

Intégration Continue

Plan

1. Principe de l'intégration continue

- 1.1 Rappel des tests d'intégration
- 1.2 Intégration Continue

2. L'intégration continue en détail

- 2.1 Intégration du code source
- 2.2 Compilation
- 2.3 Exécution des tests
- 2.4 Analyse statique

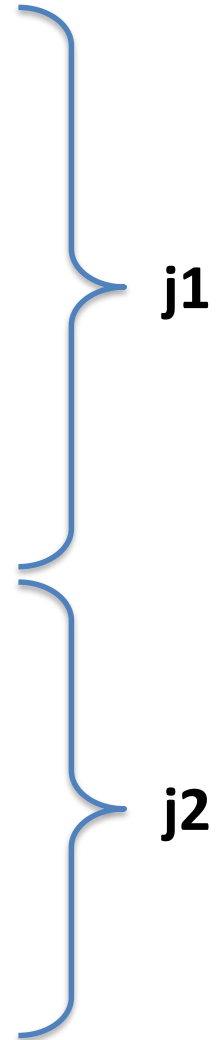
3. Jenkins

- 3.1 Introduction
- 3.2 Installation
- 3.3 Administration
- 3.4 Maven

4. Bonnes pratiques

5. Travaux pratiques

- 5.1 Installation de Jenkins/Maven
- 5.2 Création d'un job pour lancer des scripts selenium
- 5.3 Création d'un job pour lancer des scripts Cucumber





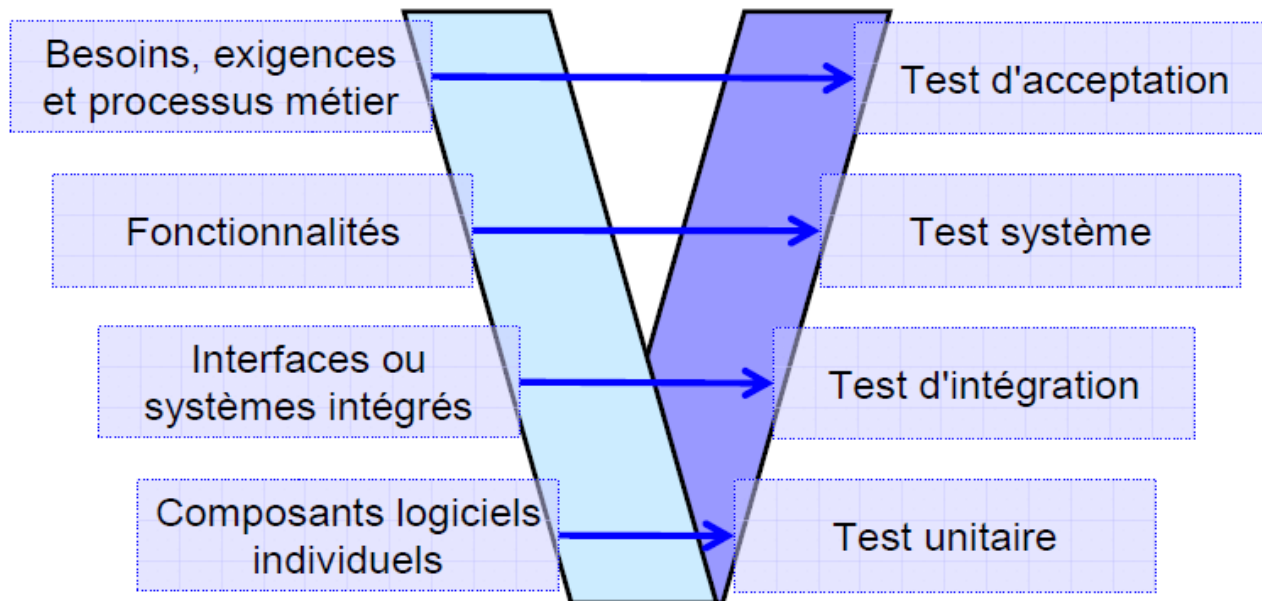
1. Principe de l'Intégration Continue

1.1 Rappel des tests d'intégration

1.2 Qu'est ce que l'intégration continue?

1.1 Rappel: les différents niveaux de test

4 principaux niveaux de test apportés avec le cycle en V:



1.1 Rappel: les différents niveaux de test

Test de composants (Test unitaire) :

Recherche des défauts et vérification du fonctionnement, des logiciels, modules, programme, objets, classes qui sont testables séparément.

Bases de test

- Exigences des composants,
- Conception détaillée,
- Code.

Objets de test

- Composants,
- Programmes,
- Conversions de données / utilitaires,
- Programmes de migration,
- Module de base des données.

1.1 Rappel: les différents niveaux de test

Test de composants (Test unitaire) :

Recherche des défauts et vérification du fonctionnement, des logiciels, modules, programme, objets, classes qui sont testables séparément.

Type de test

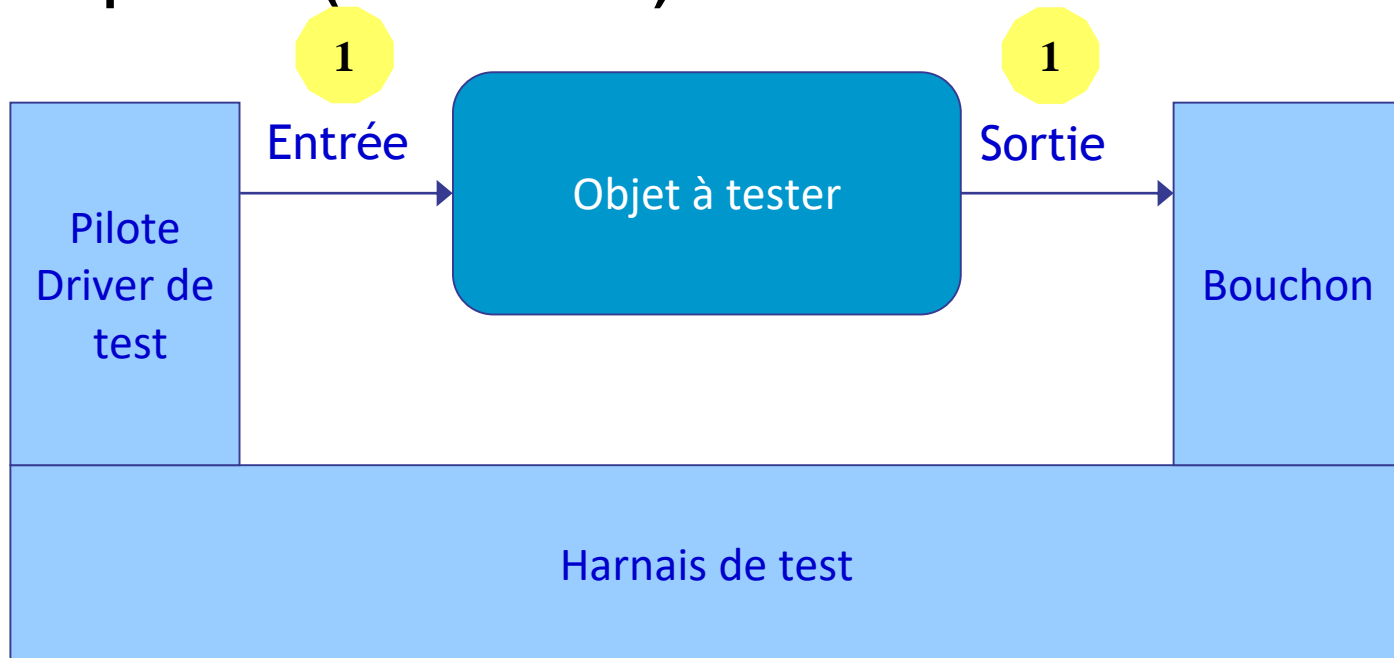
- Structurels,
- Non structurels,

Famille de test

- Fonctionnel,
- Non fonctionnel,

1.1 Rappel: les différents niveaux de test

Test de composants (Test unitaire) :



1.1 Rappel: les différents niveaux de test

Test de composants (Test unitaire) : Techniques de test

Approche Test First

- TDD « Test Driven Development », conception des tests puis développement du code ensuite.

