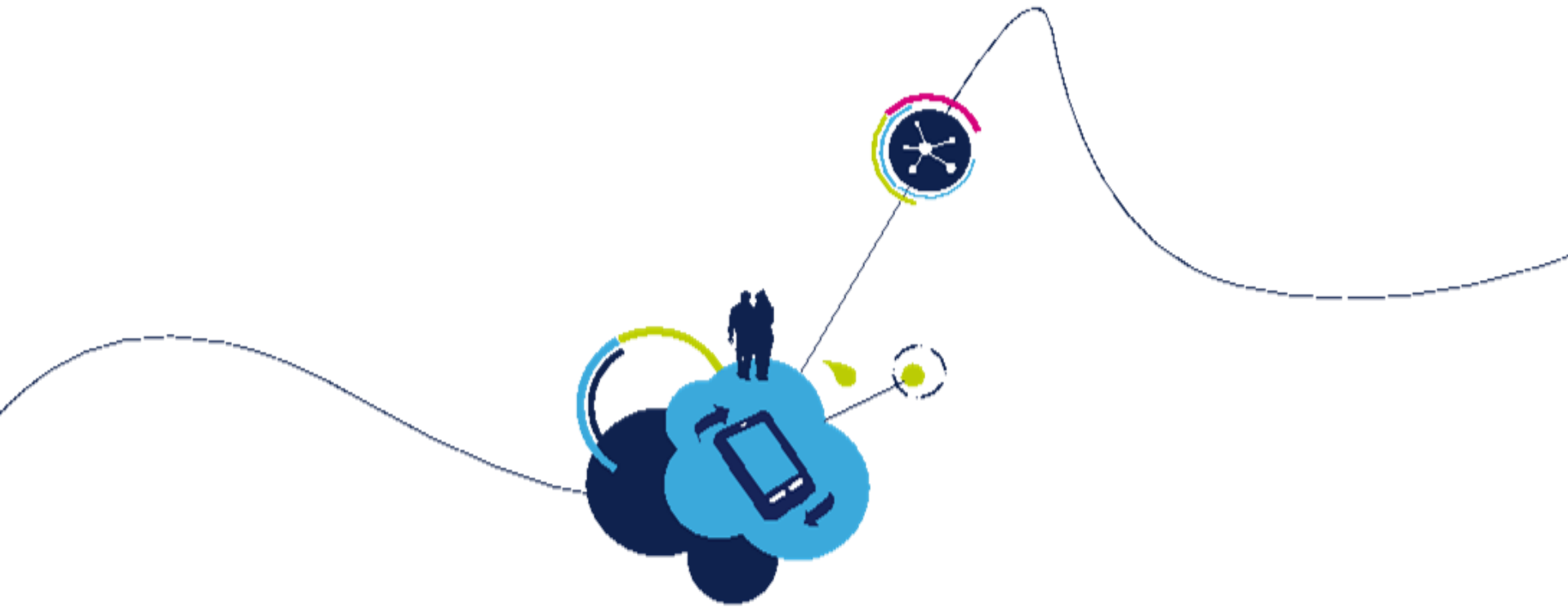
C'EST QUOI UN BUILD DISTRIBUÉ ?





