

# FORMATION JENKINS

Formation Jenkins, mettre en place l'intégration continue en Java

Badr NASS LAHSEN  
Le 16/09/2019

1

## Présentation

- Présentation du formateur
- Tour de table: Rôles et environnement
- Objectifs et attentes de la formation
- Horaires et pauses
- Plan de formation
- Questions?

## Plan de cours

- Introduction à l'intégration continue
- Mise en place et automatisation du Build
- Qualité du code
- Automatisation des tests
- Automatisation du déploiement
- Jenkins avec Docker
- Administration d'un serveur Jenkins
- Aller plus loin avec Jenkins Enterprise By CloudBees

## Tour de table

Expérience dans les projets avec Jenkins?

Avez-vous travaillé en mode Agile?

Background: Dev / Ops / Autres ?

## Vos attentes

Qu'attendez-vous  
de la formation ?

## INTRODUCTION À L'INTÉGRATION CONTINUE

## CE QUI A CHANGÉ : LE DIGITAL

On parle de **coût informatique** mais où est la **valeur métier**

Le **TTM** est toujours là mais pour **faire quoi ?**

L'informatique pour **tout le monde, dans tous les objets**

Pour aller chercher de la productivité complémentaire, je n'ai **pas besoin d'un chef mais d'autonomie**

## LES IMPACTS : AUTONOMIE, TTM & INNOVATION, MULTIPLICATION DES DEVICES

### **Explosion du nombre de mises en productions**

- Plus de MEP pour tester plus souvent
  - Tous les 3 à 6 mois (2 à 4) => A tous les mois (12) **x3 à x6**
- Plus de MEP car les « grosses applications » ont été découpées en produits agiles
  - 1 application sert entre 2 à 4 métiers **x2 à x4**
- Plus de MEP pour plus de devices, de cibles
  - 1 fonctionnalité va s'exécuter en Web, Mobile, ... **x2**

### **Les activités manuelles du cycle de production x12 à x48**

## L'INTÉGRATION CONTINUE

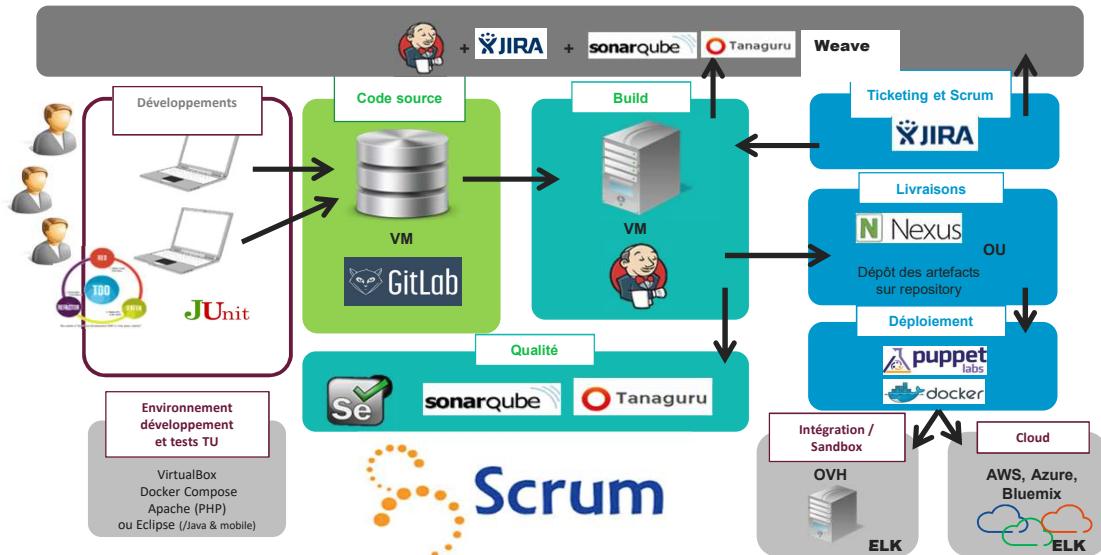
- L'intégration continue consiste à vérifier à chaque modification du code source que le résultat des modifications ne produit **pas de régression**
- Objectif :
  - Garantir la qualité du logiciel (Exécution de tests, Code Style)
  - S'assurer que chaque version du code peut produire un livrable correct
- Les principaux avantages d'une telle technique sont :
  - Le test quasi-immédiat des modifications
  - La notification rapide en cas de code incompatible ou manquant
  - Les problèmes d'intégration sont détectés et réparés de façon continue, évitant les problèmes de dernière minute
  - Une version est toujours disponible pour un test, une démonstration ou une distribution

## L'INTÉGRATION CONTINUE

[M. Fowler 2008]

- *Continuous Integration is a **software development practice** where members of a team integrate their work frequently, usually **each person integrates at least daily** leading to multiple integrations per day.*
- *Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible*
  - Reference: <http://martinfowler.com/articles/continuousIntegration.html>

## EXEMPLE: CHAINE D'INTEGRATION CONTINUE



Formation Jenkins

11

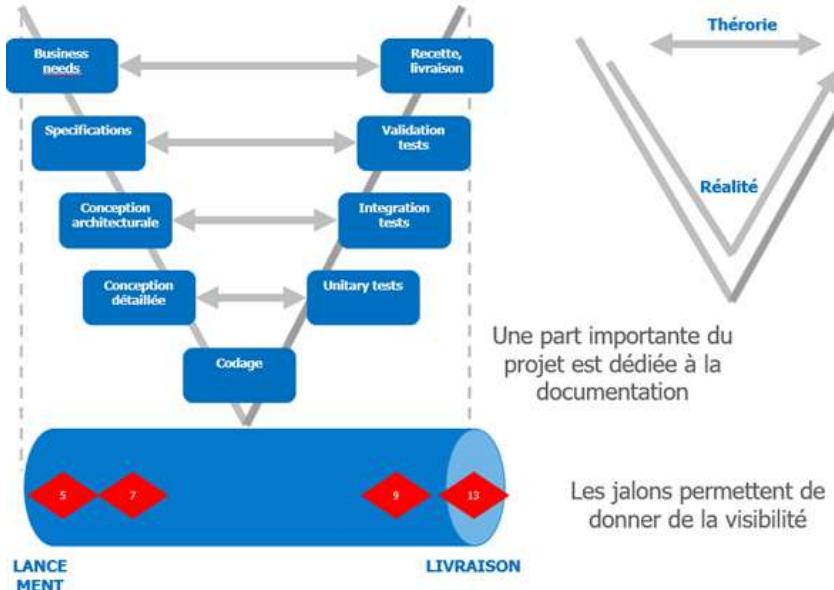
## COMMENT METTRE EN PLACE L'INTEGRATION CONTINUE?

- Pour mettre en place un système d'intégration continue complet il faut :
  - Maintenir un dépôt unique de code source versionné et commiter le plus souvent possible
  - Versionner les modifications de base de données
  - Automatiser les compilations
  - Ecrire des tests unitaires
  - Assurer la qualité du code
  - Déployer sur un environnement de développement (au plus proche de la production)
  - Automatiser les précédentes étapes.

Formation Jenkins

12

## APPROCHE CLASSIQUE: CYCLE EN V



Formation Jenkins

13

## APPROCHE CLASSIQUE: CYCLE EN V

- Les phases séquentielles
  - Recueil des besoins
  - Analyse détaillée
  - Conception détaillée
  - Développement
  - Tests, contrôle qualité
  - Intégration
- Effet tunnel pour le projet
- Une fois le plan de management du projet validé, il constitue la référence de base. La préoccupation majeure du chef de projet devient alors de coller au plus près au plan quels que soient les évènements; tout écart constaté ... est perçu comme un change ou un échec, vécu par certains comme une incompétence ou une incapacité à anticiper

Formation Jenkins

14

## APPROCHE CLASSIQUE: CYCLE EN V

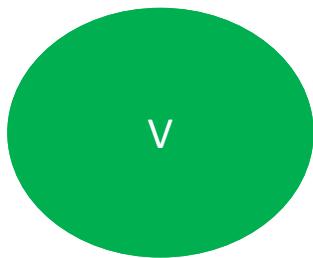
- Course de relais
- Séquentielle et prédictive
- Non rapide et non flexible
- Grosse pression sur les délais
- Négation dangereuse que le périmètre peut évoluer
- Paradoxe et difficulté du sous-traitant à s'engager
- Le sous-traitant minimise l'estimation quitte à se rattraper lors des changes durant le projet

## APPROCHE AGILE

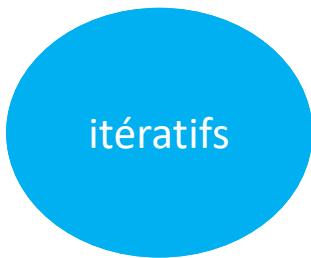
- La notion de méthode agile a été officialisée en 2001 par un document **Manifeste Agile** (*Agile Manifesto*) signé par 17 personnalités impliquées dans l'évolution du génie logiciel
- « Une méthode agile est une approche itérative et incrémentale, qui est menée dans un esprit collaboratif avec juste ce qu'il faut de formalisme. Elle génère un produit de haute qualité tout en prenant en compte l'évolution des besoins des clients »
- Une méthode agile est donc
  - **Itérative**
  - **Incrémentale**
  - **Adaptative**
  - **Collaborative**

## MÉTHODES : V, AGILE, SCRUM, ITÉRATIFS ?

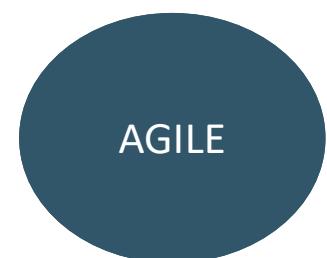
3 modes de fonctionnement possibles  
et un éventail de possibilités



1 coût, 1 délai, 1 livraison et 1 périmètre



1 coût, 1 délai ,n livraisons et 1 périmètre



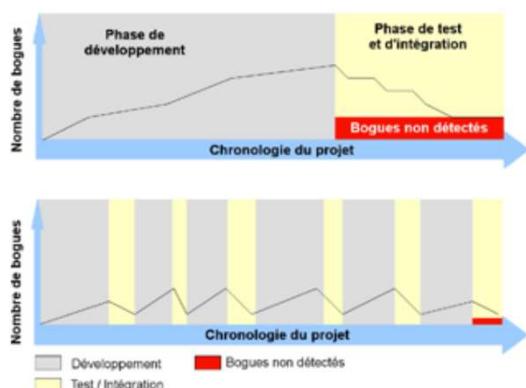
1 coût, 1 délai , n livraisons et un périmètre ajusté continuellement.

Formation Jenkins

17

## AVANTAGE DE L' AGILE ET INTÉGRATION CONTINUE

- Une méthode agile 😊



- Principe : plus l'intégration est fréquente, moins elle est longue

Formation Jenkins

18

## DEVOPS

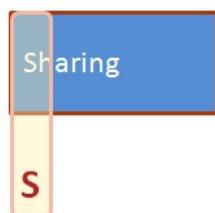
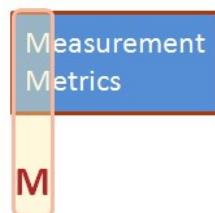
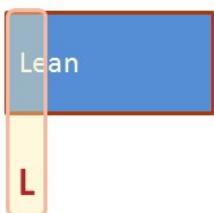
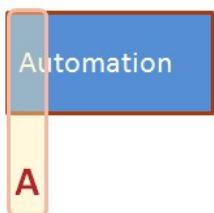
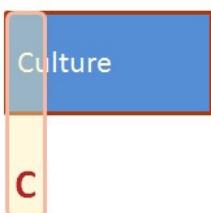
- Le mouvement **DevOps** est né de ce constat d'échec. On le symbolise par l'acronyme **CALMS**

DevOps n'est pas :

- Une méthodologie
- Un outil
- Une norme



« **CALMS** » représente les **5 piliers fondamentaux** pour une mise en œuvre réussie de la philosophie DevOps



Formation Jenkins

19

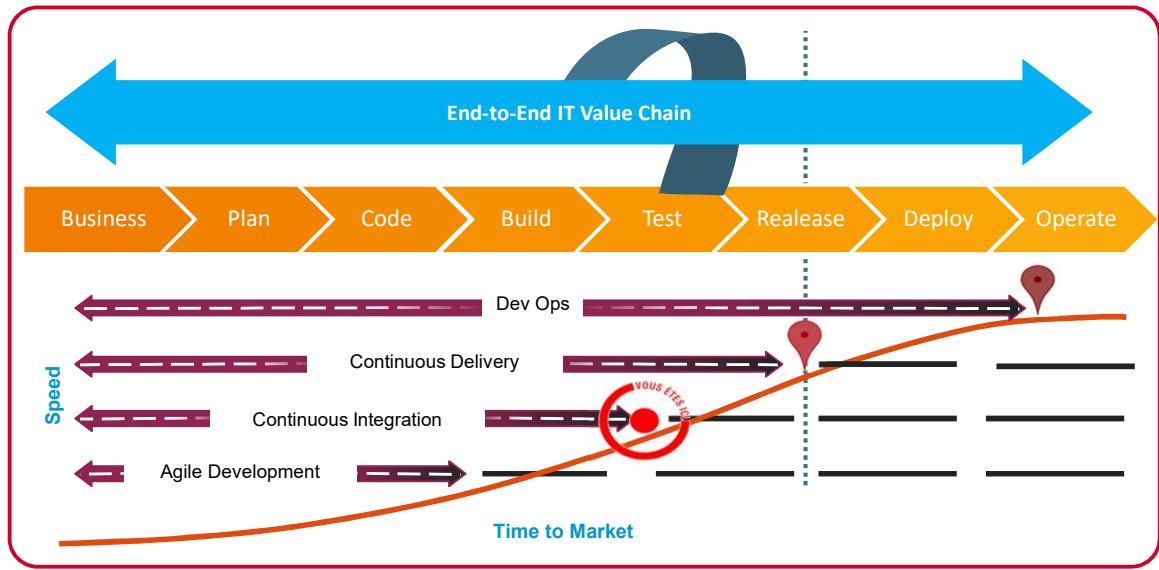
## 7 ÉTAPES D'ÉLABORATION INTÉGRATION CONTINUE

- 1) Pas de serveur de build
- 2) Builds quotidiens
- 3) Builds quotidiens et tests automatisés basiques
- 4) Arrivée des métriques
- 5) Prendre les tests au sérieux
- 6) Tests d'acceptation automatisés et un déploiement plus automatisé
- 7) Déploiement Continu

Formation Jenkins

20

## OÙ ON VA?



Formation Jenkins

21



## MISE EN PLACE ET AUTOMATISATION DU BUILD

Formation Jenkins

22

## JENKINS

Over 1.7 Million users

1400+ plugins cover  
every stage of delivery



| Formation Jenkins

23

## HISTORIQUE JENKINS / HUDSON

- Jenkins est le produit d'un développeur visionnaire, Kohsuke Kawaguchi, qui a commencé ce projet comme un loisir sous le nom d'Hudson à la fin de l'année 2004 alors qu'il travaillait chez Sun.
- Début 2008, Sun reconnaissait la qualité et la valeur de cet outil et demandait à Kohsuke de travailler à plein temps sur Hudson
- En 2010, Hudson était devenu la principale solution d'Intégration Continue avec une part de marché d'environ 70%.
- En 2009, Oracle racheta Sun. Vers la fin 2010, des tensions apparaissent entre la communauté de développeurs d'Hudson et d'Oracle

| Formation Jenkins

24

## HISTORIQUE JENKINS / HUDSON

- En Janvier 2011, la communauté des développeurs Hudson vota pour renommer le projet en Jenkins.
- La grande majorité des développeurs du cœur et des plugins suivit Kohsuke Kawaguchi
- Après ce fork, la majorité des utilisateurs suivit la communauté des développeurs Jenkins et passa à Jenkins: Plus de 75% des utilisateurs d'Hudson sont passés à Jenkins.

