

# Introduction à l'intégration continue

AUTOM

# Pré-requis du module

---

- Connaitre Shell et Linux
- Savoir utiliser une machine virtuelle
- Connaitre méthode agile



# Objectifs du module

- Présentation de l'intégration continue
- Prise en main de Jenkins
  - Savoir comment se configure Jenkins (authentification, plugins)
  - Savoir créer ou dupliquer des jobs et les ordonnancer
  - Savoir configurer des jobs
    - Ajout de tâches Maven (run tests Squash TA)
    - Ajout de tâches de script
    - Lien avec un gestionnaire de source
    - Lien avec un gestionnaire d'artefact
- Découverte de Maven
  - Principe (pom.xml)
  - Commandes
- Découverte de SVN
  - Principe (branches, commits, tags)
  - Opérations basiques (tirer, commiter, tagger, créer une nouvelle branche)
- Découverte de Nexus
  - Principe
  - Ajouter des dépendances



# Introduction :

## Présentation de l'intégration continue (qualité logicielle)



- **L'intégration continue** est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.
- L'intégration continue est une organisation de travail
- L'intégration continue de test permet au testeur de vérifier la qualité d'une application (qualimétrie, tests de non-régression...) et le respect des normes et bonnes pratiques dans le code (audit de code, qualimétrie, ...)



- Retour sur les différents cycles utilisés pour le développement applicatif
- Présentation de l'intégration continue
- Légende utilisée dans la description des schémas ci-dessous