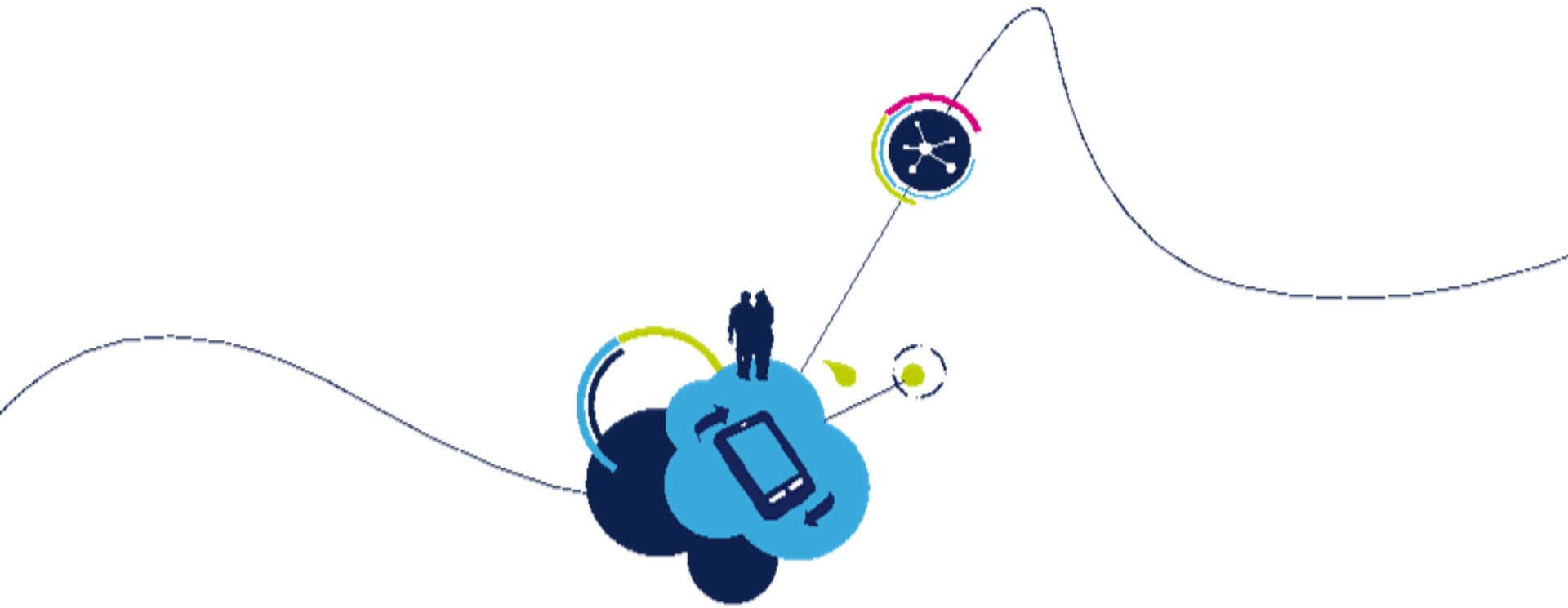
Lab n°2:

Construire une application avec Jenkins



Lab n°3:

Utilisation avancée de Jenkins



PIPELINES JENKINS MODERNES

Sommaire

- Les concepts de pipelines
- Les pipelines avancés
- Les bibliothèques partagées de pipeline


POURQUOI LES PIPELINES ?

- Les concepts de pipelines
- Les pipelines avancés
- Les bibliothèques partagées de pipeline


TYPE DE JOB PIPELINE

Enter an item name


= Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**


Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**


Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**External Job**


This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

**Multi-configuration project**


Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**

Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

VOCABULAIRE DE JENKINS 1/2

- **Maître:**

- C'est là où Jenkins est installé et exécuté
- Répond aux requêtes et s'occupe des tâches de build.

- **Agent:** (auparavant "esclave")

- Ordinateur configuré pour télécharger les projets disponibles du maître.
- A un certain nombre d'opérations à effectuer dans un périmètre bien défini.

VOCABULAIRE DE JENKINS 2/2

■ **Noeud:**

- ✓ Un ordinateur faisant partie du cluster Jenkins
- ✓ Peut être un maître ou un agent
- ✓ Nom générique que nous n'utiliserons pas ci-dessous.

■ **Exécuteur:**

- ✓ Ressource de calcul pour l'exécution de jobs
- ✓ Effectue les opérations
- ✓ Peut fonctionner sur n'importe quel noeud maître ou agent
- ✓ Peut être parallélisé sur un nœud spécifique

VOCABULAIRE DU PIPELINE

■ **‘STEP’:**

- ✓ Une seule tâche (aussi appelée « Étape de build »)
- ✓ Ça fait partie d'une séquence et spécifie à Jenkins quoi faire exactement.