## HISTORIQUE

THE LINUX FOUNDATION PROJECTS



- La Fondation Linux annonce la création d'une fondation dédiée aux projets open source d'intégration et de déploiement continu, parmi lesquels Jenkins X et Spinnaker: la Continuous Delivery Foundation (CD.Foundation ou CDF).
- Celle-ci va chapeauter plusieurs projets open source d'intégration et de déploiement continu (continuous integration/continuous delivery ou CI/CD) d'applications et microservices.
- CDF se lance avec 19 membres, dont des grandes enseignes telles que Google, Netflix, Red Hat, Alibaba, Autodesk, SAP, Huawei et GitLab. Les systèmes Open Source Jenkins, Jenkins X, Spinnaker (créé par Netflix et dirigé conjointement par Netflix et Google) et Tekton sont parmi les premiers projets hébergés par CDF, a déclaré la Linux Foundation. La fondation mère a déclaré s'attendre à ce que davantage de projets soient ajoutés au CDF, une fois qu'elle aura formé un comité de

### PROJECTS



## JENKINS DOMINE LE MARCHÉ DU DÉPLOIEMENT CONTINU

- Plus de 60 % des professionnels Java interrogés utilisent Jenkins
- Plus de 133 000 installations de Jenkins actives au 31 décembre 2016
- Plus de 1 400 plugins de la communauté Jenkins destinés à être intégrés à des technologies tierces ou à ajouter de nouvelles capacités
- Capacités avancées de déploiement continu et d'automatisation des DevOps grâce au pipeline Jenkins

## PRÉPARATION D'UN SERVEUR DE BUILD POUR JENKINS

- Conditions préalables
  - Configuration matérielle minimale: 256 Mo de RAM
    - 1 Go d'espace disque (bien qu'un minimum de 10 Go soit recommandé si Jenkins est utilisé comme conteneur Docker)
  - Configuration matérielle recommandée pour une petite équipe:
    - 1 Go de RAM 50 Go + d'espace disque
    - Exigences de logiciel: Java 8 - soit un environnement d'exécution Java (JRE) ou un kit de développement Java (JDK) est bien

## LES DIFFÉRENTS TYPES D'INSTALLATION

- Jenkins peut s'installer comme:
  - À partir d'une image docker
  - Un war sur un conteneur comme Tomcat
  - JVM autonome avec un war exécutable
  - Ou bien directement comme service depuis un rpm ou exe
- Les packages de jenkins sont disponibles:
  - <https://jenkins.io/download/>

## LE RÉPERTOIRE DE TRAVAIL DE JENKINS: USER\_HOME/.JENKINS

- Peu importe où se trouve le fichier WAR de Jenkins, Jenkins conserve toutes ses données importantes dans un répertoire spécial séparé appelé le répertoire de travail Jenkins.
- Par défaut, le répertoire de travail Jenkins sera appelé .jenkins, et sera placé dans votre répertoire de travail.
- On peut forcer Jenkins à utiliser un répertoire différent pour son répertoire de travail en définissant la variable d'environnement JENKINS\_HOME .

## INSTALLATION DE JENKINS COMME WAR EXECUTABLE

- Téléchargez le dernier fichier WAR Jenkins stable dans un répertoire approprié de votre machine.  

```
curl -L -O http://mirrors.jenkins.io/war-stable/latest/jenkins.war
```
- Puis exécutez la commande:  

```
java -jar jenkins.war &
```
- Accédez à jenkins, via l'url:
  - <http://localhost:8080>

## INSTALLATION DE JENKINS DANS UN TOMCAT

- Procédure d'installation avec Tomcat
  - Installation de JDK
  - Configuration des variables d'environnement
    - JAVA\_HOME
    - PATH
- Installation de Tomcat
- Déploiement du package war de jenkins dans le répertoire webapps
- Démarrer tomcat
- L'url :
  - <http://localhost:8080/jenkins>

# CONFIGURATION DE JENKINS

- Se connecter avec le mot de passe affiché dans les logs ou disponible via le chemin:

/var/lib/jenkins/secrets/initialAdminPassword

## Débloquer Jenkins

Pour être sûr que que Jenkins soit configuré de façon sécurisée par un administrateur, un mot de passe a été généré dans le fichier de logs ([où le trouver](#)) ainsi que dans ce fichier sur le serveur :

`/root/.jenkins/secrets/initialAdminPassword`

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

Continuer

# CONFIGURATION DE JENKINS

- Choisir les plugins:
  - Installer les plugins suggérés

## Personnaliser Jenkins

Les plugins étendent Jenkins avec des fonctionnalités additionnelles pour satisfaire différents besoins.

### Installer les plugins suggérés

Installer les plugins que la communauté Jenkins trouve les plus utiles.

### Sélectionner les plugins à installer

Sélectionner et installer les plugins les plus utiles à vos besoins.

## CONFIGURATION DE JENKINS

- Personnaliser votre login/mpd administrateur
  - Attention à ne pas perdre le mdp jenkins
- Se connecter à la console Jenkins

### Créer le 1er utilisateur Administrateur

Nom d'utilisateur:	<input type="text" value="bnasslahsen"/>
Mot de passe:	<input type="password" value="*****"/>
Confirmation du mot de passe:	<input type="password" value="*****"/>
Nom complet:	<input type="text" value="nass lahsen"/>
Adresse courriel:	<input type="text" value="badr.nasslahsen@gmail.c"/>

## CONFIGURATION DES OUTILS UTILISÉS PAR JENKINS

- JDK

JDK
Name: <input type="text" value="JDK7"/> JAVA_HOME: <input type="text" value="/usr/lib/jvm/java-7-openjdk-amd64"/> <input type="checkbox"/> Install automatically <a href="#">Delete JDK</a>

Add JDK

# CONFIGURATION DES OUTILS UTILISÉS PAR JENKINS

## • MAVEN

The screenshot shows the Jenkins 'Maven' configuration page. It includes sections for 'Maven installations' (with a 'Name' field set to 'MAVEN3.3.9' and 'MAVEN\_HOME' set to '/usr/lib/maven/apache-maven-3.3.9'), 'Install automatically' (unchecked), and a 'Delete Maven' button. Below this is the 'Maven Configuration' section, which includes fields for 'Default settings provider' (set to 'Settings file in filesystem' at path '/home/cdkstudent/Documents/settings.xml') and 'Default global settings provider' (set to 'Use default maven global settings').

Formation Jenkins

37

# DÉFINITION DE LA GESTION DES SOURCES

- La gestion des sources est une pratique d'ingénierie dont l'art est de gérer les modifications d'informations
- Applicable au-delà de la gestion des versions de code d'un logiciel. Elle concerne :
  - Gestion des fichiers d'un site web
  - Gestion de l'historique de /etc pour les ingénieurs système UNIX
  - Documentations, supports de cours, articles, ...
  - Gestion des configurations

Formation Jenkins

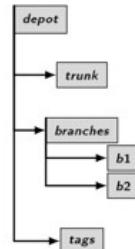
38

## PRINCIPE DE BASE DE LA GESTION DES SOURCES

- Permettre un travail collaboratif ou coopératif dans les domaines suivants :
  - développement de code source
  - ou l'écriture de documents
- Donner un droit à l'erreur en permettant de revenir à des versions antérieures
- Disposer de 2 (ou plusieurs) versions de travail (branches)
  - correction des bugs en parallèle au développement de nouvelles fonctionnalités
- Disposer d'un (ou plusieurs) dépôt(s) de référence

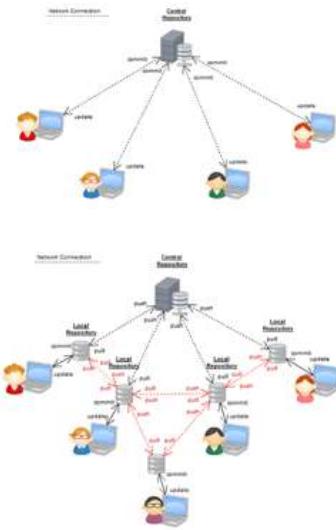
## ARCHITECTURE DE LA GESTION DES SOURCES

- Dépôt local ou distant répertoriant l'ensemble des modifications
- L'arborescence est construite sur la base de :
  - branches
  - Tags
- Les branches servent à
  - corriger un problème sur une ancienne version, développer 2 idées en parallèle, gérer sa propre
  - version du logiciel, fusionner après une divergence.
- Les tags sont des marques symboliques
- Ils permettent de définir les versions du projet
- Ils permettent de marquer l'état des artefacts du projet



# LES FAMILLES D'OUTILS DE GESTION DES SOURCES

- Version centralisée
  - Familles d'outils de gestion de versions
  - Un seul dépôt ou repository contenant toutes les versions seuls des privilégiés déclarés ont le droit de « valider » les modifications proposées:
  - Exemple : CVS, SVN
- Version décentralisée
  - Plusieurs dépôts coexistent
  - Nécessité de synchronisation
  - Nécessité pour les utilisateurs de définir des règles de fonctionnement
  - Exemple: *git, mercurial, darcs, svk*



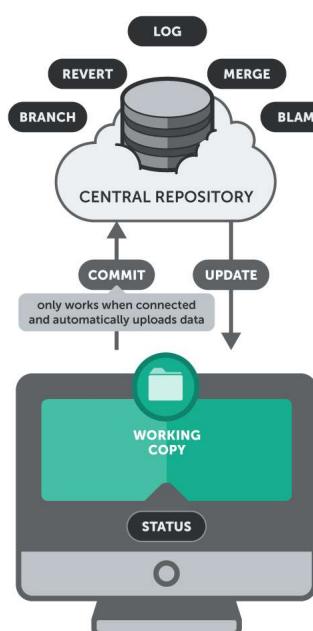
Formation Jenkins

41

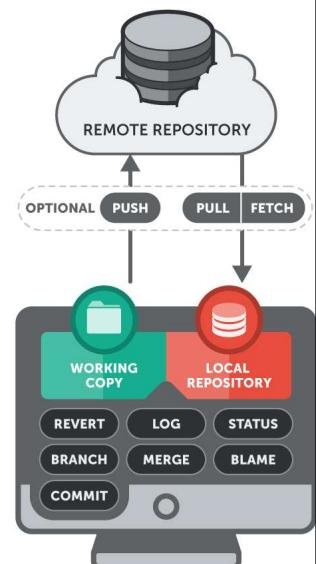
## SVN VS GIT

- Chaque développeur a l'intégralité du repo sur son poste
  - Les opérations suivantes se font hors-ligne
    - Commit
    - Show log
    - Blame
    - Switch
    - Merge
- Travailler avec des branches devient naturel
- Git est distribué
  - Votre repo peut pointer sur plusieurs serveurs (remote)
  - Exemple : Recovery en cas de panne via la mise en place d'un repo sur un Filer

### SUBVERSION



### GIT



Formation Jenkins

42

## UNE PEU DE TERMINOLOGIE...

- SVN
  - Checkout
  - Update
  - Commit
  - Switch
  - Create patch, revert (pour garder des modifs sans les commiter)
  - Create patch / apply patch



- GIT
  - Clone
  - Fetch/Pull
  - Commit + Push
  - Checkout
  - Stash
  - Cherry-pick



Formation Jenkins

43

## LES AVANTAGES DE GIT PAR RAPPORT SVN

- SVN
  - SVN est centralisé
  - Branche de travail commune
  - Et les conflits qui vont avec...
  - Process de développement non supporté

- GIT
  - Multi-branche par design
  - Supporte le process de développement
  - Des fonctions pratiques : Stash, Cherry-picking...
  - Commit local
  - Git est distribué
  - Possibilité de travail en « triangle »

Formation Jenkins

44

## ACCÈS AU GESTIONNAIRE DE CONFIGURATION DANS JENKINS

- Il existe plusieurs modes de mise à jour :
  - Une simple mise à jour (svn update)
  - Une remise à zéro complète
  - Une mise à jour après avoir supprimé les fichiers non versionnés
  - Une mise à jour avec une annulation des changements locaux (revert)
- La profondeur de mise à jour peut être définie. En général, elle est soit complète ou nulle (aucune fichier descendu en local).
- Il est possible de définir un répertoire dans lequel seront descendues les sources (ici SampleBuildProject/.). Ce répertoire est toujours relatif au répertoire workspace du job. Cette bonne pratique est très importante quand on utilise plusieurs emplacements SVN.

## CONFIGURER LE GESTIONNAIRE DE SOURCE DANS JENKINS

- Un job peut accéder à un gestionnaire de configuration.

The screenshot shows the 'Source Code Management' section of a Jenkins job configuration. Under the 'Subversion' tab, the 'Repository URL' is set to 'svn://localhost/projet1/components/SampleBuildProject/branches/\${APPLI\_V}'. A warning message indicates that the repository URL is parameterized and syntax validation is skipped. The 'Local module directory' is set to 'SampleBuildProject/'. The 'Repository depth' is set to 'infinity'. The 'Ignore externals' checkbox is checked. Below these settings, there are sections for 'Additional Credentials', 'Check-out Strategy' (set to 'Use 'svn update' as much as possible'), and 'Repository browser' (set to '(Auto)'). At the bottom right, there is a 'Advanced...' button.