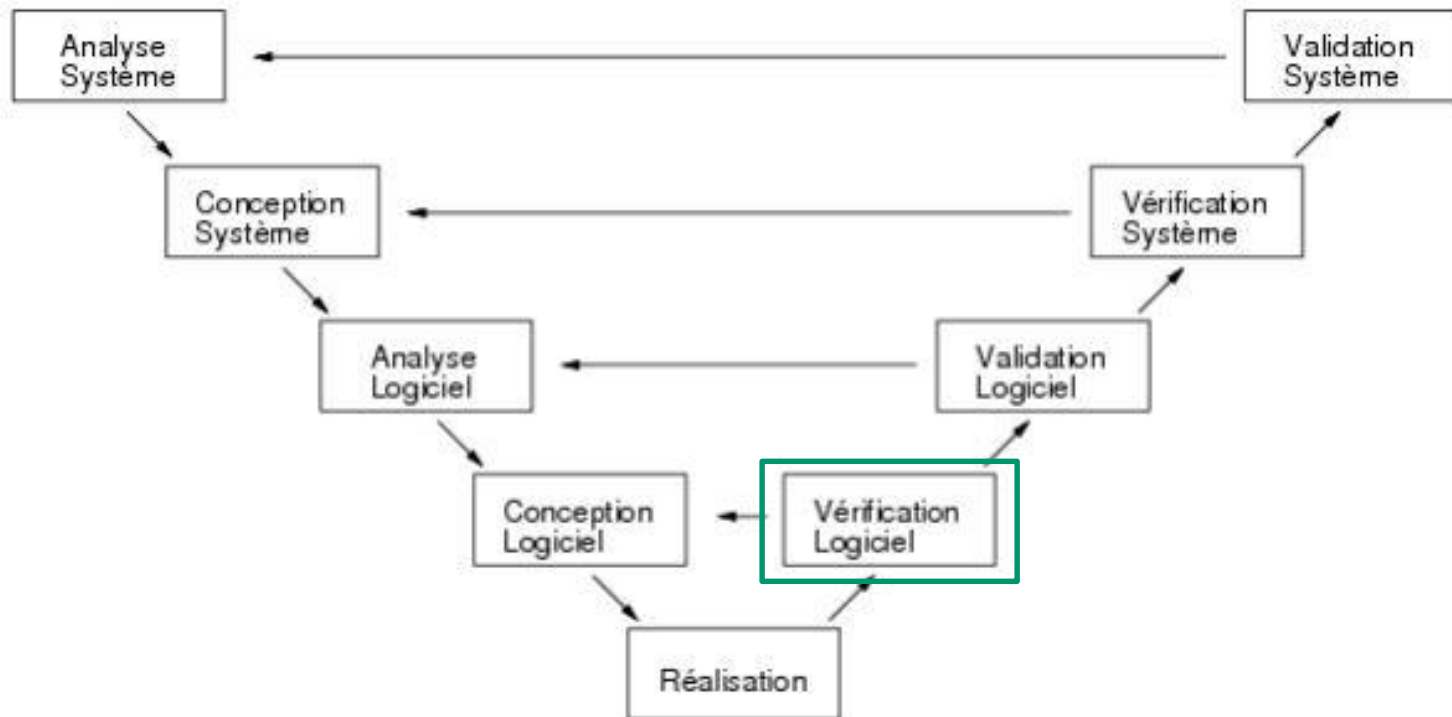


Présence des testeurs dans le processus de création d'une application



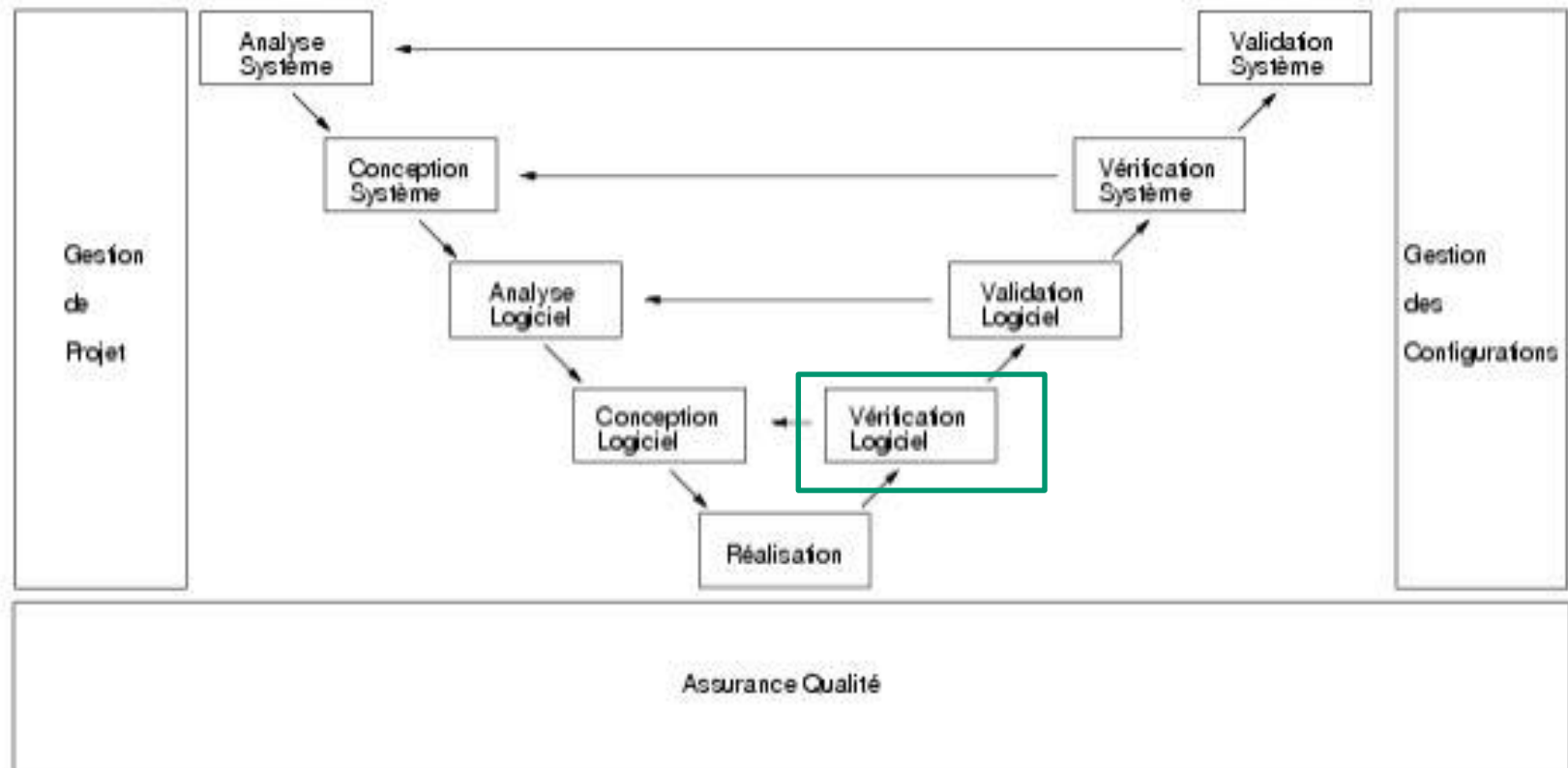
- Cycle en V

### Cycles de développement en V



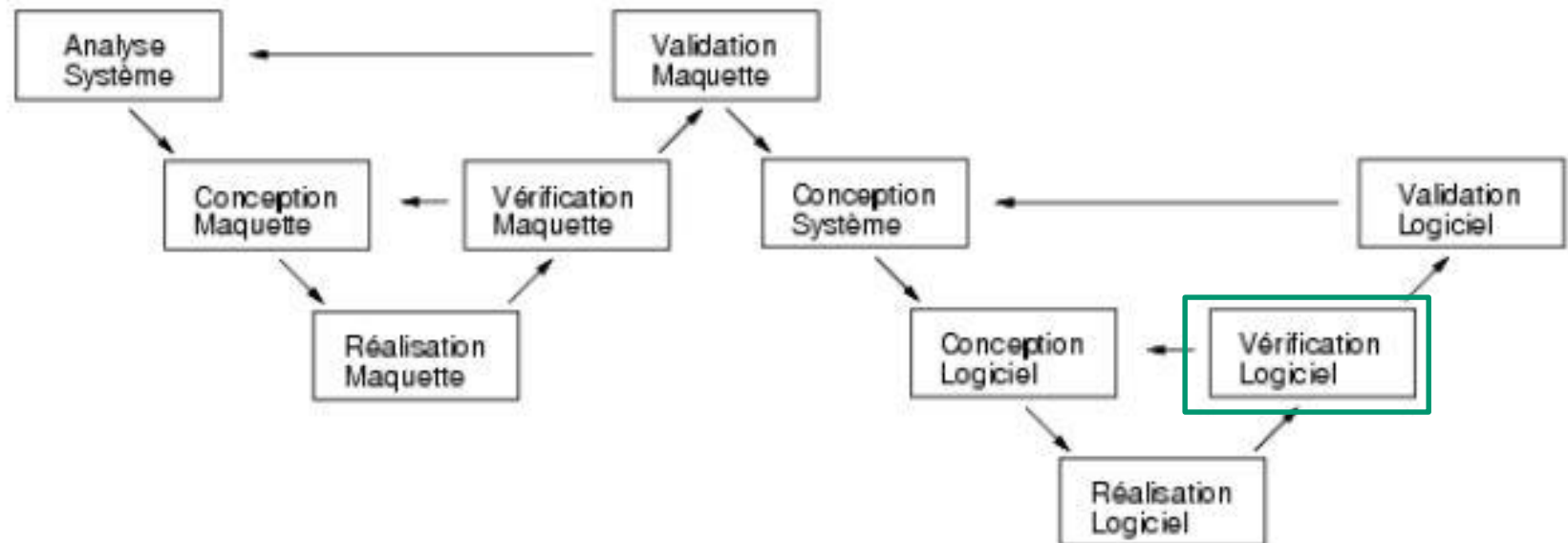
- Cycle en M

# Cycles de développement en M





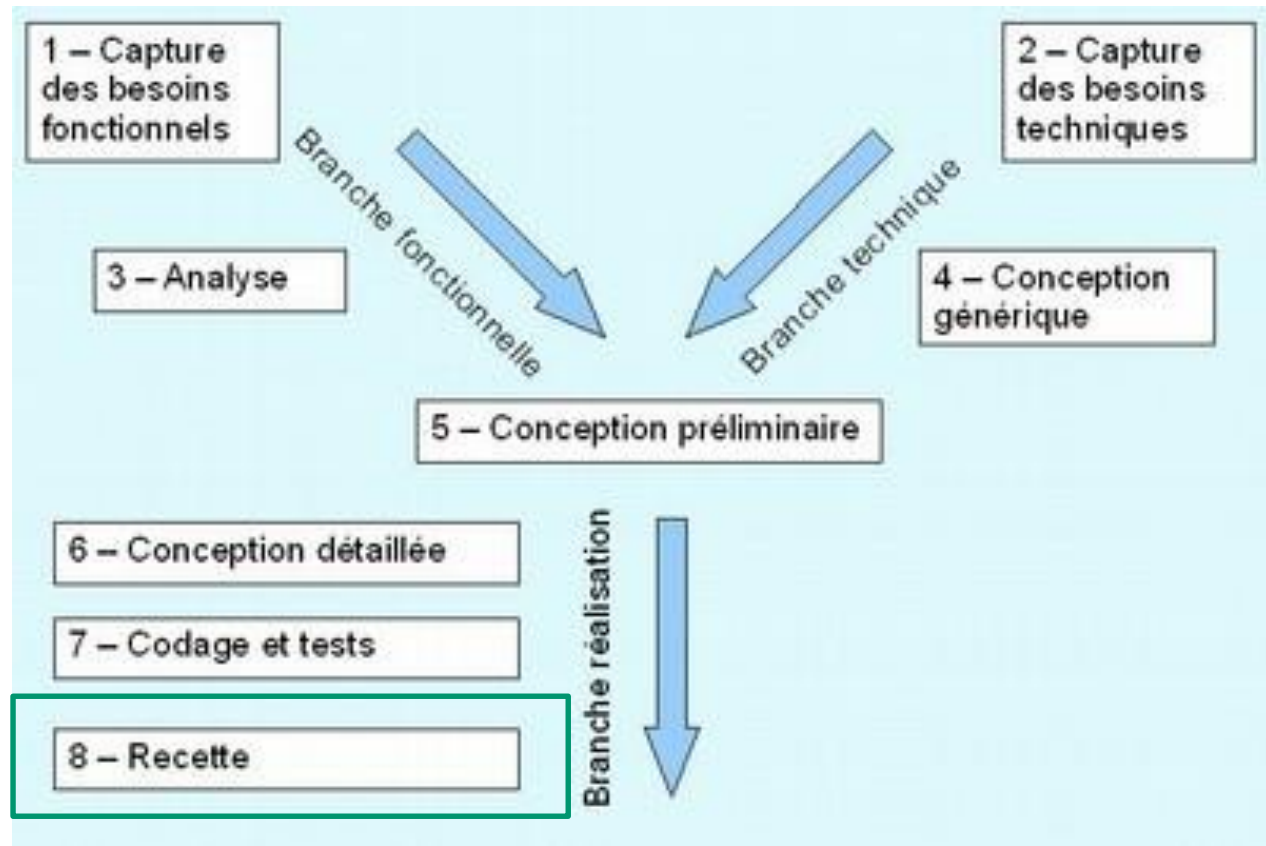
- # Cycles de développement en W



# Introduction à l'intégration continue

## Introduction : retour sur les différents cycles de développement (4/6)

- Cycle en Y



- Cycle en Y, les différentes phases :