■ **‘NODE’:**

- ✓ C'est un type de STEP , et non pas un nœud "Jenkins« , qui contient d'autres 'Steps'.
- ✓ Ordonnance les étapes contenues à travers les agents Jenkins et ses exécuteurs
- ✓ ça permet d'Orchestrer des espaces de travail éphémères sur des agents distants (leur création et leur suppression).

VOCABULAIRE DU PIPELINE

■ **‘STAGE’:**

- ✓ C'est un type de 'Step'.
- ✓ Une partie logiquement distincte des exécutions de tâches
- ✓ Peut avoir des paramètres pour le verrouillage, l'étiquetage et la commande.
- ✓ Peut comporter une ou plusieurs étapes de build.
- ✓ La bonne pratique consiste à l'utiliser les Stages, notamment pour la visualisation.

VOCABULAIRE DU PIPELINE

■ **‘STAGE’:**

- ✓ C'est un type de 'Step'.
- ✓ Une partie logiquement distincte des exécutions de tâches
- ✓ Peut avoir des paramètres pour le verrouillage, l'étiquetage et la commande.
- ✓ Peut comporter une ou plusieurs étapes de build.
- ✓ La bonne pratique consiste à l'utiliser les Stages, notamment pour la visualisation.

ANATOMIE DE L'EXECUTION DES PIPELINES

- ✓ Les scripts Pipeline Groovy sont analysés et exécutés sur le master:
 - les blocs de nœuds allouent les exécuteurs et les espaces de travail du maître.
- ✓ Les agents gèrent toujours les opérations, en faisant tourner des exécuteurs, selon périmètre.
- ✓ Exploitation du pipeline par des exécuteurs légers:
 - Exécuteur non pris en compte : créneau temporaire
 - Utilise très peu de puissance de calcul.
 - Représente un script Groovy inactif en attente d'une Step à terminer

PIPELINE-AS-CODE

- 'Jenkinsfile':
 - ✓ Pour utiliser cette approche, les projets doivent contenir un fichier nommé *Jenkinsfile* dans la racine du dépôt et qui contient un "Pipeline script."
 - ✓ Le principal Groovy script d'un pipeline est un *Jenkinsfile*
 - ✓ Le fichier *Jenkinsfile* peut être stocké dans divers SCM (Git, SVN, etc...):
 - Toujours configurable via GUI (mauvaise pratique).
 - Avantages du SCM :

Appliquer le versioning, revue de code, jouer les différents tests et intégrer avec la définition de votre pipeline de déploiement continue (CD).

PIPELINE-AS-CODE

- 'Jenkinsfile':
 - ✓ Pour utiliser cette approche, les projets doivent contenir un fichier nommé *Jenkinsfile* dans la racine du dépôt et qui contient un "Pipeline script."
 - ✓ Le principal Groovy script d'un pipeline est un *Jenkinsfile*
 - ✓ Le fichier *Jenkinsfile* peut être stocké dans divers SCM (Git, SVN, etc...):
 - Toujours configurable via GUI (mauvaise pratique).
 - Avantages du SCM :

Appliquer le versioning, revue de code, jouer les différents tests et intégrer avec la définition de votre pipeline de déploiement continue (CD).

LES PIPELINES

- **Exigences relatives aux pipelines:**
 - ✓ Au moins **Pipeline plugin**: (anciennement Workflow plugin).
 - ✓ Fonctionne avec une suite de plugins connexes qui améliorent la fonctionnalité.
 - ✓ Les plugins associés ajoutent une syntaxe de pipeline ou des visualisations.
 - ✓ Il est recommandé de commencer par:
 - **Pipeline workflow-aggregator**: installe les plugins de base et les dépendances
 - **Pipeline Stage View**
 - **Multibranch Pipeline**
 - **Pipeline Docker**

PIPELINE HELLO WORLD

Simple exemple

- Alloue un exécuteur.
- Affiche la chaîne de caractères

```
node {  
    echo 'Hello from Pipeline'  
}
```