## STRATÉGIES ET TECHNIQUES DE NOTIFICATION

- La notification par email est la forme de notification la plus évidente et la plus commune. Ainsi, quand les équipes mettent en place leur premier environnement d'intégration continue, c'est généralement la stratégie de notification qu'ils essaient en premier.

E-mail Notification

Recipients

teamlead@acme.com

Whitespace-separated list of recipient addresses. May reference build parameters like \$PARAM. E-mail will be sent when a build fails, becomes unstable or returns to stable.

Send e-mail for every unstable build

Send separate e-mails to individuals who broke the build

- Jenkins permet d'envoyer des notifications par d'autres canaux :
  - Mail
  - RSS
  - IRC
  - SMS
  - Bruit
  - Pda (notifio)

## JENKINS: NOTIONS DE BASE – JOB ET BUILD

- Un job se définit par un enchainement d'étapes.
- Un job travaille dans un répertoire appelé workspace.
- Un Build est une exécution d'un job.
- Un build terminé à un statut (Success, Unstable, Failed or Broken)
- Un build produit un ou plusieurs artifacts. Il peut être de nature très différente (class, jar, war, rapports de tests, zip, so, dll, tar )

## JENKINS: NOTIONS DE BASE –TYPES DE BUILDS

- Build local / privé
  - le développeur exécute un build
  - Compilation
  - Tests unitaires
- Build d'intégration
  - Le serveur d'intégration exécute un build
  - Idem build local
- Build de nuit
  - Le serveur exécute un build complet
    - Tests d'intégration
    - Documentation, rapports, métriques
    - Release / Déploiement

## JENKINS: NOTIONS DE BASE - LES TYPES DE JOB

- Il existe plusieurs types de job :
  - Freestyle project : job de base
  - Maven project : job qui intègre la phase build comme une étape maven et récupère des infos dans le pom
- D'autres types de job peuvent être ajoutés via l'installation de plugins
  - Multijob project : job qui permet de piloter d'autres jobs par groupe
  - Pipeline project : Définit d'un pipeline par du code groovy (Jenkins 2)

## JENKINS: NOTIONS DE BASE - LES ÉTAPES DE BUILD OU DE POST-BUILD

- Par défaut, un job est découpé en 4 grandes étapes
  - Installation d'outils si demandée et nécessaire (JDK, Maven, ...)
  - Récupération les éléments du référentiel de configuration (SVN, Git)
  - Exécution des étapes de build
  - Exécution des étapes de post-build
- Certaines étapes peuvent être exécutées avant la récupération des sources ou juste après.
- En cas d'erreur sur une étape de build, les autres étapes de build ne sont pas exécutées et le job exécutera les étapes de post-build si elles sont définies.
- Généralement, une étape de post-build peut s'exécuter en fonction du statut de build et ne le modifie pas.

## JENKINS: PRINCIPALES TYPES D'ÉTAPE DE BUILD

- Build Maven (Invoke top-level Maven targets) :
  - Permet l'exécution une commande maven
- Scripts (Execute Shell)
  - Permet d'exécuter des commandes shell voire un script préalablement installé à partir de la gestion de configuration
- Récupérer les artifacts d'un autre job (Copy artifacts from another project)
  - Permet de reprendre les artifacts produits précédemment.
- Injecter des variables d'environnement

## PARAMÈTRE D'UN JOB

- Un job peut avoir des paramètres. Ces paramètres doivent avoir une valeur par défaut afin de permettre le lancement automatique du job. Un job peut nécessiter des paramètres pour la définition de:
  - La branche de gestion de configuration
  - L'environnement
- Les paramètres d'un job sont typés, par exemple :
  - Paramètre String
  - Paramètre Choice
  - Paramètre Boolean
- Des types de paramètre peuvent être ajoutés par l'ajout de plugins

## MODE DE LANCEMENT D'UN JOB

- Il faut définir le mode de déclenchement du job :
  - Lancement manuel via l'interface
  - Lancement conditionnel sur l'exécution d'un autre
  - Lancement périodique
  - Lancement sur une promotion d'un autre job
  - Lancement sur modification du référentiel de configuration (commit SVN)
  - Modification d'une dépendance de type Snapshot (Job Maven)
- Les modes de lancement peuvent se cumuler.

## MODE DE LANCEMENT D'UN JOB

- L'exemple ci-dessous permet de démarrer le job tous les 5 minutes tous les jours.

The screenshot shows the Jenkins 'Build Triggers' configuration page. Under 'Build Triggers', the 'Poll SCM' option is checked. In the 'Schedule' field, the expression 'H/5 \* \* \* \*' is entered, indicating a periodic build every 5 minutes. A note below states: 'Would last have run at vendredi 29 avril 2016 09 h 00 CEST; would next run at vendredi 29 avril 2016 09 h 05 CEST.' There is also an 'Ignore post-commit hooks' checkbox.

## JENKINS - ENCHAINEMENT DES JOBS

- Mode de déclenchement d'un job :
  - Lancement manuel via l'interface
  - Lancement conditionnel sur l'exécution d'un autre
  - Lancement périodique
  - Lancement sur une promotion d'un autre job
  - Lancement sur modification du référentiel de configuration (commit SVN)
- Étapes de build ou de post-build déclenchant un job
  - Comme une étape du build : *Trigger/call builds on other projects*
    - Avec possibilité de faire du synchrone
  - Comme une étape manuelle en post-build : *Build other projects (manual step)*
    - Peut lancer manuellement plusieurs jobs avec les mêmes paramètres
  - Comme une étape automatique en post-build : *Trigger parameterized build on other projects*
    - Chaque groupe de jobs peut avoir son propre passage de paramètres
  - Comme une étape automatique sans passage de paramètre en post-build : *Build other projects*

## DISTRIBUTION DES TÂCHES SUR PLUSIEURS NŒUDS: NODES, MASTER, SLAVE ET EXÉCUTEURS

- Le master est le serveur principal de Jenkins.
  - Il héberge l'ensemble de la configuration (générale, jobs, plugins)
- Un slave est un serveur qui permet au master de déléguer du travail.
  - Un job peut être exécuté sur un slave spécifique ou pas.
  - Un slave doit être accessible par le master et dispose, au minimum, d'une JRE.
- Un exécuteur est la capacité d'exécuter un job.
  - Le nombre d'exéuteurs simultanés est défini par node (master et slave)
  - Un exécuteur ne peut prendre en charge l'exécution que d'un job à la fois.
- L'utilisation de slaves permet de :
  - répartir la charge (et donc préserver le master)
  - avoir des machines spécifiques (non mutualisées et avec un socle logiciel donné)
  - déclarer des slaves sur d'autres infrastructure (VPN)

## MAVEN - PRÉSENTATION

- L'outil Apache Maven fonctionne sur la base de lifecycle, phase et goals.
  - Lifecycle définit un enchainement de phases. Il existe 3 lifecycles par défaut (default, clean, site)
  - Une phase est liée à un lifecycle. Toutes les phases qui la précèdent sont exécutées.
  - Un goal est une tâche spécifique. Il peut être lié à une phase ou invoqué directement . Il peut être propre à un plugin.
- Certains plugins imposent l'exécution de phases avant leur exécution ou ne s'exécutent que dans une phase donnée. Il peut être possible de demander l'exécution du plugin lors d'une phase.
- La configuration Maven se fait dans un fichier pom.xml. Il permet de gérer les outils de construction (définition des plugins) et les dépendances logicielles (avec la notion de scope).
- Maven propose une gestion fine des dépendances des bibliothèques à travers un dépôt central qui référence la plupart des bibliothèques et frameworks Java
- Le dépôt est accessible depuis internet
  - [mvnrepository.apache.org](http://mvnrepository.apache.org)

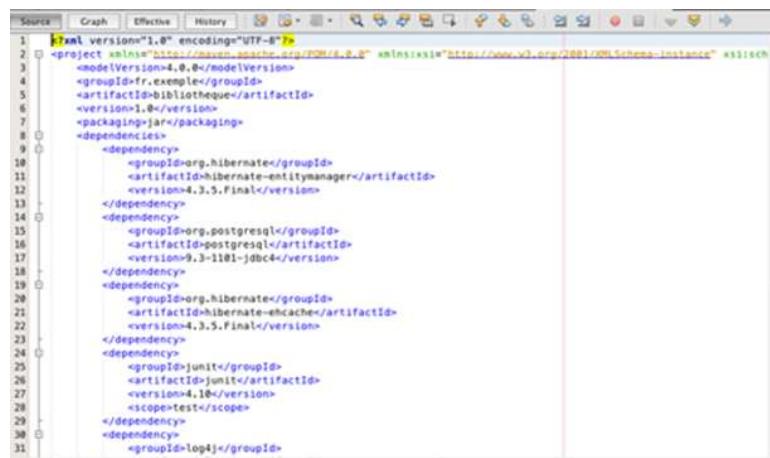
## MAVEN - PRESENTATION

- La notion de build
  - maven propose au développeur de définir lui-même à quoi correspond un 'build'.
  - compilation bien sûr, mais aussi lancement des tests unitaires examen de la couverture de test inspection du code
  - génération de la javadoc
  - création d'un livrable (jar, war, ear) Etc...
- Chaque tâche est exécutée via un plugin déclaré dans un fichier xml appelé pom.xml

## MAVEN - PRESENTATION

- La démarche est simple : le développeur se contente d'indiquer les dépendances directes dans une version donnée
  - junit 4.4 par exemple
- Maven prend en charge la récupération des dépendances indirectes

## MAVEN - LA GESTION DES DÉPENDANCE



```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>
    <groupId>fr.example</groupId>
    <artifactId>bibliothèque</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-entitymanager</artifactId>
            <version>4.3.5.Final</version>
        </dependency>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>9.3-1101-jdbc4</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-ehcache</artifactId>
            <version>4.3.5.Final</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
```

## MAVEN - PRESENTATION

- Maven propose d'avoir 4 sources folders :
  - src/main/java : classes java de l'application
  - src/test/java : classes java des tests unitaires
  - src/main/resources : fichiers xml, properties, etc... nécessaires pour l'exécution (donc à mettre dans le classpath)
  - src/test/resources : fichiers xml, properties, etc... nécessaires pour l'exécution des tests unitaires (donc à mettre dans le classpath)

## MAVEN - PRESENTATION

- Utiliser Maven consiste à
  - Créez un projet de type maven
  - Renseignez les dépendances dans le fichier pom.xml à la racine du projet.
  - Utiliser un plugin maven si ce n'est pas le cas déjà (Q4E, MIA, m2Eclipse)
- Il est possible d'enrichir le cycle de projet par défaut en rajoutant d'autres plugins
  - génération du code
  - analyse du code
  - génération des rapports
  - génération de la documentation

## MAVEN - PRESENTATION

- Les propriétés dans maven permettent de définir des valeurs sur l'ensemble du fichier. Il y a des propriétés implicites liées à la définition du projet:
  - artifactId : Nom de l'artifact produit (ex. SampleBuildProject)
  - packaging : Type d'artifact produit (ex. war)
  - version : Version de l'artifact (ex. 1.0.0)
  - build.finalName : Nom du build produit (ex. SampleBuildProject)
  - un ensemble de dépendances
  - les plugins

## MAVEN - PRESENTATION

- L'ensemble des dépendances récupérées par maven sont stockées automatiquement dans le répertoire :
  - c:\documents and settings\.m2\repository
- Le résultat du build est stocké dans le répertoire target.

## MAVEN - PRESENTATION

- Il existe quelques plugins importants
  - Plugin de ressource: maven-resources-plugin
  - Plugin de compilation: maven-compiler-plugin
  - Plugin de tests: maven-surefire-plugin
  - Plugin PMD d'analyse de code: maven-pmd-plugin
  - Plugin de packaging:
    - maven-war-plugin
    - maven-jar-plugin
    - maven-ear-plugin

## JENKINS: BONNES PRATIQUES

- Définir des règles de nommage pour les jobs et les paramètres de ces jobs
- Toujours passer les paramètres du job courant au job suivant
- Rester simple pour être compréhensible, utilisé et maintenu !
- Sauvegarder la configuration des jobs Jenkins
  - Tous les outils de construction doivent être en gestion de configuration
  - La configuration de Jenkins en fait partie
  - Ils contiennent la connaissance permettant de construire la solution
- Toujours rendre nécessaire l'authentification des utilisateurs pour déclencher une action (promotion, exécution de job, ...)
  - Pas pour surveiller qui fait une action
  - Pour comprendre pourquoi cette action a été faite (il faut donc savoir à qui poser la question)

## JENKINS: BONNES PRATIQUES

- Ne pas prévoir de mode dégradé.
  - Le paramétrage répond au processus de l'équipe
  - Les règles du projet doivent être respectées par tout le monde
  - En cas de force majeure, lancer une commande ou modifier un fichier de configuration pourra toujours être fait en conscience
  - La documentation du pipeline est beaucoup plus précieuse

## JENKINS: BONNES PRATIQUES

- C'est-à-dire qu'on doit savoir avec certitude la traçabilité de:
  - Ce que l'on construit
  - Ce que l'on teste
  - Ce que l'on analyse
  - Ce que l'on package
  - Ce que l'on tague
  - Ce que l'on déploie
  - Ce que l'on qualifie
  - Ce que l'on livre
- Bonnes pratiques
  - Contrôler les enchainements (il faut avoir fait l'étape 1 avant de faire la 2)
  - Taguer la gestion de configuration avec le numéro du build
  - Intégrer les métadonnées du build dans la solution pour assurer la traçabilité de la solution et la gestion de configuration
  - Assister la validation de la livraison en fournissant des éléments de contrôles

## INTÉGRATION CONTINUE - QUIZZ



- L'intégration continue:
  - N'est utile que pour les grands projets
  - Ne doit être exécutée qu'une fois par jour
  - Se lance à intervalles réguliers pour intégrer les nouvelles contributions
  - Sa mise en place est compliquée
  - Est utilisée par les développeurs
  - Est définie que par l'intégrateur
  - Son usage est interdit aux fonctionnels
- Les jobs de l'intégration continue:
  - Ne sont lancés que manuellement
  - Sont indépendants les uns des autres
  - Ne font que du build
  - Sont chainés en suivant les processus définis sur le projet.