Tests Statiques

- Revue de code,
- Graphe de contrôle du code,
- Couverture des branches,
- Couverture des décisions.

Tests Dynamiques

- Test fonctionnel de composants,
- Test de robustesse,...

1.1 Rappel: les différents niveaux de test

Test d'intégration :

Les tests d'intégration testent les interfaces entre les composants, les interactions entre différentes parties d'un système.

Bases de test

- Conception,
- Architecture,
- Workflows

Objets de test

- Implémentation base de données,
- Sous-systèmes,
- Interfaces,
- ...

1.1 Rappel: les différents niveaux de test

Test d'intégration :

Les tests d'intégration testent les interfaces entre les composants, les interactions entre différentes parties d'un système.

Type de test

- Fonctionnel,
- Non fonctionnel,
- Workflows

Profil testeur

- Testeur avec forte compétences techniques et en architecture

Règles

- Plus la portée est vaste, plus il est difficile d'isoler le défaut,
- Intégration incrémentale préférable au « Big Bang »,
- Utilisation de harnais de tests.

1.1 Rappel: les différents niveaux de test

Test d'intégration Big-Bang:

Un type de tests d'intégration dans lequel les éléments logiciels, matériel ou les deux sont combinés en une fois en un composant ou un système, plutôt qu'effectués par étape [selon IEEE 610].

.

Avantages

- Peu ou pas de contraintes de planification,
- S'adapte aux petits et moyens systèmes.

Inconvénients

- Approche unitaire,
- Pas de maîtrise des tests,
- Difficultés à déterminer d'où vient l'erreur en cas d'anomalies.

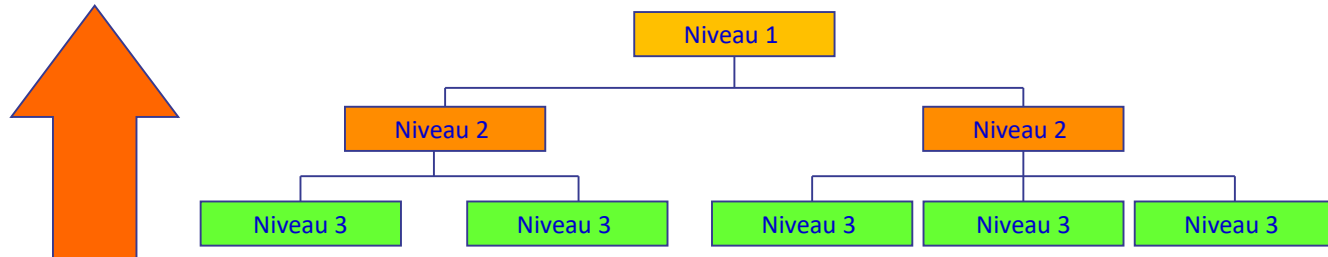
1.1 Rappel: les différents niveaux de test

Test d'intégration de bas en haut:

Une approche incrémentale de tests d'intégration où le niveau le plus bas des composants sont testés d'abord, et ensuite utilisés pour faciliter les tests des composants de plus haut niveau. Ce processus est répété jusqu'au test du composant le plus haut de la hiérarchie.

·
·

**NÉCESSITE LA CONCEPTION D'UN
NOMBRE IMPORTANT DE PILOTES
POUR DIRIGER CHACUN DES NIVEAUX D'INTÉGRATION.**



1.1 Rappel: les différents niveaux de test

Test d'intégration de bas en haut:

Avantages

- Composant de bas niveau mieux testés,
- Localisation facile des défauts,
- Facile à mettre en œuvre avec un simulateur de tests,
- Démarche préventive,
- L'observation des résultats des tests est plus facile

Inconvénients

- Le programme n'existe pas jusqu'à ce que le dernier module soit ajouté,
- Erreur de conception détectée très tard,
- Demande un effort important de simulateurs.

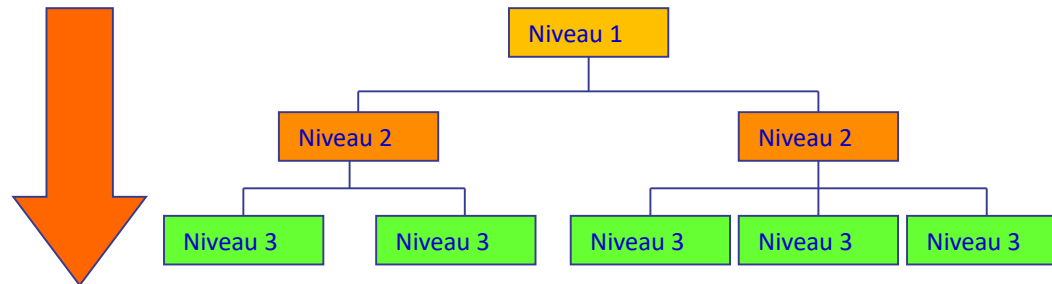
1.1 Rappel: les différents niveaux de test

Test d'intégration de haut en bas:

Une approche incrémentale des tests d'intégration où les composants en haut de la hiérarchie sont testés d'abord, avec les composants de niveau inférieur simulés par des bouchons. Les composants testés sont ensuite utilisés pour tester des composants de niveaux inférieurs. Le processus est répété jusqu'à ce que les composants de plus bas niveau ont été testés.

•
•

NÉCESSITE LA CONCEPTION D'UN NOMBRE IMPORTANT DE BOUCHONS, POUR REMPLACER LES COMPOSANTS NON ENCORE DÉVELOPPÉS.



1.1 Rappel: les différents niveaux de test

Test d'intégration de bas en haut:

Avantages

- Composant de bas niveau mieux testés,
- Localisation facile des défauts,
- Facile à mettre en œuvre avec un simulateur de tests,
- Démarche préventive,
- L'observation des résultats des tests est plus facile

Inconvénients

- Le programme n'existe pas jusqu'à ce que le dernier module soit ajouté,
- Erreur de conception détectée très tard,
- Demande un effort important de simulateurs.

1.2 Intégration continue (1/5)

Challenge :

En Agile, il faut livrer un incrément produit fiable, fonctionnel et intégré dans le logiciel à la fin de chaque sprint.