PAR OÙ COMMENCER ? ECRIRE EN GUI

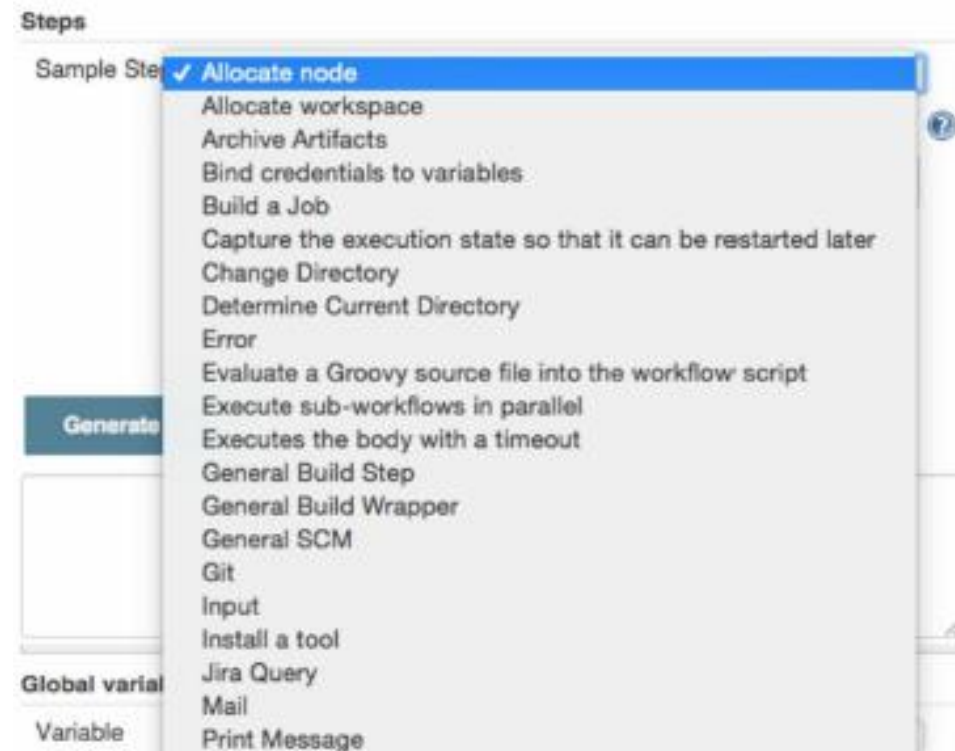
- Utilisation d'un SCM est recommandé. Cependant, l'interface graphique de Jenkins aide beaucoup
- Créer une nouvelle page de configuration de Job Pipeline :
Éditeur de texte coloré + exemples



PAR OÙ COMMENCER ? GENERATEUR DE CODE

Vous ne connaissez pas l'étape individuelle?

1. Utilisez le générateur de code pour créer des exemples de syntaxe
 2. Copiez les extraits générés par le passé dans vos scripts
- Page dédiée de l'interface graphique de Jenkins
 - Rempli dynamiquement avec les steps disponibles
 - Dépend des plugins installés



PIPELINE : EXEMPLE SIMPLE

1. Attribue un exécuteur avec un périmètre (sur un agent avec label « cpp »)
2. Clone le dépôt git spécifié.
3. Exécute une commande shell pour la construction.

```
node('cpp') {  
  git url: 'https://github.com/joe_user/cpp-app.git'  
  sh "make build"  
}
```

VUE DES ÉTAPES DU PIPELINE

- Fourni la visualisation du pipeline en fonction des steps.
- Pas de vue Jenkins : c'est l'interface graphique sur la page du job
- Montre une matrice avec l'historique de construction et les steps comme dimensions

	Test	Re-test	Deploy	Deploy again	Deploy and again	Keep deploying	Final deploy	Clean-up
#12 Oct 30 15:45 No Changes Retry	10s	9s	18s failed					
#11 Oct 30 15:44 No Changes	10s failed							
#10 Oct 30 15:42 No Changes Retry Download	9s	21s	9s	9s	22s	300ms	9s	42ms
#9 Oct 30 15:37 No Changes Retry Download	10s	10s	22s	9s	10s	313ms	21s	73ms
#8 Oct 30 15:34 No Changes Retry	9s	21s	9s	9s	22s failed			
Average stage times:	9s	12s	12s	7s	13s	153ms	7s	28ms