## JENKINS - QUIZZ



- Jenkins signale les erreurs:
  - En envoyant un mail
  - En envoyant un SMS
  - Dans les logs de build
  - Ne signale pas les erreurs ; c'est aux développeurs d'aller voir

## GESTION DE CONFIGURATION - QUIZZ



- La Gestion de configuration :
  - Permet de reconstruire la solution à tout moment de son évolution
  - Ne s'applique qu'au code et pas au binaire
  - Ne s'utilise pas avec les fichiers de propriétés
  - Permet de sauvegarder son travail



## QUALITÉ DU CODE

Formation Jenkins

73

## POURQUOI AVOIR UNE ANALYSE DE CODE ?

- Pourquoi une analyse de code ?
  - Pour garantir la qualité par rapport à un niveau attendu
  - Pour estimer la dette technique de la solution
  - Pour respecter les engagements contractuels (exigences spécifiques)
  - Pour faire grandir les compétences des développeurs
- La qualité logicielle aura un lien direct sur la maintenabilité :
  - Stabilité de la solution
  - Coûts des évolutions ou des corrections

Formation Jenkins

74

## QUALIMÉTRIE - NOTION DE DETTE TECHNIQUE

- Définition de la dette technique
  - Le terme de dette technique provient initialement de la logique d'intérêts que l'on retrouve dans le calcul d'une dette financière : il s'agit de son application dans la vie d'un projet de développement logiciel. L'analogie illustre la notion d'intérêts : s'ils ne sont pas remboursés rapidement, le coût de la dette augmente jusqu'à un point où il n'est plus possible de la rembourser intégralement, et où tout paiement ne sert qu'à rembourser ces intérêts
- La dette technique est nécessairement relative.
  - Elle est liée à un choix d'architecture et de règles de conception.
  - Il n'y a pas de vérité absolue

## OUTILS D'ANALYSE (CHECKSTYLE, FINDBUGS, CPD/PMD)

- L'analyse de code peut être très large et variée
- L'outil central reste SonarQube qui permet de définir ses propres règles et d'agrégner des analyses faites en amont:
  - Logiciel libre. Certainement le plus connu et utilisé. Permet d'analyser le code de plus de 26 langages, même ceux sous licence commerciales.
- D'autres outils d'analyse peuvent être utilisés :
  - Analyse statique de code avec des plugin Maven (pmd, checkstyle)
  - Analyse de couverture de test (jacoco, cobertura)
  - Analyse de sécurité (Owasp dependencies check)
  - Analyse plus complète avec l'outil CAST



## BONNES PRATIQUES – ANALAYSE DE QUALITE DE CODE

- Elle adresse :
  - Le respect des normes de développement
  - La mesure de la qualité logicielle par rapport à des critères de contrôle
- Attention faux positifs et la configuration de vos plugins
  - Certaines violations peuvent être admissibles sur le projet.
  - La configuration des postes doit être commune et en phase avec la configuration de l'analyse de l'intégration continue.
- La mesure de la dette technique ne sert à rien si elle n'est pas gérée avec des plans d'actions adaptés.
- Le niveau de qualité ne doit pas être un frein à la mise en place d'une analyse de code :
  - « J'ai plus de 3 000 violations ; cela ne sert à rien »

## QUALIMÉTRIE - ANALYSE DE CODE - QUIZZ



- L'analyse de code révèle:
  - Le manque des Tests Unitaires
  - Le manque de Tests d'Intégration
  - Les exceptions mal catchées
  - Les failles de sécurité
  - Le non-respect des standards
  - Doit être faite à chaque commit
  - Doit être faite à chaque livraison
  - Est une mesure objective de la dette technique



## AUTOMATISATION DES TESTS

Formation Jenkins

79

## INTRODUCTION AUX TESTS

- L'impasse sur les tests (unitaires, d'intégration...) n'est **jamais gagnante**
- Les coûts de corrections après livraison **augmentent significativement** les coûts de la qualification... et les délais
- Les tests réduisent les « surcoûts »



Formation Jenkins

80

## TYPOLOGIE DES TESTS

- Tests Unitaires
  - Test d'un élément logiciel
  - Tests fonctionnels basés sur des spécifications détaillées
  - Tests non fonctionnels (exemples):
    - Normes de codage
    - Tests de branches
    - Performances
    - Robustesse
    - ...
  - Réalisés par les Ingénieurs de développement

- Tests d'Intégration
  - Test des interfaces et des interactions entre les composants
  - Test des interactions avec les autres systèmes ou sous-systèmes
  - Tests fonctionnels basés sur les spécifications
  - Tests non fonctionnels (exemples):
    - Performances
    - Charge
    - Stress
    - ...
  - Tests sous la responsabilité de l'analyste pouvant impliquer les développeurs

Formation Jenkins

81

## TYPOLOGIE DES TESTS

- Tests Système
  - Tester que les composants de la solution respectent les exigences spécifiées qui les concernent
  - Démontrer que la solution satisfait à l'utilisation prévue lorsqu'elle est placée dans l'environnement cible (*env. proche de l'exploitation*)
  - Tests fonctionnels basés sur les cas d'utilisation
  - Tests non fonctionnels basés sur les exigences non fonctionnelles
  - Réalisés par l'Equipe de test **indépendante** de l'équipe de réalisation

- Tests d'Acceptation - Acceptance Testing
  - Déterminer que la solution est prête à être déployée
  - Tests fonctionnels et non fonctionnels
  - Réalisés par **l'Equipe de test client**

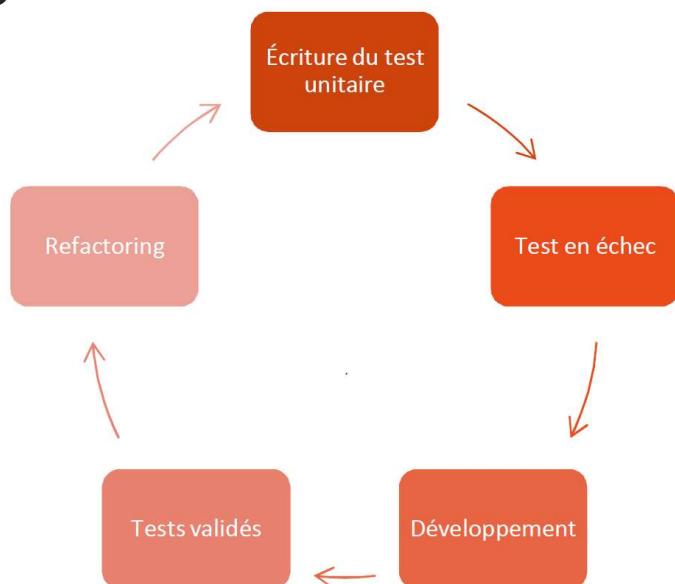
Formation Jenkins

82

## TESTS – DEMARCHE - TDD

- Du Test Driven Development

- Compréhension du besoin
- Écriture d'un test unitaire
- Lancement du test qui est en échec.
- Implémentation de la fonctionnalité pour que le test passe
- Réalisé par le développeur



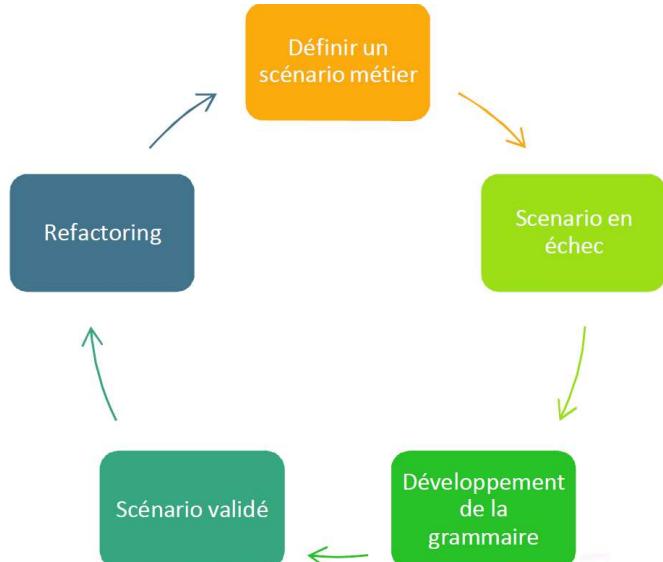
Formation Jenkins

83

## TESTS – DEMARCHE - BDD

- Au Behavior Driven Development

- Concevoir les tests à partir d'un comportement ou d'une fonctionnalité.
- Écrire en français le test par rapport à un besoin (et non la solution) à partir d'une grammaire définie
- Implémentation du code
- Implique la collaboration entre les fonctionnels et les techniciens



Formation Jenkins

84

## AUTOMATISER LES TESTS

- L'objectif est de trouver les défauts d'implémentation
  - Vérifier que le **comportement** des composants développés est celui attendu
  - Vérifier la **robustesse** des composants développés face à des conditions aux limites
- Un test de développement est un test unitaire ou d'intégration qui nécessite **ni de déploiement** de l'application **ni de donnée extérieure**
- Les tests de développement automatisés :
  - Aident au développement
  - Documentent les composants
  - Contrôlent des éventuelles régressions
  - Sécurisent le développement
  - Bonifient l'intégration continue

## AUTOMATISER LES TESTS

- Principaux framework de tests
  - JUnit, TestNG, nUnit
- Mesure de la couverture
  - Cobertura, Jacoco
  - Rapport intégrable dans le tableau de bord SonarQube.
- Dans un contexte TMA,
  - Ne pas chercher à créer des tests sur pour couvrir tout le périmètre dès le début
  - Ne pas se focaliser sur les tests « unitaires »
  - Ne pas hésiter supprimer des tests caduques
  - La présence de tests est contrôlée lors de la revue de code

## TYPES DE TESTS DE PERFORMANCE

- Tir dans les conditions **nominales**

- Usage au plus proche de la production
- Tir sur environ 1h + montée et descente de 20%
- Observation des temps de réponses, influence de la charge, des éléments systèmes

- Tir dans les condition de **stress**

- Généralement identique aux conditions nominales, sauf
  - La charge qui peut être plus élevées (x3 ou x4)
  - L'espace disque est volontairement saturé
  - Les débits réseaux sont volontairement ralenties
  - Permet de connaître les limites du système

- Tir d'**endurance**

- La durée est nettement plus élevée
- L'objectif est de faire apparaître les défaillances qui ne se manifestent qu'après plusieurs heures d'usages
- On réduit le nombre d'éléments collectés pour limiter la taille de la collecte



## AUTOMATISATION DES TESTS DE PERFORMANCE AVEC JMETER

- Plus un problème de performance est détecté et corrigé tard, plus son coût a des chances d'être élevés.
- C'est pour cela qu'il est conseillé de faire des tests de charge le plus tôt possible en ayant quand même une application mature et un environnement représentatif en terme de données.
- JMeter est un outil open source populaire, qui permet de réaliser des tests de performance.
- Jenkins s'intègre avec l'outil JMeter grâce aux plugins Jenkins Performance Plugin et JMeter Maven Plugin.

## ETAPES D'INTEGRATION ENTRE JENKINS ET JMETER

- Configurer jmeter sur votre projet à tester (Par exemple avec Maven)
- Installer le plugin perfomance plugin dans Jenkins

<input type="checkbox"/>	Archive and publish .NET code coverage HTML reports from <a href="#">NCover</a> .	0.3
<input type="checkbox"/>	<b>NUnit Plugin</b> This plugin allows you to publish <a href="#">NUnit</a> test results.	0.10
<input checked="" type="checkbox"/>	<b>Performance Plugin</b> This plugin allows you to capture reports from <a href="#">JMeter</a> and <a href="#">JUnit</a> . Hudson will generate graphic charts with the trend report of performance and robustness. It includes the feature of setting the final build status as good, unstable or failed, based on the reported error percentage.	1.2
<input type="checkbox"/>	PerfPublisher Plugin This plugin generates global and trend reports for tests results analysis. Based on an open XML tests results format, the plugin parses the generated files and publish statistics, reports and analysis on the current health of the project.	7.97

## ETAPES D'INTEGRATION ENTRE JENKINS ET JMETER

- Mettre en place une tâche de build de performance dans Jenkins

**Build**

Root POM	<input type="text" value="pom.xml"/> 
Goals and options	<input type="text" value="clean verify -Pperformance"/> 
MAVEN_OPTS	<input type="text" value="-Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m"/>  

- Configurer le plugin Performance dans votre tâche de build

Publish Performance test result report 

Performance report  

Specify the path to the Performance report files, relative to the [workspace root](#).  
- If you left this field blank the plugin will look for files matching the pattern: \*\*/\*.jtl in the workspace.  
- Or you can enclose the search specifying a list of files and folders separated by semicolon.  
- Or use an Ant Fileset pattern.

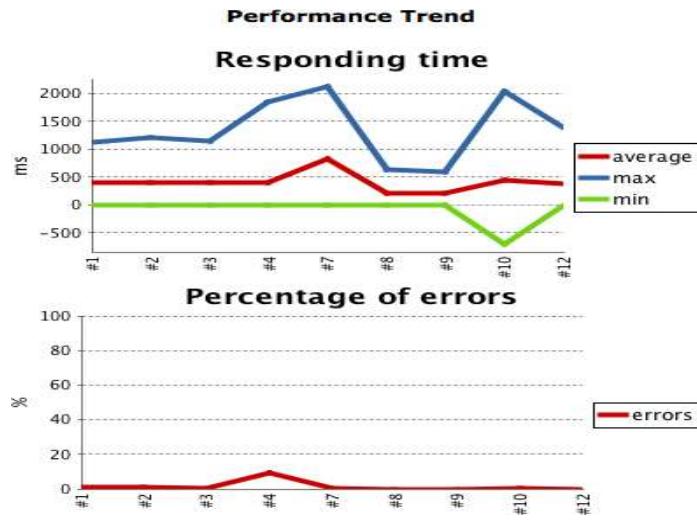
Performance threshold  Unstable  Failed

Thresholds:  %  %

Specify the error percentage threshold that set the build unstable or failed (a value of 0 means: dont use this threshold).

## ETAPES D'INTEGRATION ENTRE JENKINS ET JMETER

- Accéder aux rapports de performance sur Jenkins



Formation Jenkins

91

## TESTS DE NON RÉGRESSION

- 60 à 80% du code d'une application est produit lors des maintenances évolutives ou correctives
- Il en découle qu'un ratio tout aussi important d'anomalies sera produit après la première mise en production
- Ils ont pour objectif de
  - s'assurer de la validité des fonctionnalités préexistantes préalablement dans le logiciel.
  - Les modifications apportées à l'existant n'ont pas générées des bugs
- La démarche consiste à rejouer la totalité des tests

Formation Jenkins

92

## MESURE DE LA COUVERTURE DE TEST

- Une métrique très utile liée aux tests est la couverture de code.
- La couverture de code donne une indication sur les parties de l'application qui ont été exécutées pendant les tests. Alors que ce n'est pas en soit une indication suffisante sur la qualité du test
- C'est une bonne indication du code qui *n'a pas* été testé.
- L'analyse de la couverture de code est un traitement consommateur en CPU et mémoire, et va ralentir votre tâche de build de façon significative.
- Pour cette raison, il faut généralement exécuter les métriques de couverture de code dans une tâche de build Jenkins séparée, exécutée après que les tests unitaires et d'intégration aient réussi.