L'intégration continue est apparue avec les pratiques XP (eXtrem Programming)

L'intégration continue offre la solution à ce challenge :

- ✓ En fusionnant tous les changements faits au logiciel
- ✓ En intégrant tous les composants modifiés régulièrement

Gestion de configuration
Compilation,
Build du logiciel
Tests
Déploiement
etc...

Contenus dans



**Un seul processus
simple, automatisé
et répétitif**

1.2 Intégration continue (2/5)

Le processus d'intégration continue est constitué des activités automatisées suivantes :

- ✓ Analyse statique du code
- ✓ Compilation
- ✓ Tests unitaires
- ✓ Déploiement
- ✓ Test d'intégration
- ✓ Reporting (tableau de bord)

Intégration continue (3/5)

Syllabus 1.2.4

- Un processus de construction et de test qui est fait sur une base journalière détecte les défauts d'intégration tôt et rapidement.
- L'intégration continue permet :
 - ✓ L'exécution régulière de tests automatisés
 - ✓ L'accès par tous les membres de l'équipe aux résultats de test
 - ✓ La continuité des tests de régression durant toute l'itération
- Les tests manuels sont ainsi concentrés sur :
 - ✓ Les nouvelles fonctionnalités,
 - ✓ Les changements implémentés,
 - ✓ Les tests de confirmation des défauts corrigés

1.2 Intégration continue (4/5)

Avantages :

- ✓ Une détection au plus tôt / analyse plus facile des causes racines des problèmes d'intégration
- ✓ Un feedback régulier sur la qualité et le fonctionnement du code
- ✓ Éliminer les risques prévisibles associés à l'intégration Big-Bang
- ✓ Réduire les risques de régression
- ✓ Fournir une disponibilité constante de logiciels exécutables durant le sprint à des fins d'essai, de démonstration...
- ✓ Réduire les activités de test manuelles et répétitives
- ✓ Fournir un feedback rapide sur les décisions prises pour améliorer la qualité et les tests

1.2 Intégration continue (5/5)

Cependant, l'intégration continue n'est pas sans risque ni challenge :

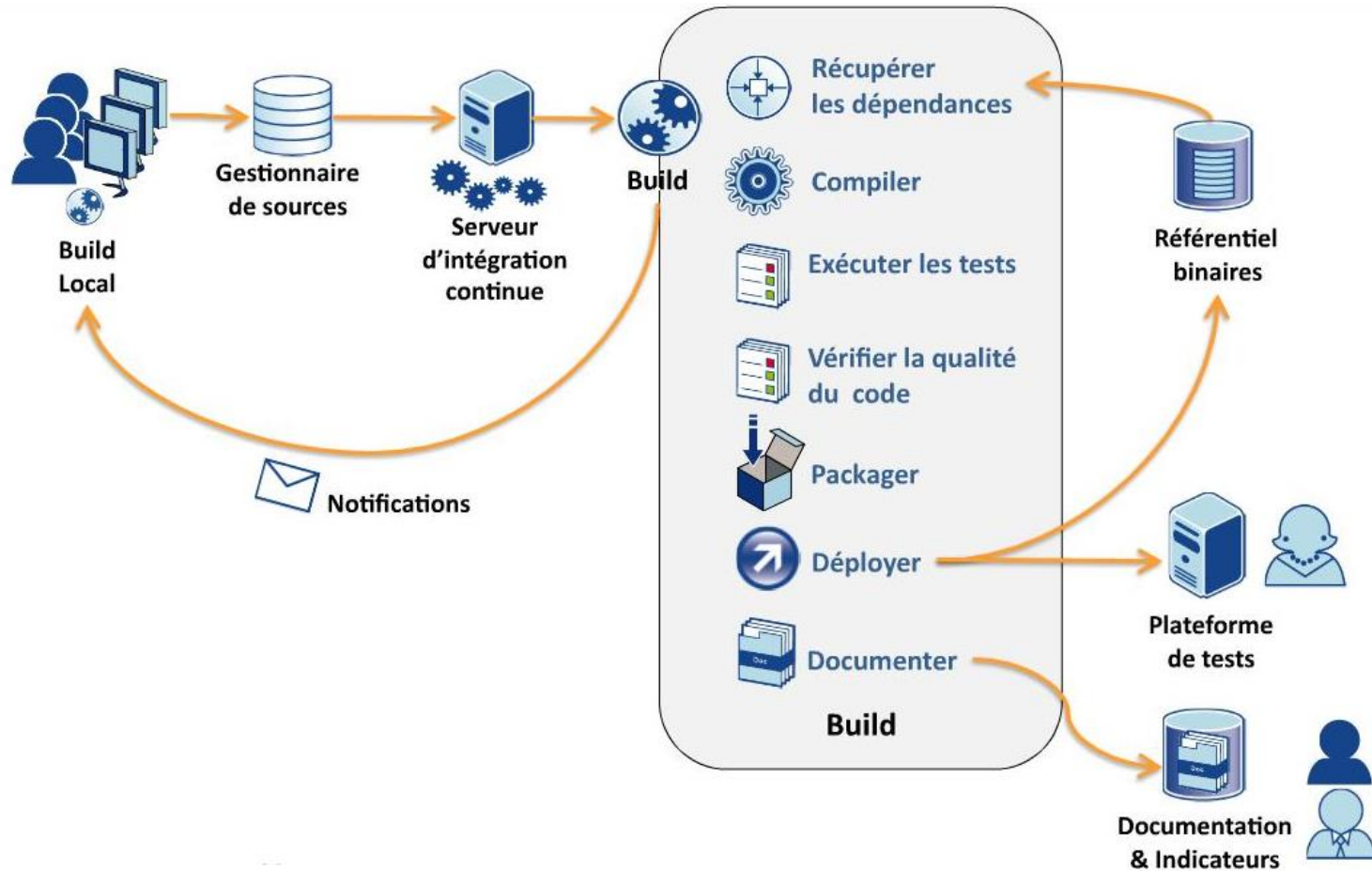
- Les outils doivent être mis en place et maintenus
- Le processus doit être défini et établi
- L'automatisation des tests :
 - ✓ Requiert des ressources supplémentaires
 - ✓ Peut être complexe à mettre en place
 - ✓ Nécessite une couverture de test approfondie
- Les équipes font parfois trop confiance aux tests unitaires et font peu de tests systèmes et d'acceptation



2. Intégration Continue en détail

- 2.1 Intégration des codes sources
- 2.2 Compilation
- 2.3 Exécution des tests unitaires
- 2.4 Analyse statique
- 2.5 Déploiement

Intégration continue



2.1 Intégrer les sources

Le premier objectif de l'intégration continue est bien sûr l'Intégration des développements des différents développeurs

- Chaque développeur code et test dans son environnement local
- Une fois validé, le développeur fait une mise à jour du référentiel de code (SCM) en faisant une opération de commit ou check-in
- SCM (Source Control Management) permet de gérer la configuration des sources

2.1 La gestion de configuration

Il s'agit d'une discipline appliquant une direction et surveillance technique et administrative pour :

- Identifier et documenter les caractéristiques fonctionnelles et physiques d'un élément de configuration,
- Contrôler les modifications de ces caractéristiques,
- Enregistrer et informer des modifications et états d'implémentation,
- Vérifier la conformité avec des exigences spécifiées [IEEE 610].*

OBJECTIF

Elle regroupe les techniques et les règles destinées à GARANTIR L'INTÉGRITÉ ET LA COHÉRENCE D'UNE CONFIGURATION LOGICIELLE OU D'UN PRODUIT au fil des modifications et des évolutions.