  - **Phase 1** : capture des besoins fonctionnels et des besoins techniques. Modélisation très haut niveau et rédaction d'un cahier des charges techniques et fonctionnelles
  - **Phase 2** : analyse et conception technique. Recherche de solutions d'architecture et de modèles de conceptions génériques
  - **Phase 3** : conception préliminaire, conception détaillée, codage et tests, recette et livraison



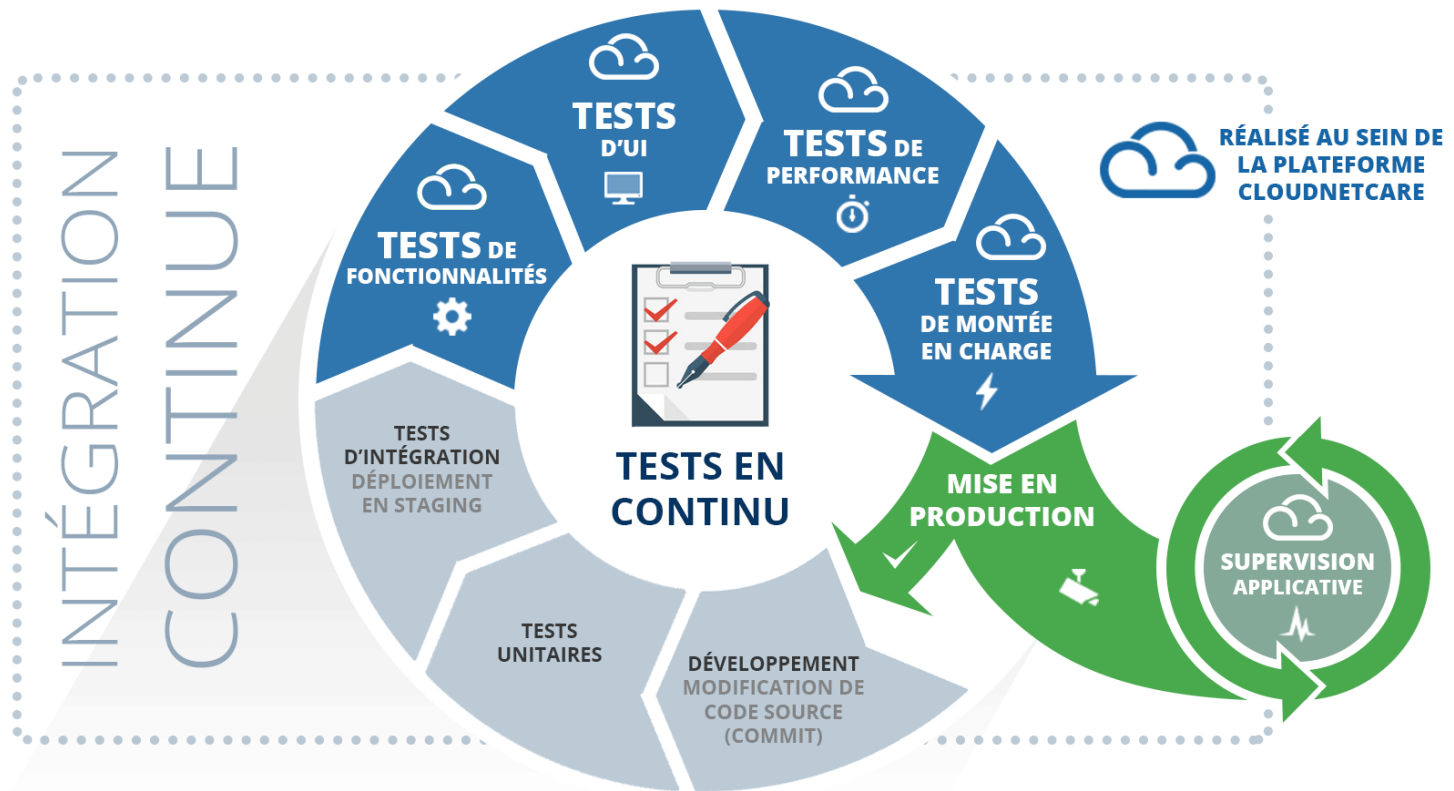
- Limites des cycles traditionnels :
  - Travail souvent dispersé, long.
  - Source de conflits au sein des équipes (MOA, MOE, TRA...)
  - Reprise de code(s) ancien(s), parfois obsolète(s)
  - Prise de retard dans les plannings