## OPTIMISER LES TEMPS D'EXÉCUTION DES TESTS

- La récupération des métriques de chaque étape de la pipeline, permet d'identifier les étapes à améliorer
- Programmer les tests coûteux le soir (Performance / BDD)

**Build Triggers**

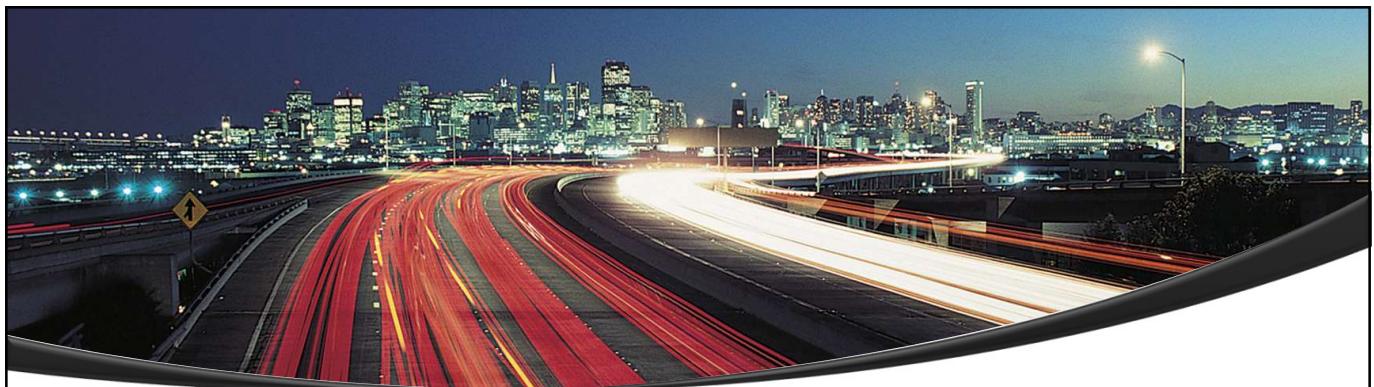
<input type="checkbox"/> Build whenever a SNAPSHOT dependency is built	?
<input type="checkbox"/> Build after other projects are built	?
<input checked="" type="checkbox"/> Build periodically	?
Schedule	@midnight
<input type="checkbox"/> Poll SCM	?

## TESTS - QUIZZ



- Les tests unitaires, à la différence des tests d'intégration : :

- Permettent de valider l'algorithme
- Permettent de valider le déploiement
- N'ont pas besoin de jeux de données
- Doivent être nécessairement bouchonnés
- Les tests unitaires sont exécutés par l'intégration continue
- Des tests obligatoirement développés en Junit
- Dépendent d'un déploiement de la solution



## AUTOMATISATION DU DÉPLOIEMENT

## CRITERES D'UN LIVRABLE

- Un livrable de qualité est :
  - Autosuffisant
    - Il contient tout le nécessaire pour être installé
  - Agnostique de l'environnement
    - Le même livrable peut être installé sur différents environnements
  - Robuste
    - En cas d'erreur, il doit être facile de revenir en arrière (rollback)

## DES LIVRABLES DE PLUS EN PLUS COMPLETS

- Historiquement, les livrables étaient constitués uniquement de l'application compilé (ou des sources exécutables)
- Avec l'**Infrastructure as Code**, un livrable contient ses propres scripts de déploiement.
- Si le déploiement change, le livreur n'a rien à changer.
- Certains vont plus loin en proposant des **infrastructures immuables**. Le livrable est une image de VM contenant l'OS, les librairies, le runtime, l'application.

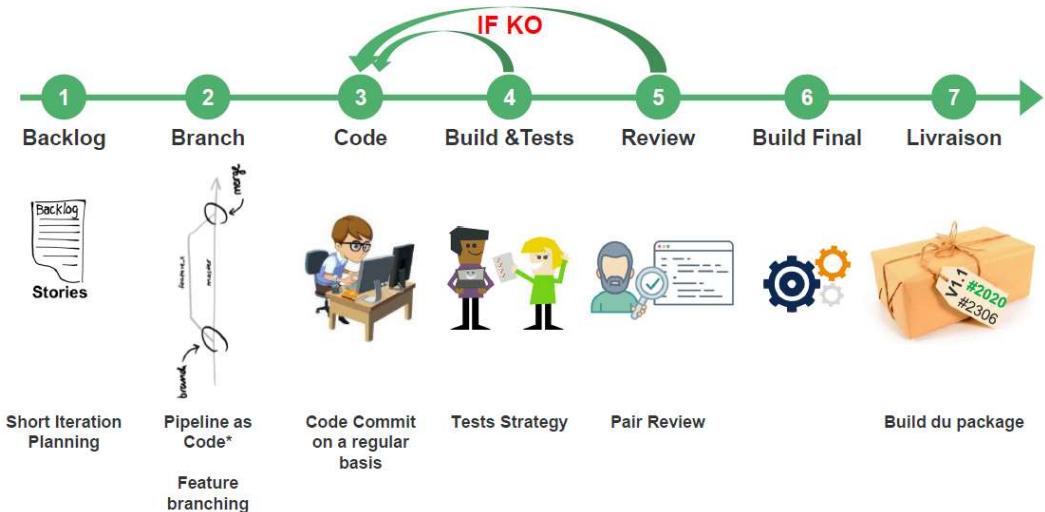
## INDUSTRIALISER LES LIVRABLES

- Un livrable doit être construit... :
  - Automatiquement
    - La construction d'un livrable ne doit pas nécessiter d'actions manuelles
  - À partir d'un checkout de la base de code
    - Afin de ne pas avoir de conflits avec des fichiers créés lors de build précédents

## POURQUOI AVOIR UN PACKAGING AUTOMATISÉ ?

- Pour construire le package de la solution en iso client
- Pour construire de façon répétitive et donc fiable
- Pour ne pas dépendre d'une personne
- Pour permettre un déploiement automatisé et donc simple et rapide

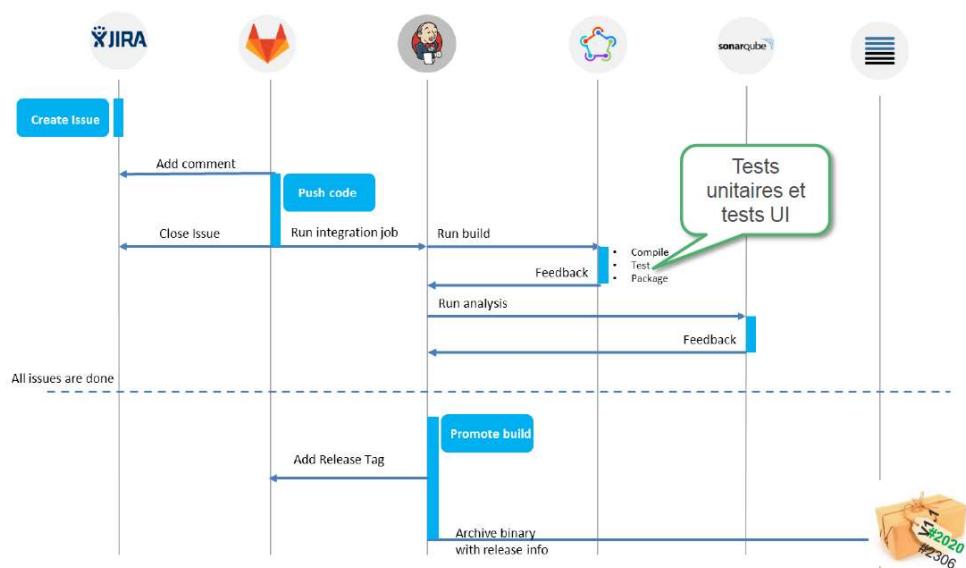
## PROCESSUS DE « RELEASE »



Formation Jenkins

101

## PROCESSUS DE « RELEASE »



Formation Jenkins

102

## LE GESTIONNAIRE DE DÉPÔTS

- Un gestionnaire de dépôts permet de stocker les dépendances d'une application, mais aussi une application en elle-même.
- Lors du développement, les développeurs peuvent utiliser des dépendances se trouvant sur Internet, pour utiliser des bibliothèques logicielles qui de base ne sont pas disponible.
- Une fois, ces bibliothèques téléchargées et « installées » dans l'application, celle-ci y sont directement embarqué.
- Pour gérer ces bibliothèques privées, nous utilisons donc des gestionnaires de dépôts.
- Ce même gestionnaire de dépôts pourra par la suite héberger notre application « finale », la versionner et la mettre à disposition à ceux qui auront le droit d'y accéder.
- L'atout principal du gestionnaire est le versionning et la mise à disposition de l'application à chaque monté de version.

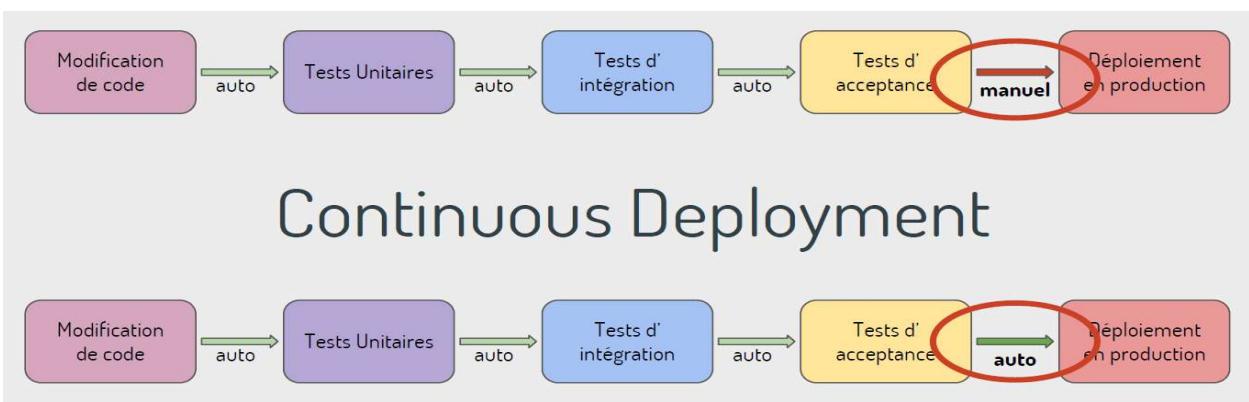
## EXEMPLE DE GESTIONNAIRE DE DÉPÔTS

Nom	Description	Logo
Sonatype Nexus	Gestionnaire de paquet qui propose de stocker des modules de différent langage. Il est très utilisé notamment pour les applications JAVA.	 Sonatype
Nuget	Gestionnaire de package développé par Microsoft	
NPM	« Node Package Manager » est le gestionnaire de paquets officiel de NodeJs (JavaScript)	
DockerHub	Gestionnaire de package Docker	

## LE DÉPLOIEMENT CONTINU

- Le **Continuous Deployment** va plus loin que le Continuous Delivery.
- Chaque modification de code, si elle passe les tests et validations, va aboutir à une **mise en production**, de manière automatique
- Le déploiement continu, est une pratique qui vise à réduire le plus possible le temps de cycle, le temps écoulé entre l'écriture d'une nouvelle ligne de code et l'utilisation réelle de ce même code par les utilisateurs finaux.
- Le déploiement continu permet depuis une livraison d'installer l'application en production. De même que pour la livraison continue, il doit être possible via un simple bouton de pouvoir revenir en arrière et donc d'avoir une application fonctionnelle sur l'environnement de production chez le client.

## LE DÉPLOIEMENT CONTINU



## MISE À JOUR DES BASES DE DONNÉES

- Un gestionnaire de base de données automatise la sauvegarde et l'exploitation des données d'une application. Elle permet de mettre à jour ou d'enrichir la base sans passer par l'application.
- Le gestionnaire permet également de faire des rollback aisément.
- L'avantage de ce gestionnaire est qu'il garantit l'intégrité d'une base de données, tout en donnant la possibilité d'y apporter des modifications via de simples scripts procédures.
- Exemple de gestionnaire de base de données :
  - Rollback : retour arrière

## EXEMPLE DE GESTIONNAIRE DE BASE DE DONNÉES :

Nom	Description	Logo
Liquibase	Sous licence Apache 2.0, gère les montées de versions, les différences entre deux schémas SQL et autres.	
Flyway	Flyway est assez équivalent à Liquibase, il est toutefois plus simple d'utilisation et possède moins de configurations ce qui peut être également un moins comparé à Liquibase	
Phinx	Identique à Flyway mais pour le langage PHP	

## JENKINS: LE PIPELINE AS CODE

- Principes
  - La définition du pipeline est exprimée sous forme d'un script Groovy
  - Ce script peut (doit) être en gestion de configuration (jenkinsfile)
  - Il permet de définir l'ensemble des étapes d'un pipeline de façon très centralisée.
- Avantages
  - La gestion de configuration de notre pipeline se simplifie.
    - L'applicatif porte ses règles de construction
    - On peut avoir plusieurs versions de pipeline en parallèle en fonction de l'évolution de la solution
    - On suit très facilement les modifications
  - Les enchainements ne sont pas portés par les jobs .
  - On limite de façon très forte le nombre de jobs
  - Il permet de travailler sur plusieurs branches en parallèle d'une même livraison avant même d'avoir faire la fusion via le job multi-branches.