2.1 La gestion de configuration

Éléments de configuration

- Les documents de spécifications,
- Le code
- Les fichiers de configuration
-

De même que le testware, les livrables de tests comportent l'ensemble des éléments de configuration spécifique aux tests :

- Les documents de spécifications des tests,
- Les documents de stratégies de tests, les plans de test, ...
- Les conditions de tests, les exigences de tests,
- Les cas de tests,
- Les procédures de tests,
- Les données de tests,
- Les scripts de tests,
- Les simulateurs, les bouchons, les harnais de tests,
- Les outils de tests,
- Les environnements de test,...

2.1 La gestion de configuration

- **PROCESSUS DE GESTION DU TESTWARE SOUS CONFIGURATION**

- 1) Identification des **ÉLÉMENTS DE CONFIGURATION** :

- Recensement de tous les éléments configurables
- Définition de la politique de configuration dans la documentation du plan de test durant la planification (outils, techniques, procédures, référentiel, ...)

- 2) Suite à un **COMITÉ DE CONTRÔLE DES MODIFICATIONS** :

- Les modifications, les évolutions sont approuvées

- 3) Sous **CONTRÔLE DE CONFIGURATION** :

- Les changements sont tracés,
- Les éléments de configuration impactés sont identifiés grâce à leur changement **d'ÉTAT DE CONFIGURATION** et la **TRAÇABILITÉ** avec les évolutions et les objets de test.

- 4) Les éléments identifiés sont ensuite référencés, archivés dans la documentation.

2.1 La gestion de configuration

•Focus TEST: LES BÉNÉFICES POUR LE TESTEUR

UNICITÉ DES BASES DE TEST

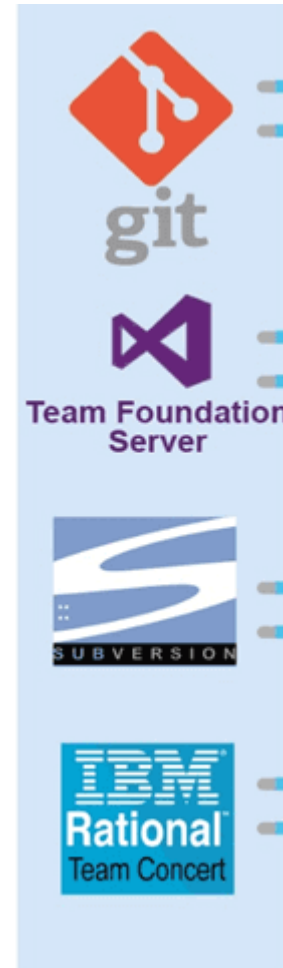
- On est sûr de ce qu'on teste.

IDENTIFICATION PRÉCISE DE L'ÉLÉMENT TESTÉ

- Facilite, par exemple, la description d'une fiche d'incident pour sa reproductibilité,
- Traçabilité des évolutions aux éléments de tests (spécifications, cas de tests, anomalies, environnement de tests),
- Facilite l'analyse d'impact des évolutions et permet d'affiner une stratégie de test.

2.1 Intégrer les sources

Les principaux outils de SCM



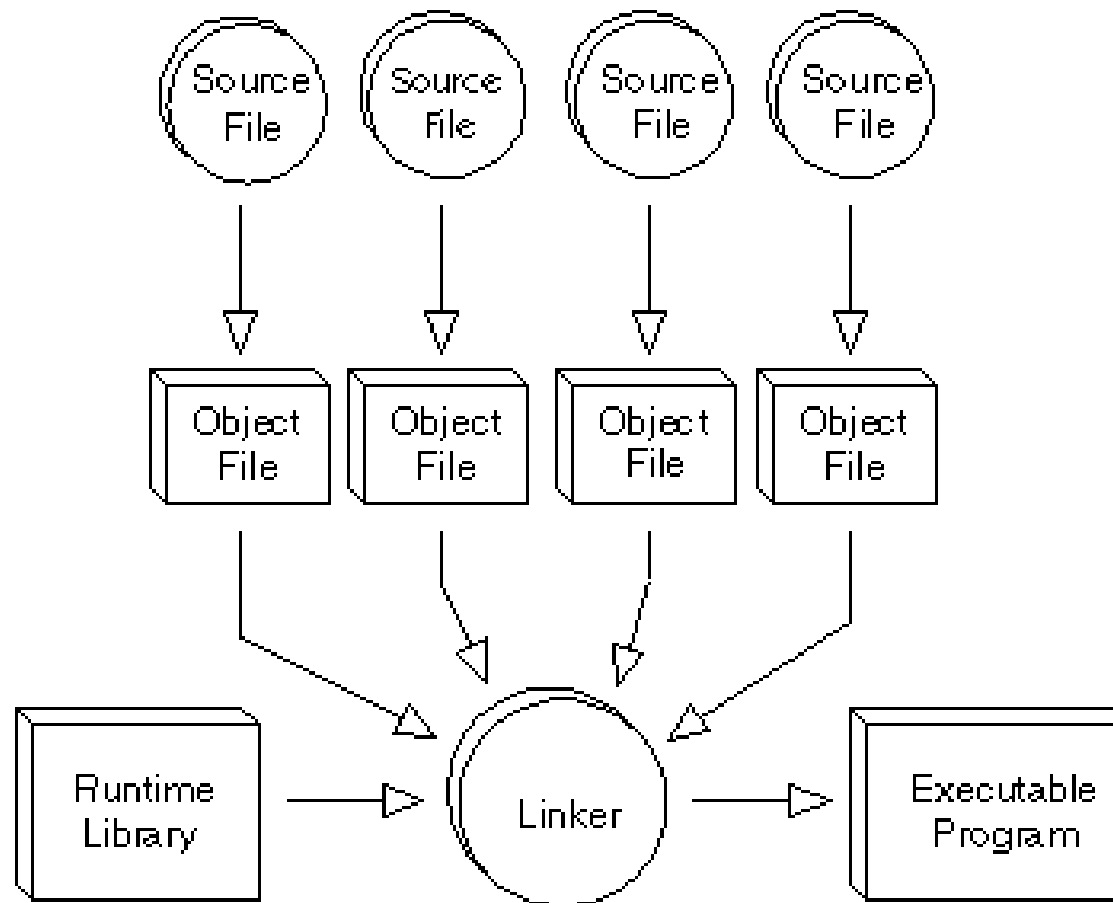
2.2 Compilation

La deuxième étape du processus consiste en la compilation des sources

- Une fois les derniers codes sauvegardés dans le SCM
- L'outil d'intégration réalise la compilation des sources
- La compilation est l'opération de transformation des sources en un livrable exécutable
Elle dépend du langage et des compilateurs



2.2 Compilation



2.2 Compilation

Compilateur

- Javac pour le langage java
- GCC pour le langage C
- Le produit issu de la compilation est appelé un **BUILD**.