# Introduction à l'intégration continue

## Introduction : l'intégration continue (1/5)

- L'intégration continue permet de repenser tout le processus de développement d'une application.
- Les testeurs sont présents à tous les niveaux du processus



- L'intégration continue dans sa forme simple :
  - Se compose d'outils qui surveillent les modifications de code dans le gestionnaire de configuration (Jenkins et ses plugins)
  - Dès qu'un changement est détecté, cet outil compile (Maven) et teste l'application (Squash TA et autres).
  - Si la moindre erreur arrive alors l'outil alerte les développeurs afin qu'ils puissent tout de suite corriger le problème.
- L'intégration continue dans sa forme complexe :
  - L'Intégration Continue peut aussi suivre la santé de votre projet en surveillant la qualité du code et les métriques de couverture et ainsi aider à maintenir la dette technique à un niveau bas et à abaisser les coûts de maintenance (qualimétrie, Sonar)
  - Créer un reporting public (client, développeurs, MOA & AMOA)
  - Facilite la communication entre les acteurs
  - Indique un état clair du développement d'une application
  - Permet le déploiement automatique d'une application
  - Permet de tester la non régression d'une application

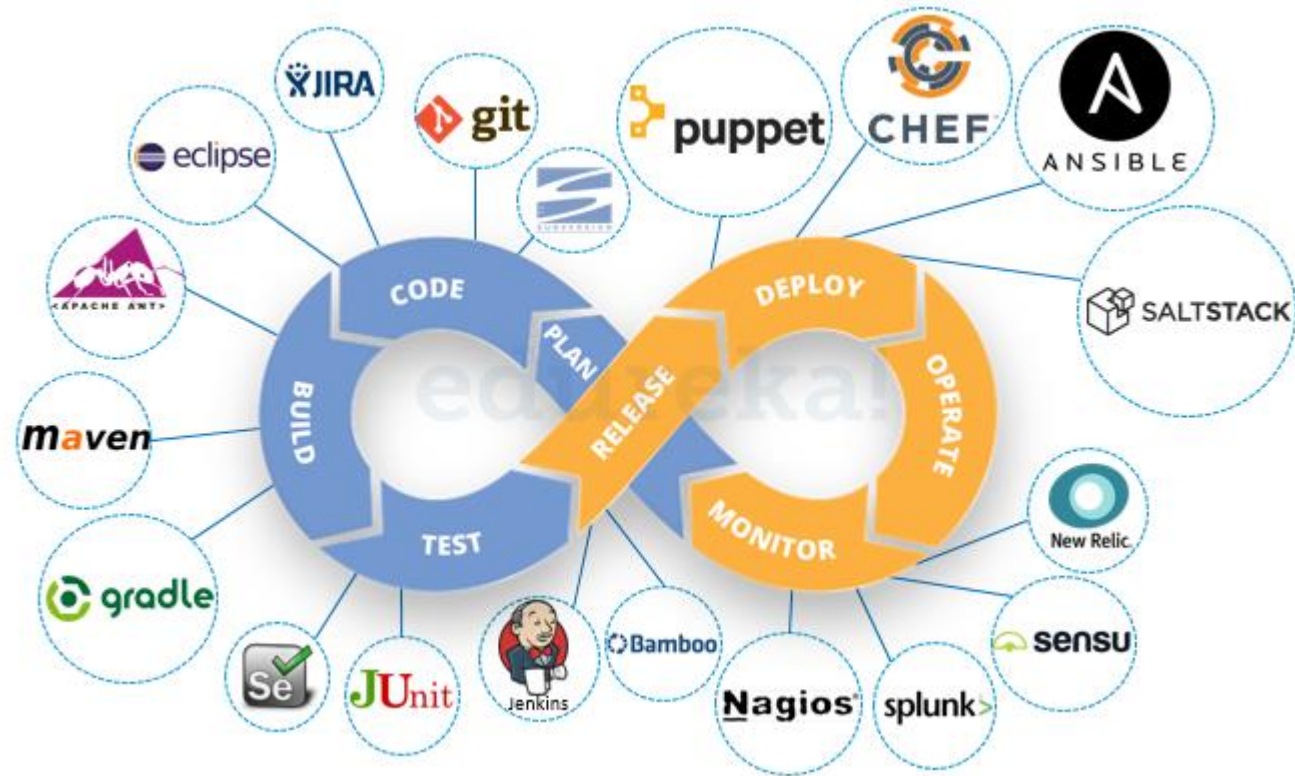
- Synthèse avantages de l'intégration continue :
  - Travail d'équipe et réactivité
  - Assurer la qualité du code au plus tôt
  - Assurer la complétude des livrables (intégrité des livrables)
  - Rend la production d'une application plus « fluide »
  - Permet de générer du reporting de qualité
  - Mise en place d'une méthode de travail DevOps





# Introduction à l'intégration continue

## Introduction : l'intégration continue (4/5)



Source : edureka!