## JENKINS: PIPELINE AS CODE - LE JENKINSFILE

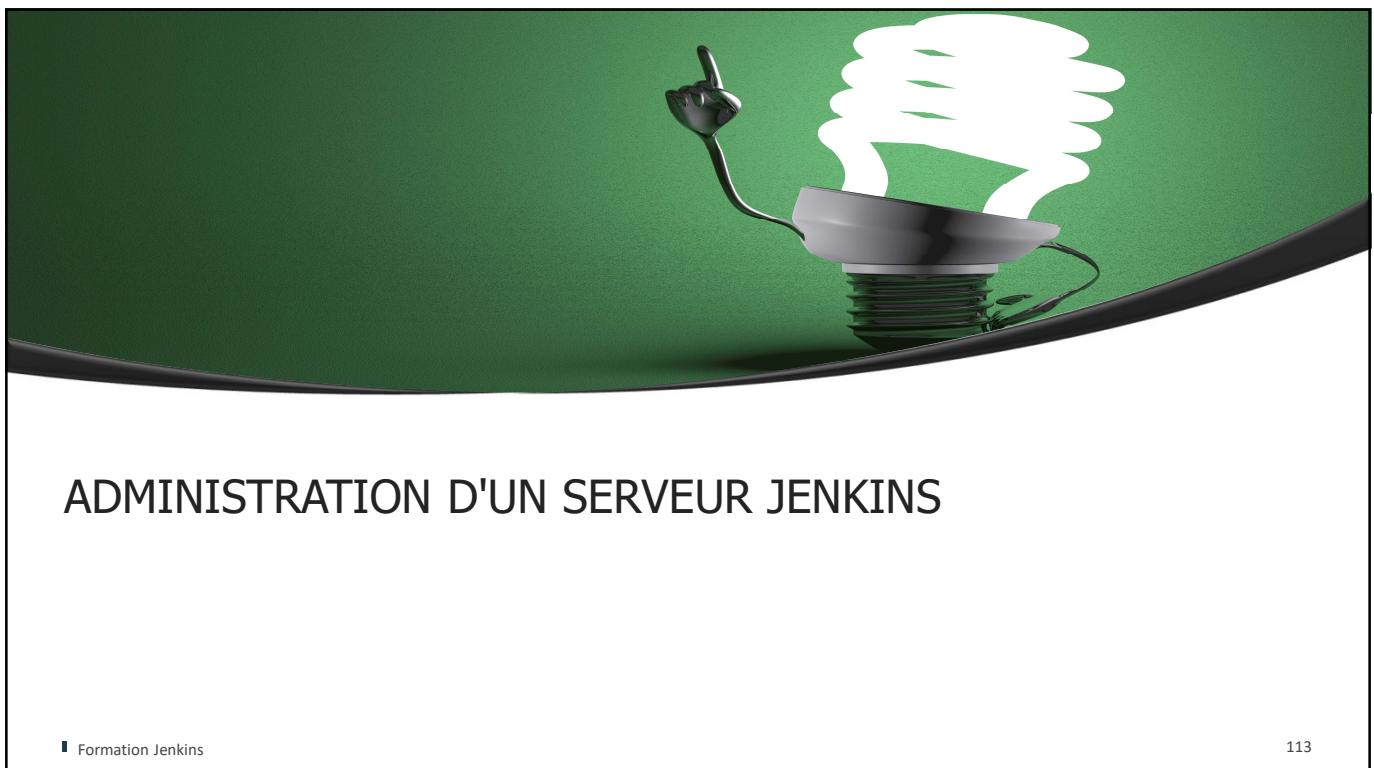
- Le script Jenkinsfile
  - se base sur le langage Groovy (aucune connaissance nécessaire)
    - qui apporte sa syntaxe et ses mots-clefs (if, for, ...)
    - Les parenthèses pour les paramètres d'appel et les ';' ne sont pas obligatoires est un simple fichier texte utilisant des fonctions propres à Jenkins
    - dispose de la puissance d'un langage évolué (variables, gestion des erreurs)
- Les fonctions d'un Jenkinsfile (1/2) :
  - `stage(nomEtape) {...}` marque le début d'une étape du pipeline
  - `node(labelsOuNom) {...}` est un bloc d'instructions qui s'exécutera sur un nœud (master ou un slave Jenkins répondant aux labels fournis ou au nom fourni)
  - `try {...} catch(err) {...} finally {...}` est une structure pour traiter manuellement les erreurs
  - `withEnv(...) {...}` est une structure déclarant des variables valables dans le bloc d'exécutions
  - `withCredentials(...) {...}` permet d'utiliser des mots de passe dans les exécutées dans le bloc

## JENKINS: PIPELINE AS CODE - LE JENKINSFILE

- Les fonctions d'un Jenkinsfile (2/2) :
  - `dir(nomRépertoire) {...}` exécute les commandes dans le répertoire
  - `writeFile` permet d'écrire un fichier
  - `sh` et `bat` permettent d'exécuter n'importe quelle commande
  - `git` et `svn` pour manipuler les repository (clone, checkout, commit, push, ...)
  - `input` permet de demander une saisie ou un choix manuel
  - `step(...)` pour exécuter une classe (pour les plugins)
  - `parallel(...)` pour exécuter des commandes en parallèle
  - `build()...` pour exécuter un autre job
- La référence est là : <https://jenkins.io/doc/pipeline/steps/>

## JENKINS: PIPELINE AS CODE - CONCLUSION

- Le pipeline as code est un vrai gain et apporte de vraies solutions.
  - Gestion de configuration du pipeline
  - Gestion des branches parallèles
- En revanche,
  - Le jenkinsfile presuppose l'installation des plugins invoqués.
  - La définition du pipeline ne se fait plus par petites boîtes mais par du code.
    - Cela implique normes de développement et relecture
  - Un pipeline as code est forcément linéaire.
    - Une étape n'a forcément qu'un successeur



## ADMINISTRATION D'UN SERVEUR JENKINS

Formation Jenkins

113

## SECURISER JENKINS

- Il est recommandé de sécuriser Jenkins, en passant par les étapes suivantes:
  - Gérer Jenkins, puis configurer la sécurité globale.
  - Sélectionnez l'indicateur Activer la sécurité.
  - La façon la plus simple est d'utiliser la base de données personnelle de Jenkins.
  - Créez au moins l'utilisateur "Anonyme" avec un accès en lecture.
  - Créez également des entrées pour les utilisateurs que vous souhaitez ajouter à l'étape suivante.

Formation Jenkins

114

# ACTIVATION DE L'AUTHENTIFICATION

 **Configure Global Security**

Enable security  
TCP port for JNLP slave agents  Fixed :   Random  Disable

Markup Formatter  Raw HTML  
Treat the text as HTML and use it as is without any translation  
 Disable syntax highlighting

Access Control

**Security Realm**  
 Delegate to servlet container  
 Jenkins's own user database  
 Allow users to sign up  
 LDAP  
 Unix user/group database

**Authorization**  
 Anyone can do anything  
 Legacy mode  
 Logged-in users can do anything  
 Matrix-based security

User/group	Overall	Slave	Job	Run	View	SCM
Anonymous	<input type="checkbox"/>					
vogella	<input checked="" type="checkbox"/>					

User/group to add:

Project-based Matrix Authorization Strategy

Prevent Cross Site Request Forgery exploits

Formation Jenkins

115

# ACTIVATION DE L'AUTHENTIFICATION

User:   
Password:   
 Remember me on this computer  
  
  
[Create an account](#) if you are not a member yet.

## Sign up

Username:   
Password:   
Confirm password:   
Full name:   
E-mail address:

Formation Jenkins

116

## GESTION DES PERMISSIONS ET DROITS D'ACCÈS

- 3 principaux mode de sécurité:
  - Sécurité basée sur le projet: Habilitations par projets
  - Sécurité basée sur les rôles: Habilitations par rôles
  - Sécurité basée sur matrice

## JOURNALISATION DES ACTIONS UTILISATEUR

- En plus de configurer les comptes utilisateurs et leurs droits d'accès, il peut être utile de garder des actions de chaque utilisateur : en d'autres termes, qui a fait quoi à votre configuration serveur. Ce type de traçage est même requis dans certaines organisations
- Il y a deux plugins Jenkins qui peuvent vous aider à accomplir cela:
  - Le plugin Audit Trail conserve un enregistrement des changements utilisateur dans un fichier de log spécial.
  - Le plugin JobConfigHistory vous permet de garder des copies de versions précédentes des diverses configurations de tâches et du système que Jenkins utilise.

## LE PLUGIN AUDIT TRAIL

- La configuration de l'audit trail s'effectue dans la section Audit Trail de l'écran de configuration principal de Jenkins
- Le champ le plus important est l'emplacement des logs, qui indique où se trouve le répertoire dans lequel les logs doivent être écrits.
- L'audit trail est conçu pour produire des logs de style système, qui sont souvent placés dans un répertoire système comme /var/log.
- Vous pouvez aussi configurer le nombre de fichiers de logs à maintenir, et la taille maximale (approximative) de chaque fichier..

Audit Trail	
Log Location	/var/log/hudson-audit-trail.log
Log File Size MB	1
Log File Count	10
URL Patterns to Log	.*/(?:configSubmit doDelete postBuildResult cancelQueue stop toggleLogKeep doWipeOutW
Log how each build is triggered	<input checked="" type="checkbox"/>

Formation Jenkins

119

## LE PLUGIN JOBCONFIGHISTORY

- Le plugin JobConfigHistory est un outil puissant vous permettant de conserver l'historique complet des changements faits à la fois sur les tâches et fichiers de configuration système.
- Vous l'installez depuis le gestionnaire de plugin de la façon habituelle.

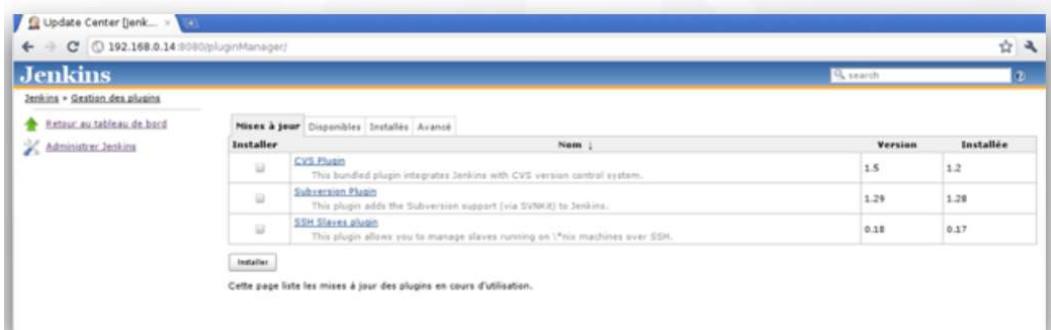
Job Config History	
Root history folder	config-history
Max number of history entries to keep	100
Save system configuration changes	<input checked="" type="checkbox"/>
System configuration exclude file pattern	queue nodeMonitors UpdateCenter
Do not save duplicate history	<input checked="" type="checkbox"/>

Formation Jenkins

120

## GESTION DES PLUGINS

- Jenkins peut être étendu via des plug-ins supplémentaires avec plus de fonctionnalités. Vous pouvez configurer vos plug-ins via le lien Manage Jenkins → Manager Plugins.
- Pour installer des plugins dans Jenkins, sélectionnez le lien Manage Jenkins → Manager Plugins et recherchez le plugin que vous souhaitez installer. Sélectionnez-le dans la liste et sélectionnez-le pour l'installer et redémarrer Jenkins.



The screenshot shows the Jenkins Update Center interface. At the top, there's a header with the Jenkins logo and the URL '192.168.0.14:8080/pluginManager/'. Below the header, the title 'Jenkins - Gestion des plugins' is displayed. Underneath, there are tabs for 'Mises à jour', 'Disponibles', 'Installées', and 'Avancé'. The 'Installées' tab is selected, showing a table of installed plugins:

Nom	Version	Installée
CVS Plugin	1.5	1.2
Subversion Plugin	1.29	1.28
SSH Slaves plugin	0.18	0.17

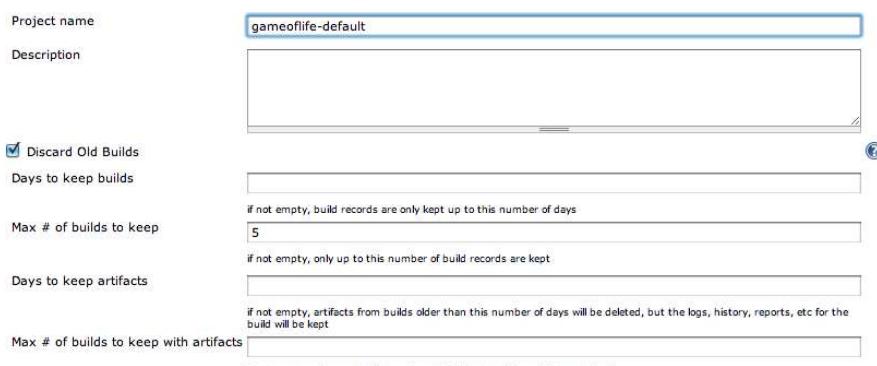
At the bottom of the page, there's a note: 'Cette page liste les mises à jour des plugins en cours d'utilisation.' and a 'Installer' button.

Formation Jenkins

121

## GESTION DE L'ESPACE DISQUE

- L'historique des builds prend de l'espace disque. De plus, Jenkins analyse les builds précédents lorsqu'il charge la configuration d'un projet.
- Il est recommandé de purger les anciens builds. Cela peut se faire sur un critère d'âge ou de nombre.



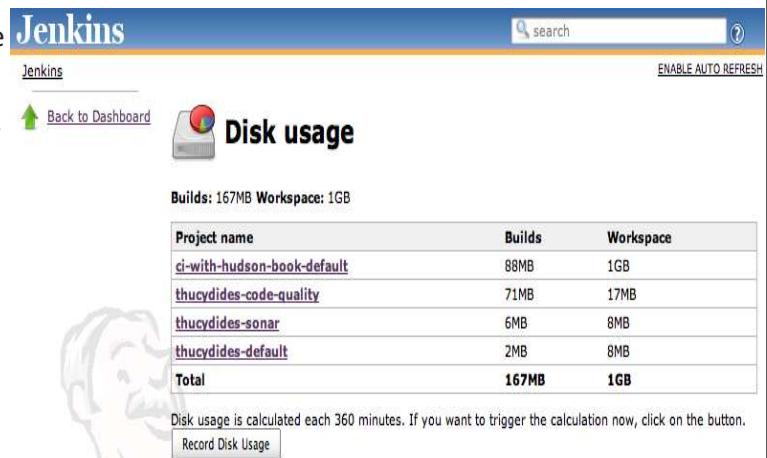
The screenshot shows the 'Build Polling and Artifacts Retention' configuration page for a project named 'gameoflife-default'. The form includes fields for 'Project name' (set to 'gameoflife-default'), 'Description' (empty), 'Discard Old Builds' (checked), 'Days to keep builds' (empty), 'Max # of builds to keep' (set to '5'), 'Days to keep artifacts' (empty), and 'Max # of builds to keep with artifacts' (empty). There are also notes explaining the retention logic for artifacts.