AVANTAGES DE LA VUE DES ÉTAPES DU PIPELINE

- Obtenir (une autre) rétroaction : une rétroaction visuelle
- Suivi des défaillances plus facile :
 - Isoler la défaillance à un stade précis
 - Le journal de sortie est visualisable « par step »
- La visualisation est liée à une seule tâche et à un seul pipeline
- Affiche les données du pipeline :
 - Date, heure, changements (avec les liens des changements)
par compilation
 - Temps d'exécution par construction et par step.
 - Registres d'état et de sortie par step.

PIPELINE : SYNTAXE ET OUTILS APACHE GROOVY

- Utilisation de l'installation de MAVEN3 comme outil Jenkins
- Syntaxe des variables groovy (**def mvnHome =**)
- Exécuter une compilation Maven dans le dépôt cloné

```
node() {  
    git url: 'https://github.com/joe_user/simple-maven-project-with-te  
    def mvnHome = tool 'MAVEN3'  
    sh "${mvnHome}/bin/mvn -B verify"  
    // Windows syntax instead of sh:  
    // bat "${mvnHome}\\bin\\mvn -B verify"  
}
```


PIPELINE : PÉRIMÈTRES

- La syntaxe groovy utilise la syntaxe "Scope" (~ fonctions anonymes).
- Exemple d' exécution d'un script dans un sous-dossier

```
node() {  
  git url: 'https://github.com/joe_user/dockerized-app.git'  
  dir('scripts') {  
    sh 'bash ./admin-script.sh'  
  }  
}
```

PIPELINE : ENVIRONNEMENT

- Le DSL pipeline fournit la variable « env »
- Ses propriétés sont des variables d'environnement sur le noeud courant.
- Peut surcharger certaines variables d'environnement; changement observé dans les étapes suivantes.
- Exemple de gestion des outils Maven pour agent fongible:

```
node {  
  git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'  
  withEnv(["PATH+MAVEN=${tool 'MAVEN3'}/bin", "M2_HOME=${tool 'MAVEN3'}"]) {  
    sh 'mvn -B verify'  
  }  
}
```

PIPELINE : ENVIRONNEMENT

- Le DSL pipeline fournit la variable « env »
- Ses propriétés sont des variables d'environnement sur le noeud courant.
- Peut surcharger certaines variables d'environnement; changement observé dans les étapes suivantes.
- Exemple de gestion des outils Maven pour agent fongible:

```
node {  
  git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'  
  withEnv(["PATH+MAVEN=${tool 'MAVEN3'}/bin", "M2_HOME=${tool 'MAVEN3'}"]) {  
    sh 'mvn -B verify'  
  }  
}
```

PIPELINE: STAGES

- Représente une étape abstraite.
- S'attend à un "label", une chaîne fournie comme argument.
- 'Stage' est un bloc avec périmètre.

```
node {  
  stage('Checkout SCM') {  
    git url: 'https://github.com/jglick/simple-maven-project-with-tes  
  }  
  stage('Build') {  
    sh 'mvn -B verify'  
  }  
}
```

PIPELINE : APPROBATION MANUELLE

- Blocage du flux d'exécution jusqu'à une validation humaine.
- Vous pouvez « régler » le message, les boutons...
- Échouera la compilation si le bouton "Non" est enfoncé.
- Bonnes pratiques :
 - Exécutez-le à l'extérieur d'un nœud pour ne pas bloquer pas un exécuteur
 - Utilisez timeout pour éviter d'attendre une quantité infinie de temps
 - Utilisez des structures de contrôle d'exceptions (Try/Catch/Finally)

```
stage('Waiting for Approval') {  
    input message: "Does http://localhost:8888/staging/ look good?"  
}
```