- La conduite du changement, une étape à prendre en compte!
- Place de l'intégration continue dans les cycles de développement V, M, W et Y?



# Introduction à l'intégration continue

## Introduction : L'intégration continue dans l'entreprise

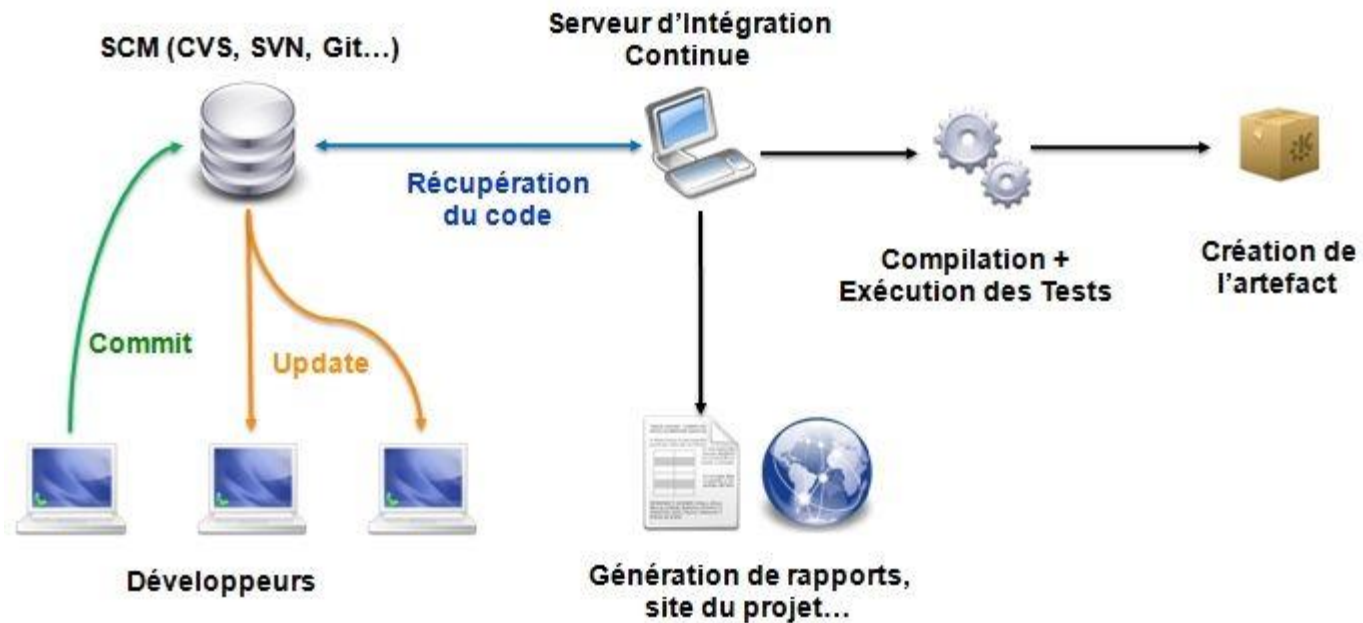
- **Phase 1** → Pas de serveur de build
- **Phase 2** → Builds quotidiens
- **Phase 3** → Builds quotidiens et tests automatisés basiques
- **Phase 4** → Arrivée des métriques
- **Phase 5** → Les tests, une phase longtemps moquée
- **Phase 6** → Tests d'acceptation
- **Phase 7** → Le déploiement continue



# Introduction à l'intégration continue

## Introduction : Phase 1 → Pas de serveur de build

- **Phase 1** → Pas de serveur de build → à mettre en place (développeur)
  - Il s'agit principalement du travail du développeur
  - Le testeur peut participer à la création d'un serveur de build



- **Phase 2 → Builds quotidiens**
  - Le serveur de build existe
  - Les développeurs « commit » le code tous les jours (dans le meilleur des cas)
  - Le code est « buildé » et testé (de préférence la nuit)



# Introduction à l'intégration continue

## Introduction : Phase 3 → Builds quotidiens et tests automatisés basiques

- **Phase 3 → Builds quotidiens et tests automatisés basiques**
  - Dès qu'un nouveau code est « commité », des tests automatisés sont exécutés (non régression).
  - Exécution également de tests unitaires (développeurs)
  - Les erreurs sont remontées immédiatement
  - Les développeurs corrigent le code
  - Le client suit en temps réel l'avancée d'amélioration de la qualité du code
  - Suivi reporting par email et tableau de bord



- **Phase 4 → Arrivée des métriques**
  - Des métriques de qualité et de couverture de code sont maintenant mesurées pour aider à évaluer la qualité du code (qualimétrie)
  - Le build de métrique de qualité du code génère automatique l'API de l'application (API Application Programming Interface)
  - Oblige un respect des bonnes pratiques (dans la mesure du possible)
  - Montage d'un tableau de bord de suivi de qualimétrie (voir partie qualimétrie, module 2)



- **Phase 5 → Les tests, une phase longtemps moquée**
  - Très longtemps les tests n'ont pas été pris au sérieux.
  - Dans l'intégration continue, les tests font maintenant partie du cycle de vie
  - Différents types de tests sont utilisés et améliorent la confiance du client
  - ... améliorent aussi (et surtout) la qualité de l'application livrée
  - Le nouveau processus peut être développé → testé → déployé
  - Le déploiement permet de réaliser des tests de bout en bout, des tests de stress et des tests de performances.