Formation Jenkins

122

## GESTION DE L'ESPACE DISQUE

- Le plugin Disk Usage est un des plus utiles pour un administrateur Jenkins. Ce plugin conserve et rapporte l'espace disque utilisé par vos projets. Il vous permet de repérer et corriger les projets qui utilisent trop d'espace.
- Vous pouvez installer le plugin Disk Usage de la façon habituelle, depuis l'écran "Gestion des plugins".
- Le plugin Disk Usage enregistre la quantité d'espace disque utilisée par chaque projet.
- Il ajoute également un lien "Disk usage" sur la page "Administrer Jenkins" qui permet d'afficher la quantité totale d'espace utilisé par vos projets

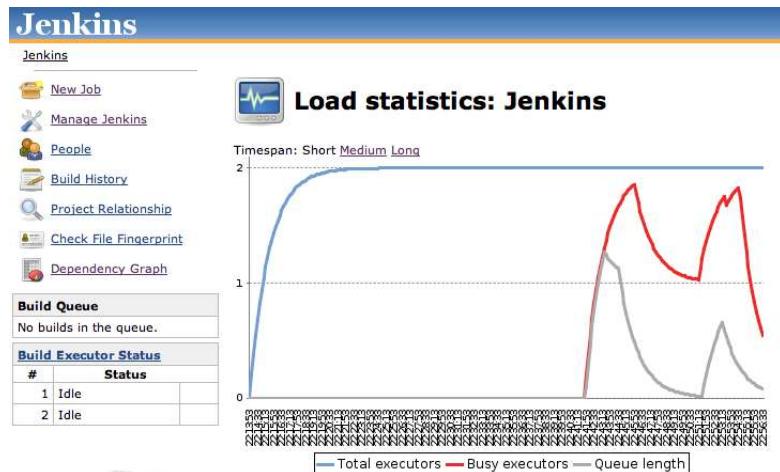


Formation Jenkins

123

## MONITORING DE LA CHARGE CPU

- Jenkins inclut une surveillance des activités serveur. Sur l'écran "Administrer Jenkins",
- cliquez sur l'icône "Statistiques d'utilisation".
- Cela affiche un graphique de la charge serveur au cours du temps
- Ce graphique contient trois métriques: nombre total d'exécuteurs, nombre d'exécuteurs occupés et longueur de la queue.



Formation Jenkins

124

## MONITORING DE LA CHARGE CPU

- Une autre possibilité est d'installer le plugin Monitoring.
- Ce plugin utilise JavaMelody afin de réaliser des rapports HTML complets sur l'état de votre serveur de build.
- Les rapports incluent la charge système et processeur, les temps moyen de réponse et l'utilisation de la mémoire.
- Une fois ce plugin installé, vous pouvez accéder aux graphiques JavaMelody depuis la page "Administrer Jenkins", en utilisant les entrées du menu "Monitoring of Hudson/Jenkins master" ou "Hudson/Jenkins nodes".

## LE PLUGIN JENKINS MONITORING



## GESTION DES SAUVEGARDES / RESTAURATION

- Jenkins stocke tous les paramètres, les journaux et les artefacts de construction dans son répertoire personnel,
  - par exemple, dans / var / lib / jenkins sous l'emplacement d'installation par défaut
- Pour créer une sauvegarde de votre installation de Jenkins, il suffit de copier ce répertoire.
- Le répertoire des travaux contient les tâches individuelles configurées dans l'installation de Jenkins.
- Vous pouvez déplacer un travail d'une installation Jenkins à une autre en copiant le répertoire de travail correspondant. Vous pouvez également copier un répertoire de travail pour cloner un travail ou renommer le répertoire.
- Cliquez sur le bouton recharger config dans l'interface utilisateur Web de Jenkins pour forcer Jenkins à recharger la configuration à partir du disque.

## GESTION DES SAUVEGARDES / RESTAURATION

- Jenkins prend en charge le plug-in SCM+Sync+configuration+plugin, qui permet de stocker toutes les modifications dans un dépôt Git.
  - <https://wiki.jenkins-ci.org/display/JENKINS/SCM+Sync+configuration+plugin>
- Il est également possible de maintenir la configuration de Jenkins dans un repo Git.

## BACKUP PLUGIN

- Ce plugin vous permet de configurer et de lancer des sauvegardes tant des configurations de vos tâches de build que de votre historique des builds.
- L'écran "Configuration" vous donne un important contrôle sur les éléments à sauvegarder.
- Vous pouvez choisir de seulement sauvegarder les fichiers XML de configuration, ou de sauvegarder avec l'historique des builds.
- Vous pouvez aussi sauvegarder les espaces de travail des tâches (généralement non nécessaire, comme discuté plus haut) et toutes empreintes numériques générées.

## BACKUP PLUGIN



### Backup config files

#### Backup configuration

Hudson root directory /Users/johnsmart/Projects/Demos/hudson-demo/jenkins-data

Backup directory

Format

File name template

Custom exclusions

Verbose mode

Configuration files (.xml) only

No shutdown

#### Backup content

Backup job workspace

Backup builds history

Backup maven artifacts archives

Backup fingerprints

## DES SAUVEGARDES AUTOMATISÉES PLUS LÉGÈRES

- Si tout ce que vous voulez sauvegarder est votre configuration des tâches de build, le plugin Backup Manager peut être considéré excessif. Une autre option est d'utiliser le plugin "Thin Backup", qui permet de planifier des sauvegardes complètes et incrémentales de vos fichiers de configuration.
- Vu qu'ils ne sauvegardent pas votre historique des builds ou vos artefacts, ces sauvegardes sont très rapides et peuvent ainsi être réalisées sans stopper le serveur pour les réaliser.

### Backup Configuration

Backup settings	
Backup directory	/var/data/backup/thin
Backup schedule for full backups	0 0 * * 1-5
Backup schedule for differential backups	0 * * * 1-5
Max number of stored full backups	40
<input checked="" type="checkbox"/> Clean up differential backups	

## RESTAURER UNE CONFIGURATION PRÉCÉDENTE

- Pour restaurer une configuration précédente, allez simplement à la page "Restore" et choisissez la date de la configuration que vous voulez réappliquer
- Une fois la configuration précédente restaurée, vous devez recharger la configuration Jenkins depuis le disque ou redémarrer Jenkins.

### Restore Configuration

Restore options	
restore backup from	2011-03-20 21:00



## JENKINS AVEC DOCKER

Formation Jenkins

133

## DOCKER

- Docker est une **solution de conteneurs**. Un conteneur utilise des solutions du Kernel Linux pour isoler des processus Linux.
- Docker permet de créer des images de certaines dépendances (Bases de données) et de les démarrer instantanément.
- Au lieu d'installer une dépendance en local, on peut donc utiliser un conteneur Docker qui fonctionnera de manière identique sur tous les postes

Formation Jenkins

134

## DOCKER – UN CONTENEUR

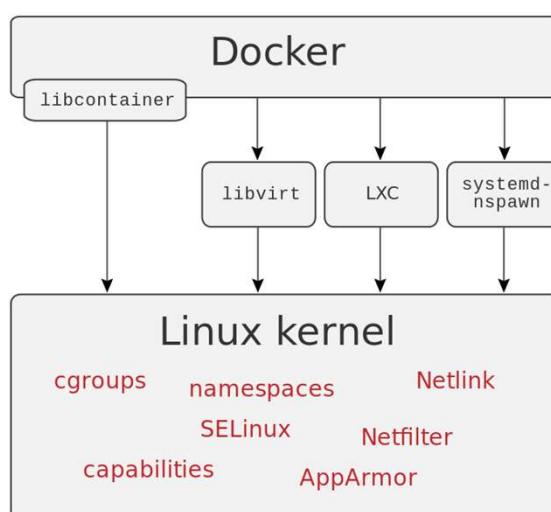
- Analogie pertinente
  - Contient ce dont on a besoin
    - (Du code, de la conf, des données....)
    - cela ne concerne que le producteur/**developpeur** et le consommateur/**end-user**
  - Facile à transporter
    - Le transporteur ne se soucie pas de ce qu'il contient
  - Facile à stocker/**héberger**



Formation Jenkins

135

## QU'EST-CE QUE DOCKER?



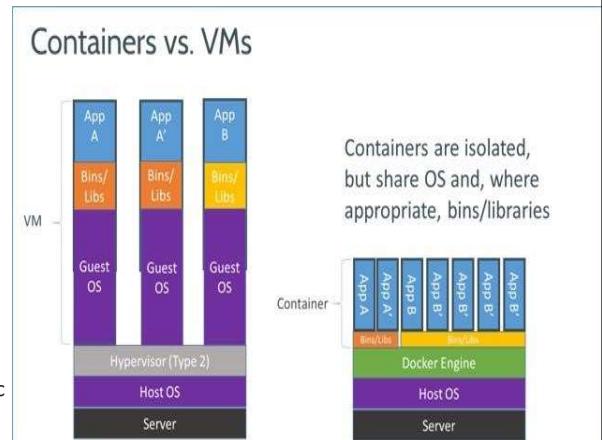
Formation Jenkins

136

# CONTENEURS VS MACHINES VIRTUELLES

- Un Conteneur

- Ca ressemble à une VM
  - Connexion via shell (ssh ou autre)
  - Ses propres process
  - Sa propre interface réseau
  - Son propre root
  - On peut installer des packages (apt-get install ...)
  - On peut démarrer des services
  - On peut faire du routing réseau (iptables)
- Ca ressemble à un « chroot »
  - Utilise le même kernel (et les mêmes modules) que l'hôte
  - Il n'a pas de PID 1 (System V, Systemd), ni de run-level
  - C'est un tas de process visibles depuis l'hôtes (invisibles avec une machine)



Formation Jenkins

137

# DOCKER FILE

- Fichier Texte
- Format déclaratif
- Permet de construire une image

Formation Jenkins

138

## DOCKER FILE – MOTS CLÉS

- FROM : définit le container de base (par exemple debian)
- RUN : permet de lancer une commande. Lancer à chaque compilation
- CMD : Lance une instruction au lancement du container (limiter à un par fichier)
- ENTRYPOINT : point d'entrée, permet d'exécuter une commande
- EXPOSE : expose un port
- ENV : pour avoir des variables dans Dockerfile
- ADD : ajout d'un fichier local dans le container
- VOLUME : monte un volume local sur le container
- WORKDIR : correspond à un cd

## DOCKERFILE - EXEMPLE

```
FROM registry.cdk.corp.mycompany/cdkbase

MAINTAINER CDK Team <cdk@mycompany.com>

ENV PROXY_PROTOCOL http
ENV PROXY_ADDR renn.proxy.corp.mycompany
ENV PROXY_PORT 8080
ENV HTTP_PROXY ${PROXY_PROTOCOL}://${PROXY_ADDR}:${PROXY_PORT}/
ENV HTTPS_PROXY ${HTTP_PROXY}
ENV http_proxy ${HTTP_PROXY}
ENV https_proxy ${HTTP_PROXY}

RUN echo "Acquire::http::Proxy \"${HTTP_PROXY}\";" > /etc/apt/apt.conf.d/proxy && \
    echo "Acquire::https::Proxy \"${HTTP_PROXY}\";" > /etc/apt/apt.conf.d/proxy && \
    mkdir -p /opt/java && cd /opt/java && \
    wget --user=cdk --password=viewer http://.../jdk-7u75-linux-x64.gz && \
    gunzip jdk-7u75-linux-x64.gz && \
    tar xf jdk-7u75-linux-x64 && \
    rm -f jdk-7u75-linux-x64 && \
    ln -s /opt/java/jdk1.7.0_75/bin/java /usr/bin/java && \
    ln -s /opt/java/jdk1.7.0_75/bin/javac /usr/bin/javac
```

## DOCKER - BUILD, SHIP & RUN



- Build

- Construire **une image** d'un conteneur
  - De façon manuelle et interactive (docker commit)
  - De façon automatisée (Dockerfile)

- Ship

- Partage des images sur un registry
  - Docker push
  - Docker pull

- Run

- Conteneur Linux
  - cGroups

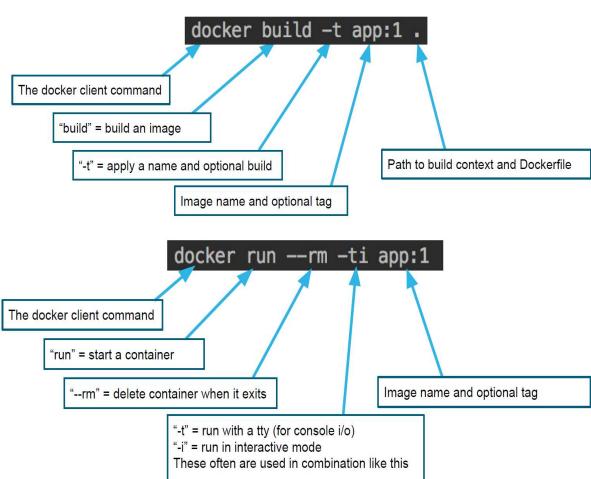
## DOCKER: LIGNE DE COMMANDE

- Build

- Ex : docker build -t mongodb .
- Compilation d'une image en container
- Une fois compilé, le container est figé
- Ajout de tag

- Run

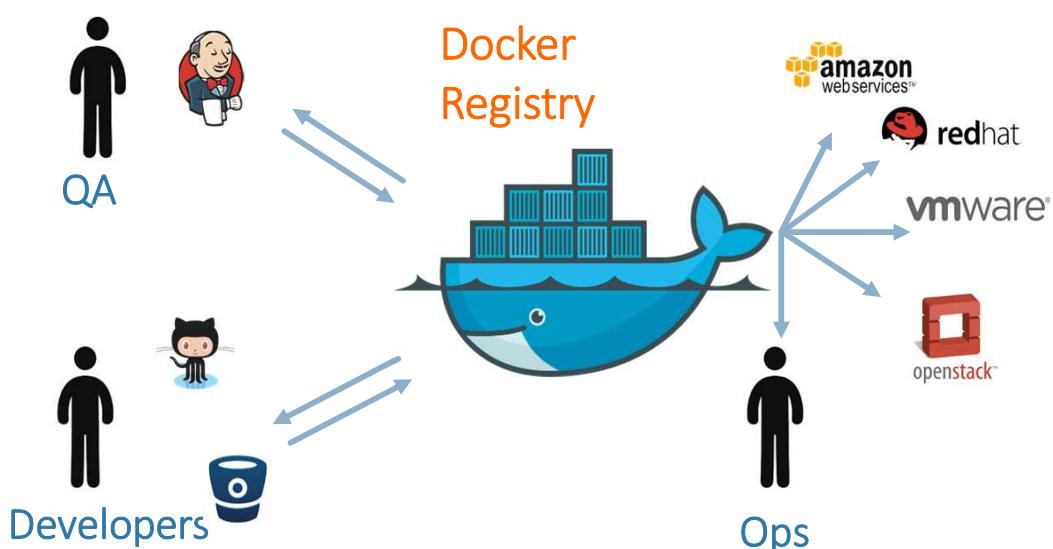
- Ex : docker run 27017:27017 mongodb
- Execution du container (retourne un id de container)
- Transfert de port
- Monter un volume
- Logs
- Mettre des variables d'environnement
- Possibilité d'ajouter des configurations réseaux (mac adresse, DNS, mode réseau)
- Divers : clean container, restart, contraintes CPU et mémoire, runtime privilege, lxc configuration



## DOCKER: LIGNE DE COMMANDE

- Start/stop
  - Ex : docker stop {id / nom container}
  - Démarrage / Arrêt du container
- port
  - Ex : docker port {id / nom container}
  - Liste les ports du container
- rm
  - Ex : docker rm {id / nom container}
  - Suppression du container
- ps
  - Ex : docker ps -a
  - Liste tous les containers

## DOCKER REGISTRY



## INTEGRATION DE JENKINS AVEC DOCKER

- Jenkins fonctionne avec docker de deux façons:
  1. Jenkins peut soit s'installer sur un conteneur docker.
  2. Soit être installé sur un serveur (non de type conteneur docker), et orchestrer des images docker, notamment des images éphémères
- Il est fortement recommandé de ne pas cumuler les deux options au sein du même conteneur docker (conteneur au sein d'un conteneur à éviter!)