2.3 Les tests unitaires

- ❑ **Objet de test:** Code
- ❑ **Objectif:** vérifier le composant développé en isolation (indépendamment des autres composants, de la base de données,)
- ❑ **Responsabilités:** Spécification des tests par les développeurs
- ❑ **Exécution des tests:** Intégrée dans l'intégration

Les tests sont exécutés à chaque commit après compilation

- ❑ **Tests en Isolation:** Utilisation de mocks/bouchons pour les dépendances
- ❑ Tests automatisés
- ❑ **Condition de lancement:** A chaque commit du développeur



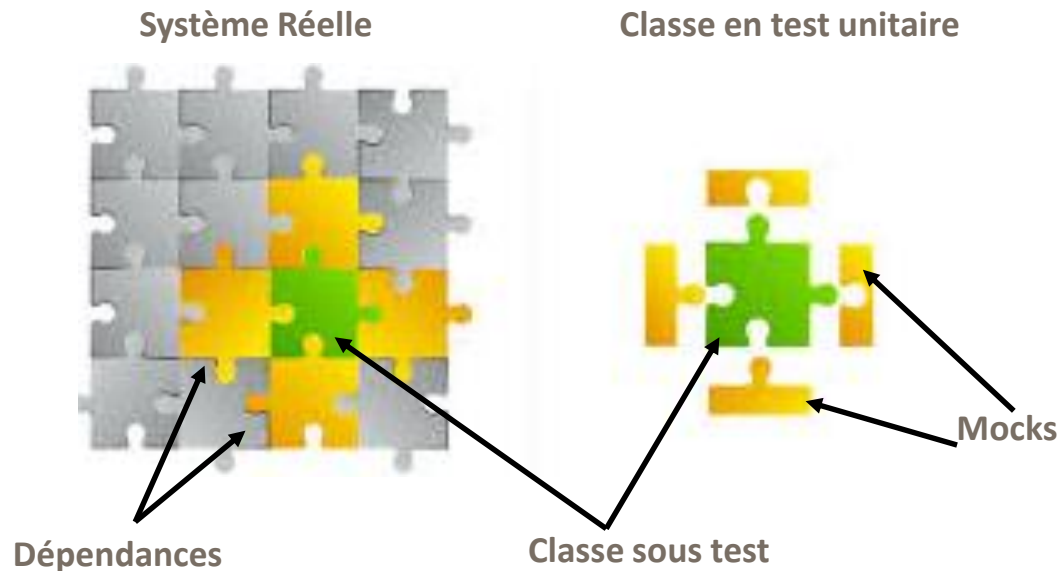
JUnit

mockito

2.3 Les tests unitaires

Test en isolation totale

- ❑ Test Unitaire = Unitaire, isolé, seul, 1 objectif
- ❑ 1 Classe, 1 Méthode à la fois
- ❑ Il faut éviter de faire des tests en accédant aux composants externes (DB, ...=
- ❑ Il faut utiliser des Mocks et/ou bouchons



2.3 Les tests unitaires

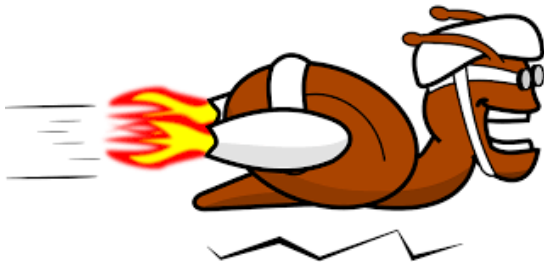
Simple et claire

- ❑ Le test unitaire doit être simple
- ❑ Il doit contenir les éléments suivants
 - Setup
 - Step
 - Assertion
 - TearDown
- ❑ Le nom de la méthode de test unitaire doit permettre de comprendre le comportement testé
- ❑ Pas de conditions, ni de boucles dans un test unitaire

2.3 Les tests unitaires

Rapide

- ☐ Un test unitaire doit être rapide
- ☐ Il doit s'exécuter durant le build
- ☐ Un test long implique
 - soit une complexité de test de l'objet (Refactoring nécessaire)
 - soit des dépendances externes (à déplacer en test d'intégration)



2.4 Analyse statique

Analyse statique: Analyse des artefacts logiciels par exemple, exigence ou code, effectuée sans l'exécution de ces artefacts de développement de logiciels. L'analyse statique est habituellement effectuée au moyen d'un outil assistant dans cette tâche.

ANALYSE STATIQUE

- Effectuée sans exécuter le code,
- Détecte les défauts qui sont difficiles à trouver par les tests dynamiques dans le code source et les modèles logiciels,
- Trouve des défauts comme les revues,
- **LES OUTILS D'ANALYSE STATIQUE** analysent :
 - Le code du programme :
Exemple : Le flux de contrôle et le flux de données
 - Les sorties générées :
Exemple : HTML/XML

ANALYSE DYNAMIQUE

- Exécute le code logiciel,
- Trouve des défaillances.

2.4 Analyse statique

Buts de l'analyse statique:

- La détection très tôt de défauts avant l'exécution des tests,
- Une information très tôt sur certains aspects suspects du code ou de la conception, par le calcul de métriques, par exemple une mesure de complexité élevée,
- L'identification de défauts difficilement détectables par des tests dynamiques,
- La détection de dépendances et d'inconsistances dans les modèles logiciels tels que des liens dans les modèles logiciels,
- L'amélioration de la maintenabilité du code et de la conception,
- La prévention des défauts, si les leçons sont prises en compte lors du développement.

2.4 Analyse statique

Défauts découverts par l'analyse statique:

- La détection très tôt de défauts avant l'exécution des tests,
- Une information très tôt sur certains aspects suspects du code ou de la conception, par le calcul de métriques, par exemple une mesure de complexité élevée,
- L'identification de défauts difficilement détectables par des tests dynamiques,
- La détection de dépendances et d'inconsistances dans les modèles logiciels tels que des liens dans les modèles logiciels,
- L'amélioration de la maintenabilité du code et de la conception,
- La prévention des défauts, si les leçons sont prises en compte lors du développement.

2.4 Analyse statique

Défauts découverts par l'analyse statique:

CODE MORT

- Code qui n'a pas de raison d'être, rend la lecture du code source plus complexe et déconcentre les responsables de la maintenance.

BOUCLES INFINIES

- Une boucle dont la condition de sortie n'a pas été définie ou ne peut pas être satisfaite. En conséquence, la boucle ne peut se terminer qu'à l'interruption du programme qui l'utilise.