- **Phase 6 → Tests d'acceptation automatisés (fitness)**
  - Signifie s'assurer que l'application est conforme aux spécifications techniques et fonctionnelles
  - Tests de validation
  - Acceptation fonctionnelle automatique et manuelle
  - Acceptation technique : tests unitaires, et tests d'intégration
  - Tests alpha et beta





# Introduction à l'intégration continue

## Introduction : Phase 7 → Le déploiement continu (1/2)

- **Phase 7 → Le déploiement continu**
  - Mise en production automatique de l'application
  - Confiance importante dans la qualité du code livré
- Distinguer livraison continue et déploiement continu :
- **Déploiement continu :**
  - Un déploiement continu correspond à la livraison d'une version applicative testée.
  - Ce processus est automatique et sous la responsabilité des techniques
  - Les utilisateurs ont accès immédiatement à la nouvelle version livrée
- **Livraison continue :**
  - Une livraison continue correspond à la livraison d'une version applicative testée.
  - Cependant, c'est le métier et non le technique qui décide du moment où les utilisateurs auront accès à la nouvelle version livrée



### Partie 1 : Présentation des outils (non exhaustive)



- Présentation sommaire des outils que nous utiliserons :
- **Jenkins** : création et gestion de jobs pour l'intégration continue. Fonctionne par exemple sur un serveur d'application Tomcat
- **Maven** : outil pour la gestion et l'automatisation de production des projets logiciels Java (semblable à l'outil ant)
- **SVN et/ou GIT** : logiciel de gestion de version
- **Nexus** : gestionnaire de dépôts open source



# Introduction à l'intégration continue

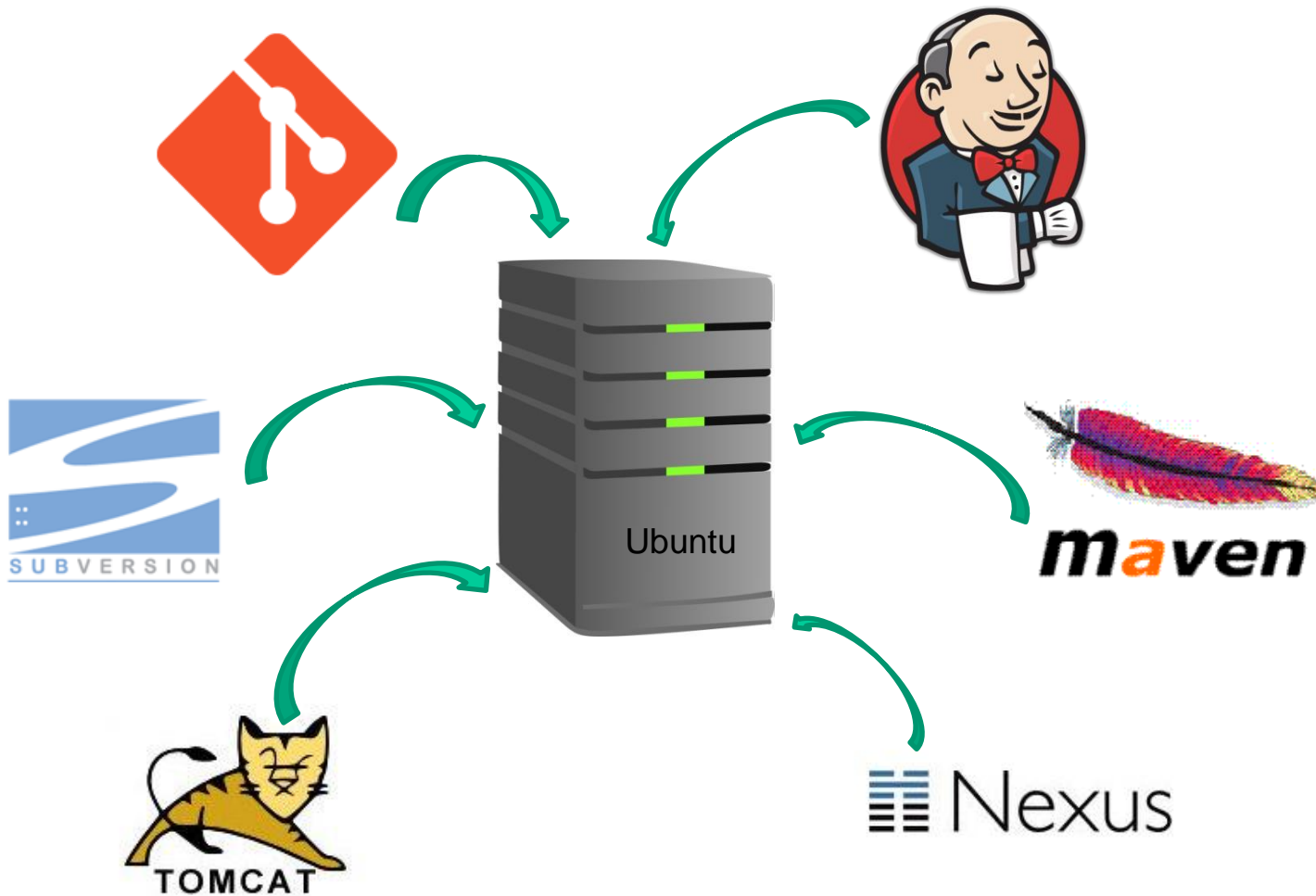
## Présentation des outils : Jenkins vs Hudson