ALLER PLUS LOIN AVEC JENKINS ENTERPRISE BY CLOUDBEES

## USAGE TYPIQUE DE JENKINS

- Cas 1: Monolithique
  - Le nombre de jobs augmente et le serveur devient lent
  - Conflit entre les équipes
  - Maintenance onéreuse
    - Plusieurs plugins à gérer
    - Difficile de donner des environnements avec une conf dédiée aux équipes
    - Peur de l'upgrade
  - Ne répond pas en général aux besoins des équipes de développement
  - La gestion des droits d'accès devient complexe

The screenshot shows the Jenkins master interface. At the top, there's a cartoon Jenkins head icon. Below it is a navigation bar with tabs: Build, Deploy, Jenkins Tasks, and a plus sign. Underneath the tabs is a table titled 'Last Success' with three rows. Each row contains a green circle icon, a small Jenkins head icon, a task name, and a timestamp. The tasks are: 'Deploy to Production Server' (8 mo 22 days ago), 'Deploy to Staging Server' (1 yr 1 mo ago), and 'Deploy to Test Server' (2 hr 4 min ago). The word 'Master' is centered at the bottom of the interface.

Formation Jenkins

147

## USAGE TYPIQUE DE JENKINS

- Cas 2: Utilisation de plusieurs masters
  - Manque de vision centralisé dans un tableau de bord
  - Upgrade des plugins dans les différents masters
  - Le build sur un serveur ne peut pas être reproduit sur un autre
  - Effort dupliqué de maintenance et donc de coût

The image contains four separate screenshots of Jenkins master interfaces, each labeled 'Master'. They are arranged in a 2x2 grid. Each interface shows a table with three deployment tasks: 'Deploy to Production Server' (8 mo 22 days ago), 'Deploy to Staging Server' (1 yr 1 mo ago), and 'Deploy to Test Server' (2 hr 4 min ago). The interfaces are associated with different teams: 'TEAM A', 'TEAM C', 'TEAM B', and 'TEAM D'. Each team has its own unique color scheme for the Jenkins head icons in the table.

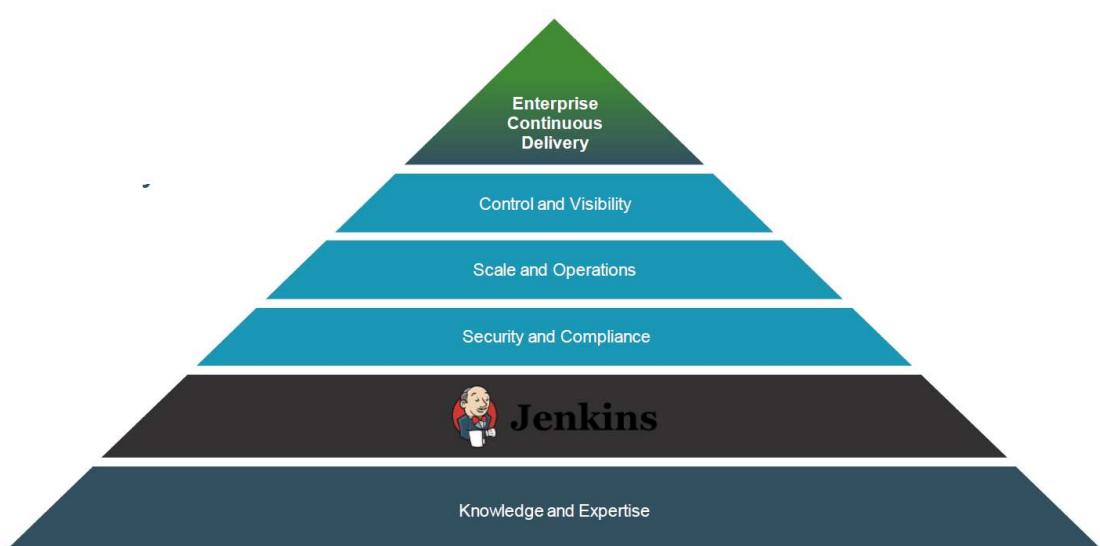
Formation Jenkins

148

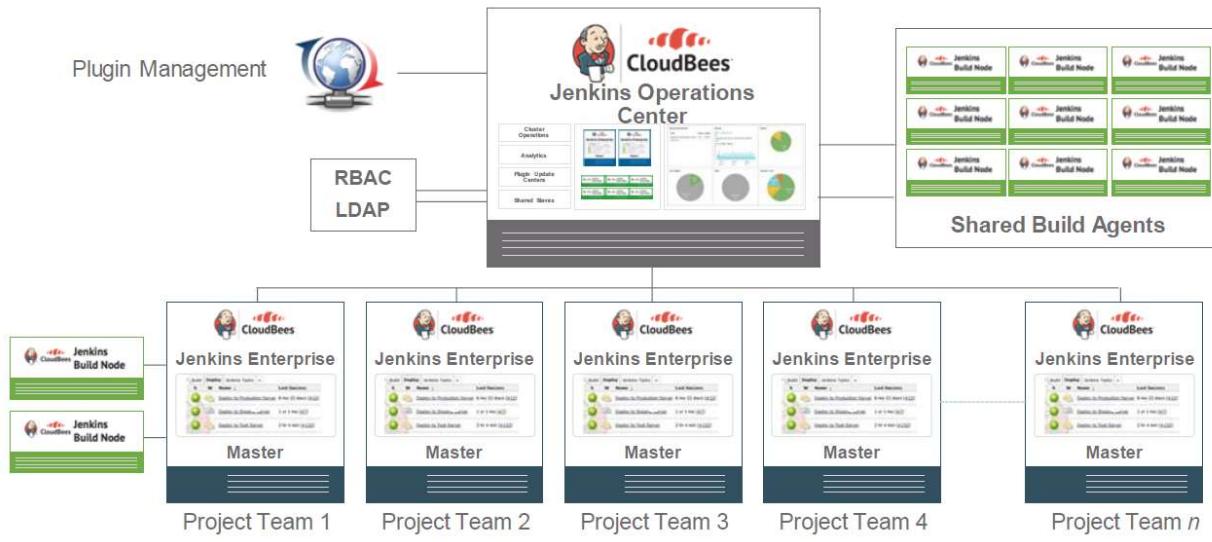
## CLOUDBEES JENKINS ENTERPRISE

- CloudBees Jenkins Enterprise offre toutes les fonctionnalités CloudBees Jenkins Team, en y ajoutant la sécurité, l'évolutivité, la facilité de gestion et la résilience.
- Construit sur une architecture cloud, CloudBees Jenkins Enterprise réduit les coûts d'infrastructure et améliore la disponibilité grâce à des basculements et des récupérations automatiques des agents.
- L'édition Enterprise convient parfaitement aux entreprises qui recherchent un logiciel « Jenkins as a Service » élastique et prêt à être utilisé dans le cloud ou en local.

## CONTINUOUS DELIVERY A L' ÉCHELLE DE L'ENTREPRISE



## VUE DE L'ARCHITECTURE CLOUDBEES



Formation Jenkins

151

## COMPARAISON JENKINS OPEN SOURCE ET CLOUDBEES

Fonctionnalité	Open Source	CloudBees Jenkins Solutions	
	Jenkins	Team	Enterprise
<b>COMMUNAUTÉ</b> La plate-forme d'automatisation leader en matière d'intégration et de déploiement continu Écosystème de 1 000 plugins	X X	X X	X X
<b>ASSISTANCE TECHNIQUE PROFESSIONNELLE</b> Plugins Jenkins vérifiés et gérés Distribution vérifiée du système de base Jenkins Assistance 24 h/24, 7 j/7 par des experts à disposition		X X X	X X X
<b>INSTALLATION ET MISES À JOUR</b> Publications mensuelles Versions corrigées pour la prise en charge à long terme Corrections pour les vulnérabilités de sécurité de type « zero-day » Fonctionnalités de CloudBees Jenkins Enterprise Installation en quelques minutes sur un cloud privé ou une infrastructure existante		X X X	X X X X
<b>DÉPLOIEMENT CONTINU AVEC JENKINS</b> Créez des pipelines de livraison complexes avec Jenkins Pipeline Builds qui résistent aux pannes de masters à l'aide de Jenkins Pipeline Syntaxe déclarative simple pour la modélisation de pipeline	X X X	X X X	X X X
<b>DÉPLOIEMENT CONTINU POUR L'ENTREPRISE</b> Informations sur les performances de pipeline grâce à la fonctionnalité de visualisation des étapes Redémarrage des builds à partir d'emplacements contrôlés en cas de panne de master et d'agent de build Configuration automatique du déploiement continu pour les pull requests et les branches	X	X	X X X
<b>CONTENEURS DOCKER</b> Créez des pipelines de livraison à l'aide de Jenkins Pipeline et Docker Obtenez des informations complètes sur les blocages de pipeline grâce à la traçabilité de Docker Partagez des environnements de génération standardisés (y compris les configurations Docker) sur un cluster de masters	X X	X	X X X

Formation Jenkins

152

## COMPARAISON JENKINS OPEN SOURCE ET CLOUDBEES

<b>GESTION DES ÉQUIPES</b> Organisez des équipes à l'aide de la fonctionnalité Folders Fonctionnalités d'organisation et de séparation d'équipes étendues à l'aide de Folders Plus Partagez des agents de build de façon dynamique Provisionnement de master en un clic (et programmé)	X	X	X X X X
<b>SÉCURITÉ</b> Gestion des informations d'identification Authentification unique/externe à l'aide de la fonctionnalité de contrôle des accès par rôle Isolez les agents de build sensibles	X	X	X X X
<b>CLOUD</b> Intégration à des technologies cloud leaders Prise en charge de clouds privés Support natif d'Amazon Web Services et OpenStack	X	X	X X X
<b>PRODUCTIVITÉ DES DÉVELOPPEURS</b> Gagnez du temps et favorisez la collaboration à l'aide de la fonctionnalité GitHub Pull Request Builder Évitez les mauvais Commits Git grâce à la fonctionnalité Validated Merge Partagez des bonnes pratiques à l'aide de la fonctionnalité de modèles	X	X	X X X
<b>RÉSILIENCIA DES BUILDS ET DES MASTERS</b> Pipelines qui résistent aux pannes de masters à l'aide de la fonctionnalité Long-Running Build Basculement automatique permettant une récupération en cas de panne de serveur master à l'aide de la fonctionnalité de haute disponibilité	X	X	X X

Formation Jenkins

153

## COMPARAISON JENKINS OPEN SOURCE ET CLOUDBEES

<b>UTILISATION OPTIMISÉE</b> Générations de builds plus rapides sur les agents de build dispersés géographiquement à l'aide de la fonctionnalité d'archivage rapide Ralentissez les builds en cas de charges élevées à l'aide de la fonctionnalité Label Throttle Build			X X
<b>GESTION D'ENTREPRISE</b> Gérez l'obtention et le déploiement de plugins grâce au centre de mises à jour personnalisées Redimensionnez l'infrastructure, les masters et les agents de façon élastique Sauvegarde et récupération automatique			X X X
<b>ANALYSES D'ENTREPRISE</b> Surveillance et alertes de l'état de l'infrastructure Tableaux de bord personnalisés sur les analyses de performance et les builds Informations sur l'utilisation des plugins sur les masters			X X X

Formation Jenkins

154



## SYNTHESE

Formation Jenkins

155

## JENKINS - SYNTHESE

- Jenkins permet:

- Réduction du temps de correction des bogues
- Intégration des modifications régulièrement
- La portion de code à débugger est faible lors de la détection des anomalies
- Amélioration du travail collaboratif
- La fin du « ça marche sur mon poste »
- Amélioration de la qualité du logiciel
  - Le code et le design de l'application répondent aux exigences des standards, le résultat du build est un produit complètement fonctionnel et testable
- Promotion des bonnes habitudes de test

Formation Jenkins

156

## JENKINS - SYNTHESE

- Jenkins permet d'obtenir automatiquement des indicateurs d'avancement et d'état qualitatif d'un projet en cours de développement
- L'objectif de la construction d'intégration est de produire un logiciel exécutable qui peut être déployé et testé fonctionnellement. Une démonstration avec le client est toujours possible
- Les outils d'assurance qualité participent au contrôle des risques:
  - Risque de faible qualité logiciel
  - Risque de découverte tardive des défauts
- Pilotage / Suivi
  - Jenkins est devenue un outil fondamental pour le pilotage des projets : elle améliore la visibilité
- Qualité / Productivité
  - L'intégration continue est un formidable levier d'amélioration de niveau d'expertise technique : elle est l'un des moteurs de d'amélioration des gains en termes de qualité et de productivité
- Transparence / Rétroaction
  - L'intégration continue améliore le professionnalisme de processus de développement

## BIBLIOGRAPHIE

- <http://martinfowler.com/articles/continuousIntegration.html>
- <https://jenkins.io/doc/>
- <http://sonar.codehaus.org/>
- <http://maven.apache.org/>
- <https://jenkins-le-guide-complet.github.io/>
- <http://pkg.jenkins-ci.org>



## QUESTIONS / RÉPONSES