Exemple :

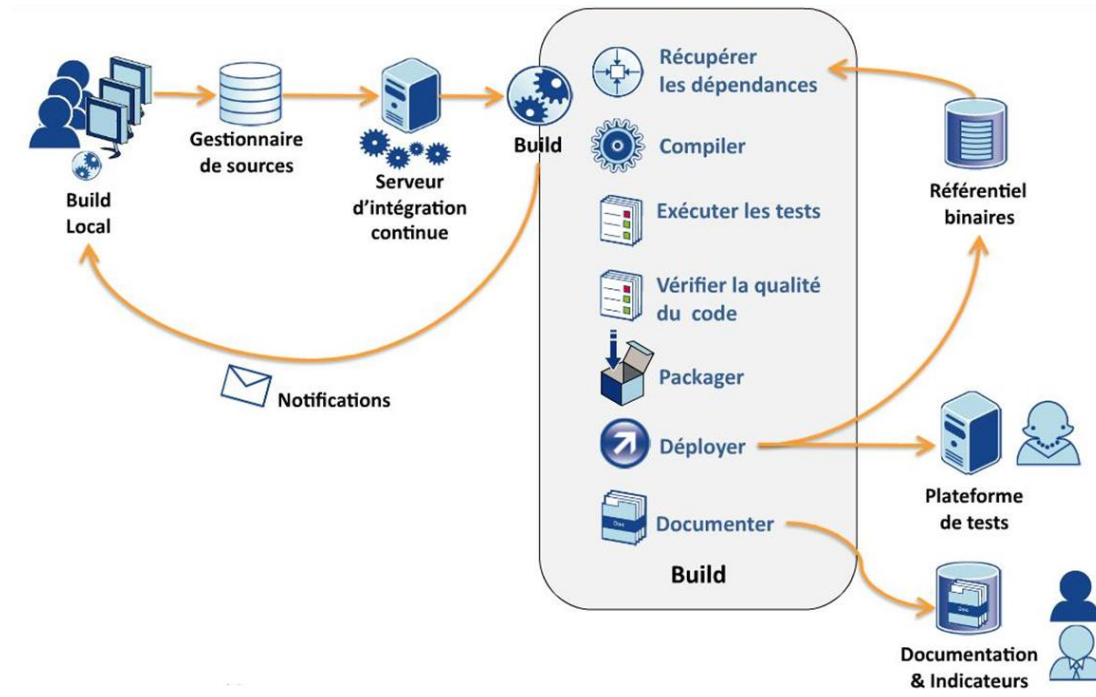
```
int a = 0;  
int i = 0;  
while (i < 1)  
    { a++; }
```

« i » restera toujours « 0 », donc la boucle ne va pas s'arrêter.

2.5 Déploiement

Le déploiement consiste en l'installation du package dans un environnement

→ Il est généralement automatisé





3. Jenkins

3.1 Introduction

3.2 Installation

3.3 Administration

2.4 Analyse statique

2.5 Déploiement

3.1 Jenkins: Introduction



Projet open source créé en 2004

- Projet open-source développé en javascript
- Anciennement Hudson
- En 2011, naissance de Jenkins

Configuration user-friendly

- Une interface Web pour la gestion de l'intégration continu
- Configuration totalement UI
- Des plugins pour gérer les interactions avec les autres outils

3.1 Jenkins: Introduction



Très large communauté

- Intégré dans le paysage logiciel de la plupart des entreprises
- Solution supportée et récompensée
 - Bossie Awards
 - Geek Choice
 - DevOps & SCM
 -

CAPITA



ebay



GitHub

LinkedIn

3.1 Jenkins: Introduction



Dashboard

- Aperçu des différents builds
- Pilotage des builds
- Information de la tendance avec l'indicateur « Météo »

The screenshot shows the Jenkins dashboard interface. At the top, there's a header with the Jenkins logo, a search bar, and the user name 'Joshua Wyatt Smith'. Below the header, there's a sidebar on the left with navigation links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', 'My Views', 'Job Config History', and 'Job Priorities'. The main area displays a table of builds, filtered by 'All'. The table has columns for status (S), weather icon (W), name, last success, last failure, and last duration. The builds listed include 'BuildCernVMKernel', 'BuildKernel', 'CernvmSetupBuildEnvironment', 'Check-VecGeom-AVX', 'Check-VecGeom-scalar', 'cleanupnodes', 'cling-generic-build', and 'cling-periodic'. Each build row includes a status icon (green for success, red for failure, grey for in progress) and a weather icon (sun for success, cloud for failure, rain for in progress). The 'Build Queue (26)' section is visible at the bottom left, showing a list of builds in the queue with status icons.

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☁	BuildCernVMKernel	6 mo 19 days - #3	6 mo 19 days - #1	14 min
●	☀	BuildKernel	N/A	N/A	N/A
●	☀	CernvmSetupBuildEnvironment	3 hr 12 min - #10283	17 days - #8484	1 sec
●	☁	Check-VecGeom-AVX	8 days 20 hr - #87	20 hr - #95	8 min 41 sec
●	☁	Check-VecGeom-scalar	8 days 19 hr - #89	19 hr - #97	3 hr 7 min
●	☁	cleanupnodes	N/A	10 mo - #3	2.2 sec
●	☁	cling-generic-build	15 days - #174	4 days 1 hr - #207	18 min
●	☁	cling-periodic	3 mo 18 days - #1498	36 min - #2617	1 min 10 sec

3.1 Jenkins: Introduction



Job Jenkins

→ Traitement Jenkins des sources d'un projet

☒ Git
Repositories

Project name:

Description:

Repository URL:

Credentials:

☒ Build periodically
Schedule:

Would last have run at Monday, January 25, 2016 2:05:48 AM UTC; would nex Tuesday, January 26, 2016 2:05:48 AM UTC.

String Parameter

Name:

Default Value:

Description:

Execute shell

Command

```
#!/bin/bash -x
mkdir build ; cd build
cmake -DCMAKE_INSTALL_PREFIX=../install ../lcgcmake
make -j12 -k
error_value=$?
####lcgcmake/cdash/isDone.sh 1
if [ $error_value -eq 0 ]
then
    echo "Successfully build"
    lcgcmake/cdash/isDone.sh 1
    exit 0
else
    echo "THE BUILD HAS FAILED *****"
    lcgcmake/cdash/isDone.sh 2
    exit 1
fi
cd ..
rm -rf build
rm -rf install
```

3.2 Jenkins: Installation

Installation

- Téléchargement à partir du site: <https://jenkins.io/>
- Récupérer la dernière release stable (Long Term Support)
- Tous les principaux OS sont supportés
- Pré-requis: java

Type

- Installation de type WAR
- Installation par package pour Windows/Linux/Mac
- Installation par Docker



3.2 Jenkins: Installation

Démarrage du war

→ En ligne de commande: **java -jar jenkins.war**

Lancement initial

→ Web: <http://localhost:8080>

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.

Installation en cours...

Installation en cours...

<input type="checkbox"/> Folders	<input type="checkbox"/> OWASP Markup Formatter	<input type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding
<input type="checkbox"/> Timestampers	<input type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input type="checkbox"/> Gradle
<input type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Pipeline: Stage View
<input type="checkbox"/> Git	<input type="checkbox"/> Subversion	<input type="checkbox"/> SSH Slaves	<input type="checkbox"/> Matrix Authorization Strategy
<input type="checkbox"/> PAM Authentication	<input type="checkbox"/> LDAP	<input type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer

3.2 Jenkins: Installation

Plugin par défaut

Installation en cours...

Installation en cours...