# Installation et paramétrage des outils

## Présentation des outils

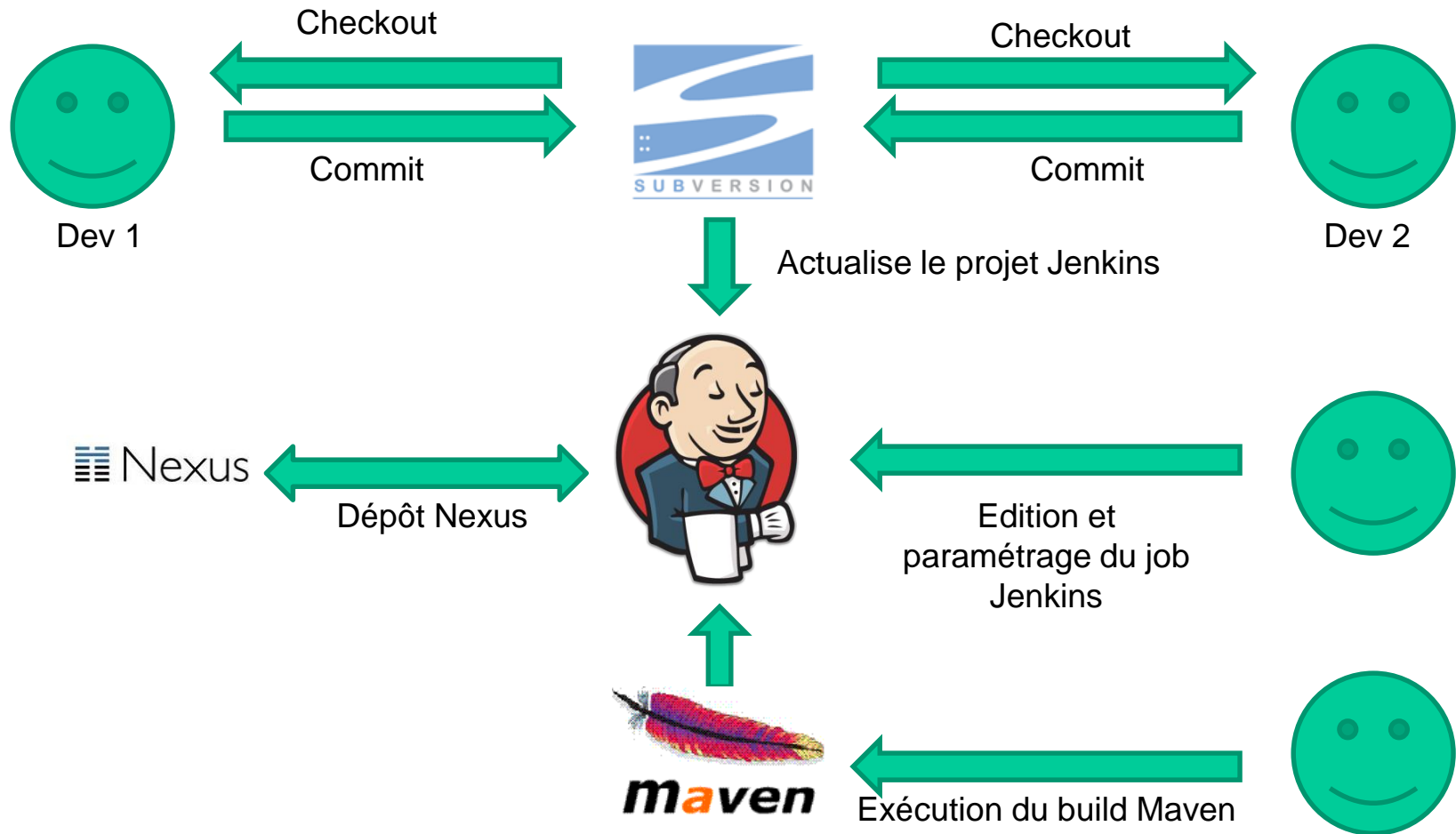
- Contenu de la machine Ubuntu



# Introduction à l'intégration continue

## Présentation des outils : schéma de fonctionnement

- Processus simplifié (la complexité varie selon le contexte)



### Partie 2 : Installation de Jenkins



- **Pourquoi utiliser Jenkins?**

- Permet de détecter au plus tôt les anomalies d'intégration
- Ces anomalies sont corrigées progressivement
- Les vérifications automatisées sous Jenkins permettent de revérifier la complétude d'une livraison à chaque fois
- Permet de connaître et suivre précisément les versions des livraisons

- **Les pré-requis nécessaires**

- Le code source doit être partagé entre les développeurs et les testeurs (svn, git...)
- Les développeurs doivent pousser régulièrement leur code pour le suivi
- Des tests automatisés doivent être mise à disposition

- **Principe de fonctionnement de Jenkins**

- Exécuter un job (un build → par exemple avec Maven) qui déroule l'ensemble des tests automatisés sur le code source livré par les développeurs.





- Les testeurs et Jenkins : cas d'utilisation
- Démonstration d'exécution d'un build maven sous Jenkins
- Rappel :
  - La bonne pratique consiste à installer chaque outil sur une plateforme virtuelle dédiée, avec un utilisateur et un groupe dédié.
  - Dans le cadre de nos exercices, tous les outils seront installés localement sur une seule et même machine virtuelle et Windows.



- **Retour sur la correction mini-projet Unix**
- **Exercice 1** : Test d'intégration et installation de Jenkins (une demie journée)
  - Reprendre une livraison d'installation de Jenkins réalisée lors du cours Unix
  - Tester la livraison complète (livrables + documentations + suivi installation manuelle + utilisation des scripts automatisés)
  - Vous devez organiser votre plateforme de test de manière à tester la procédure manuelle et automatique (chacun est libre dans la procédure de test) sans tout refaire plusieurs fois
  - Jenkins doit être ensuite utilisable
  - Vous m'enverrez par email votre retour, les anomalies trouvées.



- **Exercice 2 :** La version précédemment installée est la 2.45 de Jenkins (15 minutes).
  - Utiliser le jenkins.war 2.46 et mettre à jour votre Jenkins. Vous pourrez vérifier la version avant et après mise à jour depuis le GUI Jenkins (15 minutes)
  - Info : penser à vérifier les droits (utilisateur/groupe)
- **Exercice 3 :** Installation de Squash TA Serveur sur Windows (25 minutes)
  - Télécharger Squash TA serveur pour Windows
  - Installer Squash serveur sur votre poste Windows
  - Démarrer le et vérifier son bon fonctionnement (voir les logs)



- **Rappel général sur l'arborescence de Tomcat :**
  - Emplacement de Tomcat : selon les cas 😊
  - bin : contient les scripts Tomcat
    - catalina.sh → démarre, stop, indique si Tomcat tourne
    - daemon.sh → démarre Tomcat comme service
    - configtest.sh → vérifier la configuration système de Tomcat
    - startup.sh → démarre Tomcat
    - shutdown.sh → stop Tomcat
    - version.sh → indique la version de Tomcat
  - conf : configuration de Tomcat
    - catalina.properties → configuration de Tomcat
    - logging.properties → log4j
    - serveur.xml → configuration de votre serveur Tomcat