PIPELINE : STEPS PARALLÈLES

- Les steps peuvent être exécutées en parallèle :
 - Longue étape de fonctionnement pour optimiser Pipeline.
 - Différents cas d'utilisation indépendante.
- Chaque "branche parallélisée" est une étape.

```
parallel 'integration-tests':{  
  node('mvn-3.3'){  
    sh 'mvn verify'  
  }  
}, 'functional-tests':{  
  node('selenium'){  
    sh 'bash /run-selenium-tests.sh'  
  }  
}
```

PIPELINE : CONTRÔLE DE L'EXÉCUTION

- Contrôler l'exécution de votre pipeline :

```
// Try up N times  
retry(10) { . . . }
```

```
// Pause the flow:  
sleep time: 10, unit: 'MINUTES'
```

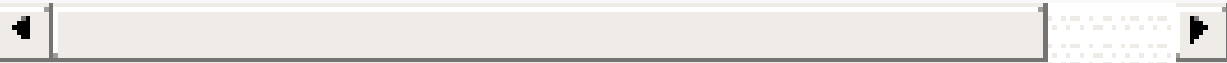
```
// Wait for event  
waitUntil { . . . }
```

```
// Timeout an operation  
timeout(time: 100, unit: 'SECONDS') { . . . }
```

PIPELINE : SYSTÈME DE FICHIERS

- Read file:

```
readFile file: 'some/file', encoding: 'UT
```



- Write file:

```
writeFile file: 'some/file', text: 'hello'
```


PIPELINE : PLUGINS PRIS EN CHARGE

- Si un plugin est conforme à Pipeline, il fournit plus de mots-clés
- Exemple avec l'éditeur de rapport de test junit :

```
stage('Build') {  
  sh 'mvn clean install -fn'  
  junit './target/**/*.xml'  
}
```

- Syntaxe d'invocation pour les plugins non supportés:
Jusqu'à ce qu'ils puissent évoluer pour offrir un soutien natif aux pipelines:

```
step( // DSL keyword to invoke arbitrary step  
  [ // groovy map  
    $class: 'build_step_class_name', // Same as in job's config.xml on d  
    constructor_argument: 'value',  
    constructor_argument_2: 'value'  
  ]  
)  
// Exemple with the Archive Artifact step:  
step([$class: 'ArtifactArchiver', artifacts: 'target/**/*.jar'])
```

PIPELINES AVANCÉS

MULTI-BRANCH PIPELINES ?

Un job jenkins pipeline présente les défis suivants:

- ✓ Il ne représente qu'une seule branche du SCM
- ✓ Pas de découverte automatique
- ✓ Aucune séparation des préoccupations

DEBUTER AVEC LES PIPELINES MULTI-BRANCH

1. Créer le Multi-Branch
2. Configurer votre source SCM
3. (En option) Configurer un webhook depuis SCM
4. Envoyer un fichier Jenkins sur n'importe quelle branche
5. Branche de merge : jobs gérées automatiquement
6. Tout est automatisé : plus de cauchemar pour l'administrateur Jenkins

CONFIGURATION DES PIPELINES MULTI-BRANCH

1. Politique de rétention **personnalisable**
 - Section de configuration "Orphaned Item Strategy«
2. **Sécurisé** : Run Pipeline dans le bac à sable de Groovy
 - Code considéré comme "non sécurisé" nécessite une validation admin
 - Éviter l'exécution de code inconnu sans protection
 - Fournir des variables supplémentaires pour les pipelines plus complexes: `BRANCH_NAME` et `CHANGE_ID`

ANALYSE DE L'ORGANISATION

- Utiliser un SCM hébergé avec Jenkins (Github, Bitbucket, etc.) ?
 - Les plugins correspondants doivent être installés

- L'administrateur configure l'organisation pour ce type de Jobs
 - Un justificatif d'identité (token API généralement) nécessaire
 - Correspond à un "dossier d'organisation" au niveau supérieur

- Chaque projet correspond à un pipeline Multi-branch:
 - Dans le "dossier organisation"
 - Plus d'automatisation
 - Automatiser la création de webhooks

BIBLIOTHÈQUES PARTAGÉES DE PIPELINE

- D.R.Y. : ne vous répétez pas!
- Augmente l'utilisation de votre pipeline Jenkins:
 - Plus de projets
 - Plus d'équipes
- Tire parti du coût de maintenance:
 - Écrivez une fois, propagez partout
 - Pipeline as code partout
- Utilisation des outils pour éviter les silos:
 - Collaborer au lieu de forcer

QU'EST-CE QU'UNE BIBLIOTHÈQUE PARTAGÉE DE PIPELINE?