<input type="radio"/> Folders	<input type="radio"/> OWASP Markup Formatter	<input type="radio"/> Build Timeout	<input type="radio"/> Credentials Binding
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup	<input type="radio"/> Ant	<input type="radio"/> Gradle
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Branch Source	<input type="radio"/> Pipeline: GitHub Groovy Libraries	<input type="radio"/> Pipeline: Stage View
<input type="radio"/> Git	<input type="radio"/> Subversion	<input type="radio"/> SSH Slaves	<input type="radio"/> Matrix Authorization Strategy
<input type="radio"/> PAM Authentication	<input type="radio"/> LDAP	<input type="radio"/> Email Extension	<input type="radio"/> Mailer

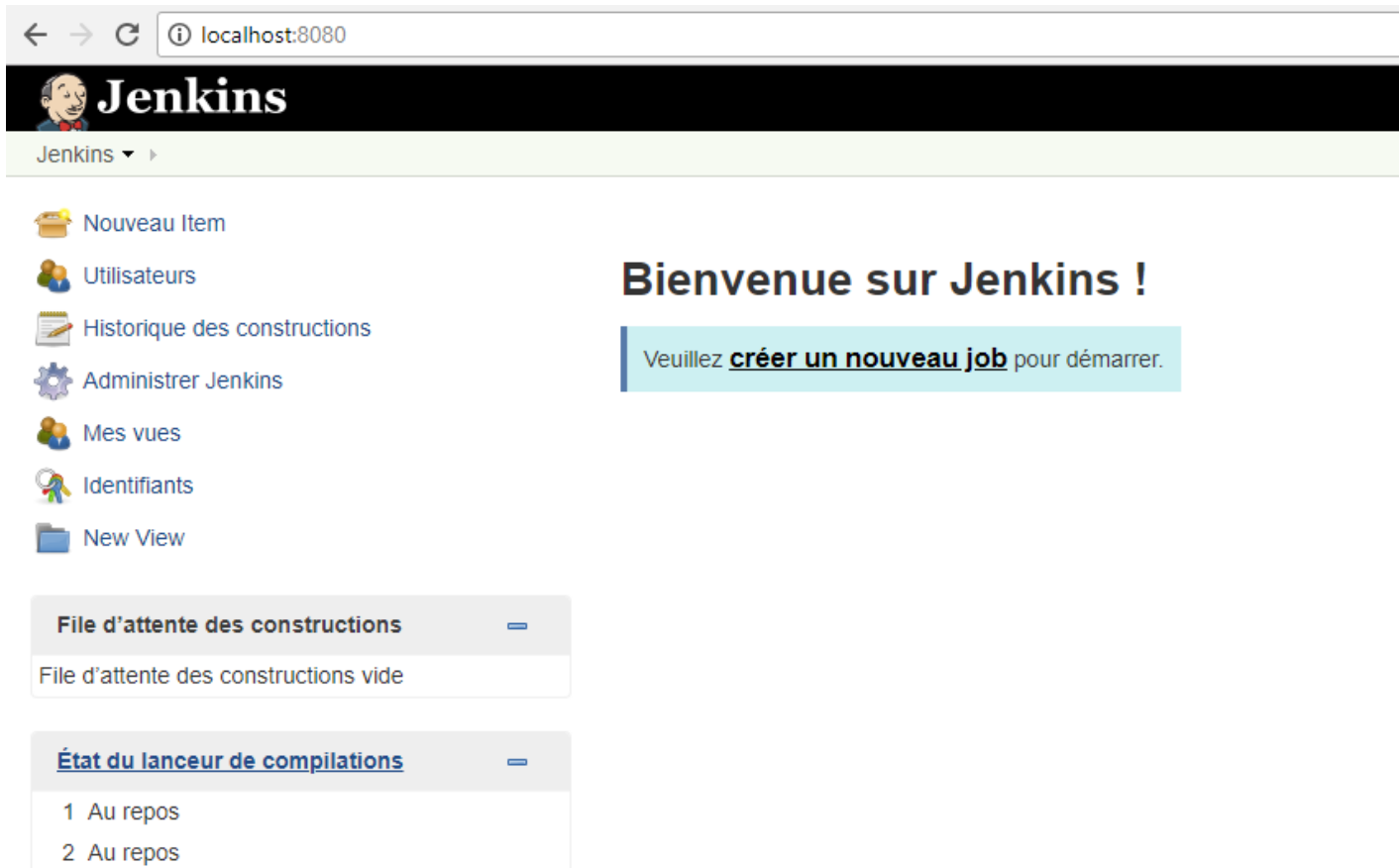
3.2 Jenkins: Installation

Création administrateur

Créer le 1er utilisateur Administrateur

Nom d'utilisateur:	<input type="text" value="admin"/>
Mot de passe:	<input type="password" value="....."/>
Confirmation du mot de passe:	<input type="password" value="....."/>
Nom complet:	<input type="text" value="Administrateur"/>
Adresse courriel:	<input type="text" value="admin@jenkins.fr"/>

3.2 Jenkins: Installation



The screenshot shows the Jenkins web interface in a browser window. The address bar displays 'localhost:8080'. The Jenkins logo and name are at the top. A navigation menu on the left includes links for 'Nouveau Item', 'Utilisateurs', 'Historique des constructions', 'Administrer Jenkins', 'Mes vues', 'Identifiants', and 'New View'. The main content area features a large 'Bienvenue sur Jenkins !' message with a light blue box containing the instruction: 'Veuillez **créer un nouveau job** pour démarrer.' Below this, there are two expandable sections: 'File d'attente des constructions' (currently collapsed) and 'État du lanceur de compilations' (expanded, showing a list of 1 'Au repos' and 2 'Au repos' items).

← → ↻ ⓘ localhost:8080

Jenkins

Jenkins ▾ ▶

- Nouveau Item
- Utilisateurs
- Historique des constructions
- Administrer Jenkins
- Mes vues
- Identifiants
- New View

Bienvenue sur Jenkins !

Veuillez **créer un nouveau job** pour démarrer.

File d'attente des constructions —

File d'attente des constructions vide

État du lanceur de compilations —

- 1 Au repos
- 2 Au repos

3.3 Jenkins: Administration

Administrer Jenkins



Configurer le système

Configurer les paramètres généraux et les chemins de fichiers.



Configurer la sécurité globale

Sécuriser Jenkins; définir qui est autorisé à accéder au système.



Configure Credentials

Configure the credential providers and types



Configuration globale des outils

Configurer les outils, leur localisation et les installeurs automatiques.



Recharger la configuration à partir du disque

Supprimer toutes les données en mémoire et recharger tout à partir du système de fichiers. Utile quand vous modifiez les fichiers de configuration directement sur le disque.



Gestion des plugins

Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.



Informations sur le système

Affiche diverses informations relatives au système pour aider à la résolution de problèmes.



Logs systèmes

Le log système capture la sortie `java.util.logging` relative à Jenkins.



Statistiques d'utilisation

Vérifiez l'utilisation des ressources et décidez si vous avez besoin d'ordinateurs supplémentaires pour vos builds.



Jenkins CLI

Accéder ou gérer Jenkins depuis votre shell ou depuis votre script.



Console de script

Exécute des scripts arbitraires pour l'administration, la résolution de problèmes ou pour un diagnostic.



Gérer les nœuds

Ajouter, supprimer, contrôler et monitorer les divers nœuds que Jenkins utilise pour exécuter les jobs.



A propos de Jenkins

Afficher les informations de version et de licence



Gérer les anciennes données

Nettoyer les fichiers de configuration des restes de vieux plugins et des versions antérieures.



Gérer les utilisateurs

Créer/supprimer/modifier les utilisateurs qui peuvent se logger sur ce serveur Jenkins



Préparer à la fermeture

Cesser d'exécuter de nouveaux builds, afin que le système puisse se fermer.

3.3 Jenkins: Administration

Configurer le système

- Configurer les paramètres globaux du serveur jenkins
- Configurer les accès aux différents types de serveur SCM
- Configurer le serveur SMTP pour les notification Mail

Configurer la sécurité globale

- Définir les types de connexion sécurisée à l'application
- Définir les autorisations aux différentes fonctionnalités

Gérer les utilisateurs

- Administrer les utilisateurs jenkins

Utilisateurs

Ces utilisateurs peuvent se logger sur Jenkins. C'est le groupe contenant [cette liste](#), qui contient également les utilisateurs créés automatiquement qui ont simplement fait des commits sur certains projets et n'ont pas d'accès direct à Jenkins.

User Id	Nom
 admin	Administrateur



Configurer la sécurité globale

☒ Activer la sécurité

Disable remember me ☐

Contrôle de l'accès

Royaume pour la sécurité (Realm)

☒ Base de données des utilisateurs de Jenkins

☐ Autoriser les utilisateurs à s'inscrire

☐ Déléguer au conteneur de servlets

☐ LDAP

Autorisations

☒ Les utilisateurs connectés peuvent tout faire

☐ Allow anonymous read access

☐ Mode legacy

☐ Stratégie d'autorisation matricielle basée sur les projets

☐ Sécurité basée sur une matrice

☐ Tout le monde a accès à toutes les fonctionnalités

3.3 Jenkins: Administration

Configuration globale des outils

- Configurer les principaux outils
- JDK
- Maven
- Ant
- Git
- Gradle
- Docker



3.3 Jenkins: Administration

Gestion des plugins

- Installation des plugins permettant de piloter les différentes étapes de l'intégration
 - Plugin de compilation
 - Plugin d'exécution et de reporting de test
 - Plugin de déploiement
 - Plugin de gestion de package
 - Plugin de notification
 - Plugin d'intégration avec des outils
 -

Gestion des plugins

our au tableau de bord

ministrer Jenkins

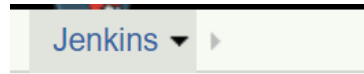
late Center

Filtre:

Mises à jour Disponibles Installés Avancé

Installer ↓	Nom	Version
<input type="checkbox"/>	Selenium Capability Axis Axis for current capabilities of a Selenium Server	0.0.6
<input type="checkbox"/>	Selenium Auto Exec Server(AES) This plugin is a Selenium Auto Exec Server(AES) (http://www.enjoyxstudy.com/selenium/autoexec/) plugin.	0.5
<input type="checkbox"/>	Selenium HTML report This is an Jenkins plugin to visualize the results of selenium tests	1.0
<input type="checkbox"/>	TestingBot This plugin integrates videos/screenshots of your TestingBot.com Selenium tests	1.13
<input type="checkbox"/>	Nervvana Nervvana is a robust cloud-based Selenium testing solution. Nervvana Plugin is a software which enables continuous integration with Nervvana cloud	1.02.06
<input type="checkbox"/>	Selenium Builder	1.14

3.4 Créer un job



Création d'un job:  [Nouveau Item](#)

Saisissez un nom

» Champ obligatoire



Construire un projet free-style

Ceci est la fonction principale de Jenkins qui sert à build (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.



Pipeline

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



Construire un projet multi-configuration

Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, 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.

OK

3.4 Créer un job

→ Informations générales du build


General Gestion de code source Ce qui déclenche le build Environnements de Build Build Actions à la suite du build

Nom du Projet

Formation


Description


[Plain text] [Prévisualisation](#)


☒ Ce build a des paramètres 


Ajouter un paramètre ▾

☐ GitHub project

☐ Supprimer les anciens builds 

☐ Throttle builds 

☐ Désactiver le projet 

☐ Exécuter des builds simultanément si nécessaire 

Avancé...

3.4 Créer un job

→ Gestion de code source: Configuration du dépôt à récupérer

3.5 Maven

Enjeux

- Simplifier et uniformiser le processus de build

Histoire

- Mot Yiddish: « accumulateur de connaissance »
- Né en 2001

Objectifs

- Définir un standard de build d'un projet java
- Documenter le projet
- Partager les librairies entre les projets
- Eviter de stocker dans les outils SCM les librairies

3.5 Maven

Un projet Maven

- Fichiers du code sources
- Fichiers de configuration
- Licences
- Fichiers de ressources
- Dépendances

POM: Project Object Model

- Fichier Maven
- Normalise et décrit le projet

```
pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4
5   <groupId>org.example</groupId>
6   <artifactId>jpademo</artifactId>
7   <version>1.0</version>
8   <packaging>jar</packaging>
9
10  <scm>
11    <connection>scm:git:ssh://my.git.server.internal/home/git/jpademo</connection>
12    <developerConnection>scm:git:ssh://my.git.server.internal/home/git/jpademo</developerConnection>
13  </scm>
14  <ciManagement>
15    <system>jenkins</system>
16    <url>https://my.jenkins.internal/jenkins</url>
17  </ciManagement>
18
19  <name>jpademo</name>
20  <url>http://maven.apache.org</url>
21  <build>
22    <plugins>
23      <plugin>
24        <groupId>org.apache.maven.plugins</groupId>
25        <artifactId>maven-compiler-plugin</artifactId>
26        <version>2.3.2</version>
27        <configuration>
28          <source>1.6</source>
29          <target>1.6</target>
30        </configuration>
31      </plugin>
32    </plugins>
33
34    <plugin>
35      <groupId>org.apache.maven.plugins</groupId>
36      <artifactId>maven-jar-plugin</artifactId>
37      <version>2.2</version>
```

3.5 Maven

POM

- groupId
- artifactId
- packaging
- version
- Name
- url
- description
- dependencies
- plugin
- build



4. Bonnes pratiques

4. Bonnes pratiques



Détecter et résoudre tôt les défauts

- On commit que du code qui marche
 - Exécution de build en local par le développeur
 - Commenter chaque commit
 - Découper en tâches les activités de codage
- Tester le build
 - Favoriser les approches xDD
 - Intégrer la couverture de code par les tests
- Intégrer après chaque commit
 - Attention, le temps de build ne doit pas dépasser 10min
 - Distinguer les différents types de tests unitaires/bdd, fonctionnels
 - Exécuter une intégration complète au moins une fois par jour
 - Corriger immédiatement les builds en échecs

4. Bonnes pratiques



Sécuriser le build

- Automatiser au maximum le build
- Disposer d'un environnement fiable et propre
 - Privilégier les checkout
 - Nettoyer souvent les dépôts maven
 - Intégrer sur des plateformes distinctes

4. Bonnes pratiques



La qualité du code, le meilleur garde-fou

- Mise en place de l'analyse du code le plus tôt possible
- Définir des objectifs raisonnables et incrémentales
- Automatiser avec des outils tels que Sonar, Cobertura



5. Travaux pratiques

- 5.1 Installation de Jenkins/Maven/Git
- 5.2 Création d'un job pour lancer des scripts Selenium
- 5.3 Création d'un job pour lancer des scripts SoapUI
- 5.4 Création d'un job pour lancer des scripts Cucumber
- 5.5 Sonar