- ✓ Un ensemble d'SCM contenant un code de pipeline réutilisable.
- ✓ Configuré 1 fois à l'intérieur de Jenkins.
- ✓ Cloné au moment de la compilation.
- ✓ Chargé et utilisé comme bibliothèque de codes sur Jenkins Pipelines.

COMMENT UTILISER LES BIBLIOTHÈQUES PARTAGÉES DE PIPELINE?

- Configurer 1 (ou plus) là où vous en avez besoin :
Code de confiance : par les administrateurs au niveau Jenkins
Code non fiable : par les développeurs au niveau des Multi-branch/dossiers
- Définir les politiques :
Branche par défaut / tag / changeset
Les développeurs peuvent-ils remplacer la version par défaut ?
- Chargez-le depuis votre pipeline :
Par annotation
Par mot-clé DSL
Implicitement

NOTE SUR LES BIBLIOTHÈQUES PARTAGÉES

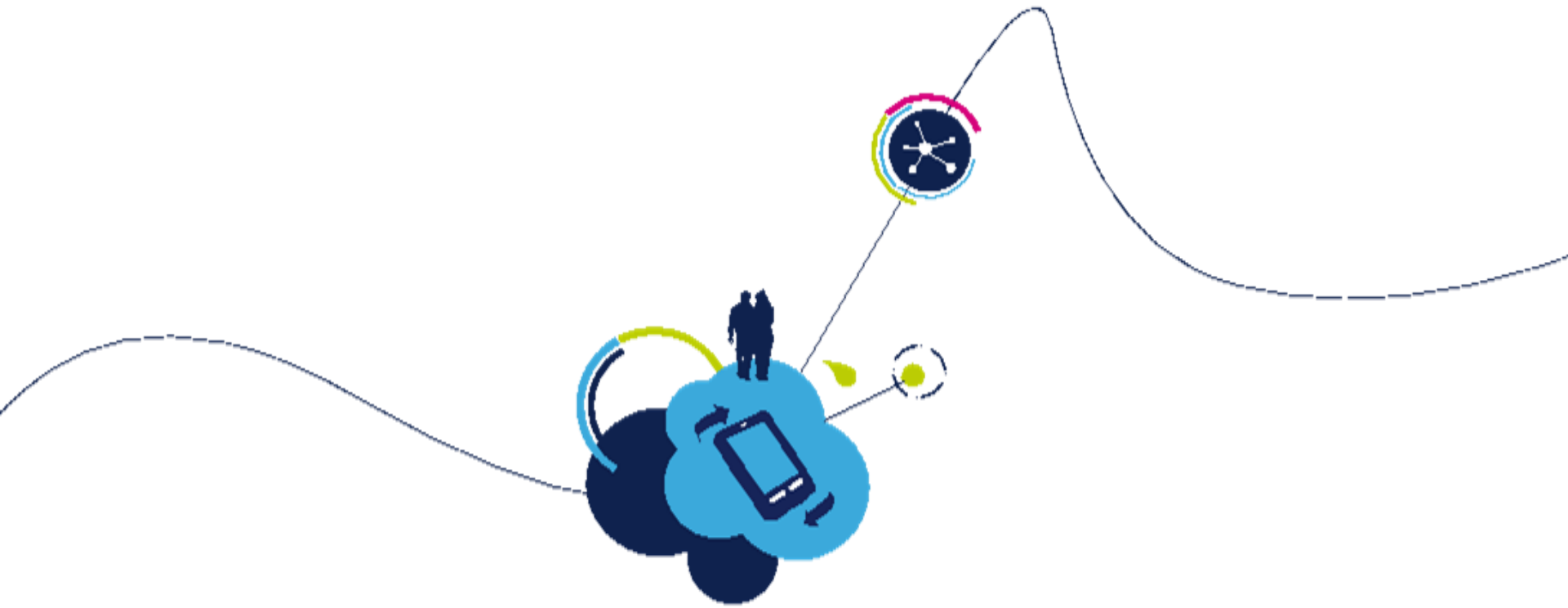


- Extrêmement puissant, à forte valeur ajoutée
- Courbe d'apprentissage : la 1ère étape n'est pas facile
- C'est un code qui doit être testé

Ajoute un coût supplémentaire: investissement en temps

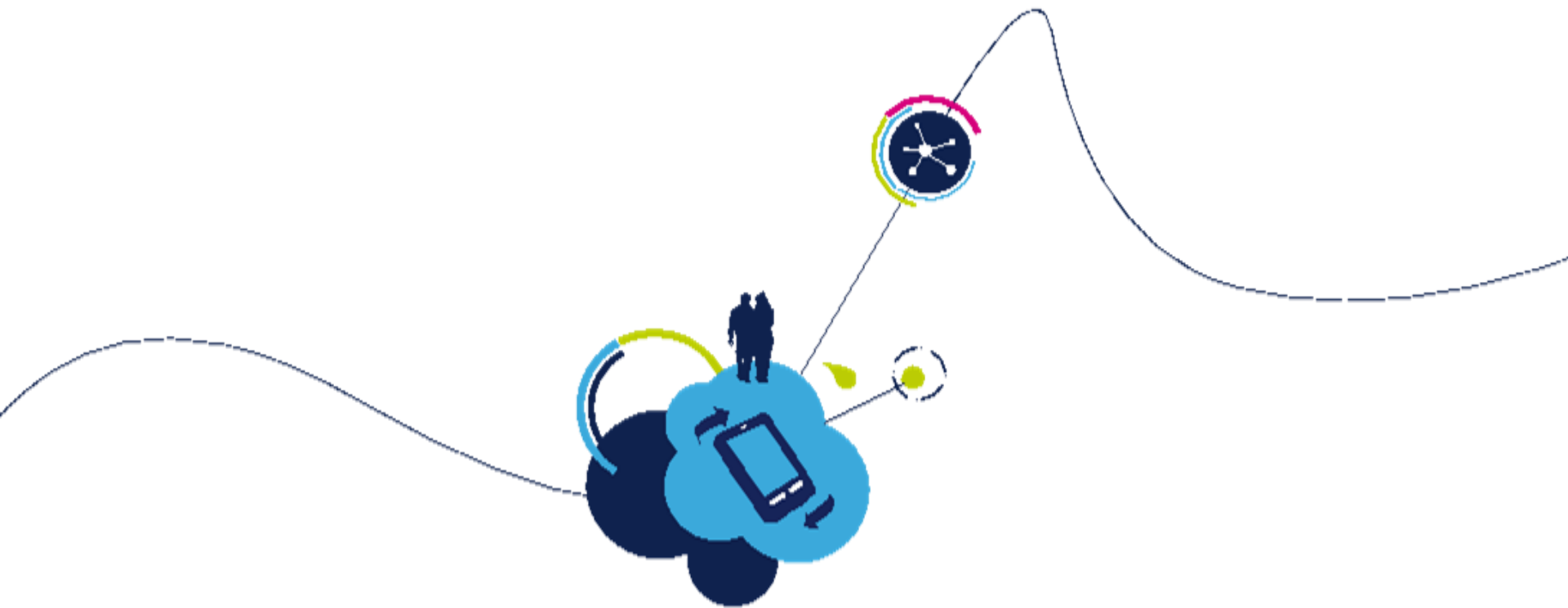
- De nombreuses utilisations:

Prenez le temps de lire la documentation



Lab n°4:

Pipelines Jenkins Modernes



Merci