- **Rappel général sur l'arborescence de Tomcat :**
  - lib : contient les librairie Tomcat (ne pas toucher)
  - LICENSE : la licence Tomcat
  - logs : contient les logs Tomcat. Pour rappel, catalina.out contient les logs de lancement. Voir détail
  - NOTICE : xsd utilisées par Tomcat
  - RELEASE-NOTES : mises à jour
  - RUNNING.txt configuration pour installation Tomcat
  - temp : répertoire pour les fichiers temporaires de Tomcat
  - webapps : répertoire déploiement application
- **Exercice 5 (10 minutes) :** depuis la configuration, modifier le port Tomcat par 8082



- **Eteindre Tomcat**
  - Dans bin, lancer la commande `./shutdown.sh` puis vérifier que Tomcat est coupé `ps aux | grep tomcat`
- **Copier le fichier jenkins.war dans le répertoire webapps**
- **Lancer Tomcat et vérifier ses logs**
  - Dans bin, lancer la commande `./startup.sh && tail -f ../logs/catalina.out`
- **Vérifier que Tomcat tourne**
  - `ps aux | grep tomcat`
- **Accéder à Jenkins**
  - Dans un navigateur Web : `http://{ip_machine}:{port}/jenkins`



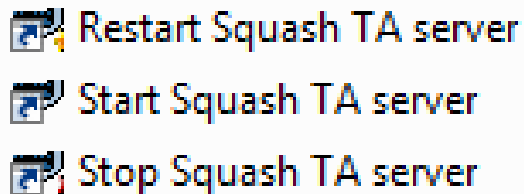
- Une fois Tomcat démarré, tout le contenu se trouve dans TOMCAT\_HOME/.jenkins
- Contenu du répertoire .jenkins : détails vu ensemble
- Le répertoire .jenkins/workspace contient les jobs créés depuis Jenkins, nous y reviendrons avec précision!



# Installation et paramétrage des outils

## Installation de Jenkins : Présentation de Squash TA serveur

- Squash TA server intègre un serveur d'application Tomcat qui héberge Jenkins
- Vous trouverez dans le Squash TA server tous les paramètres de lancement nécessaires pour le bon fonctionnement de build Maven Squash TA
- Squash TA server intègre une JDK
- Le fonctionnement et le paramétrage est identique à nos connaissances Tomcat Linux.
- Les raccourcis :





## Partie 2 : Présentation Maven



- Historique
  - Projet Apache
  - Versions maintenues : Maven 2 et Maven 3
- Principes
  - Gestion de build de projet (comme Ant , Gradle...)
  - Déclaration de build (basée sur le fichier pom.xml)
  - Métadonnées complètes sur le projet
  - Dans le test, très utilisé pour l'exécution de tests automatisés

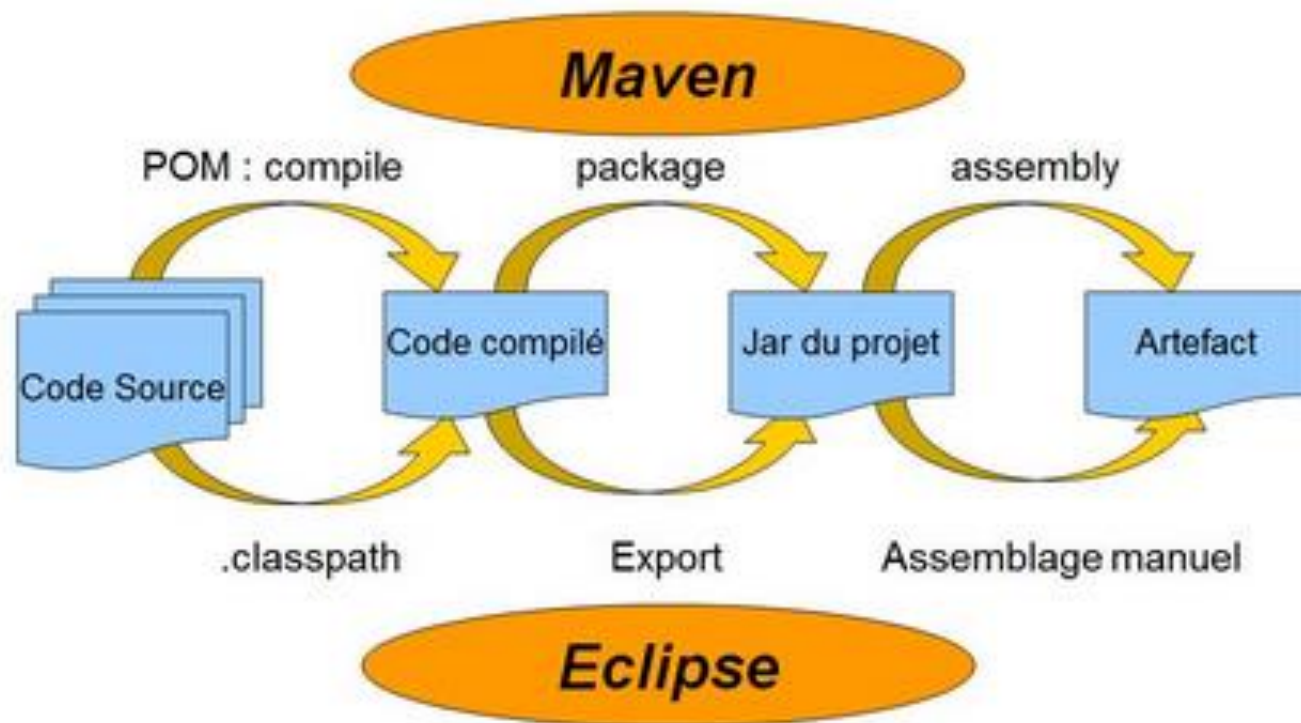


- Fonctionnalités :
  - construction , compilation
  - documentation
  - rapport
  - gestion des dépendances
  - gestion des sources
  - mise à jour de projet
  - déploiement



# Installation et paramétrage des outils

## Présentation Maven : Maven dans Eclipse

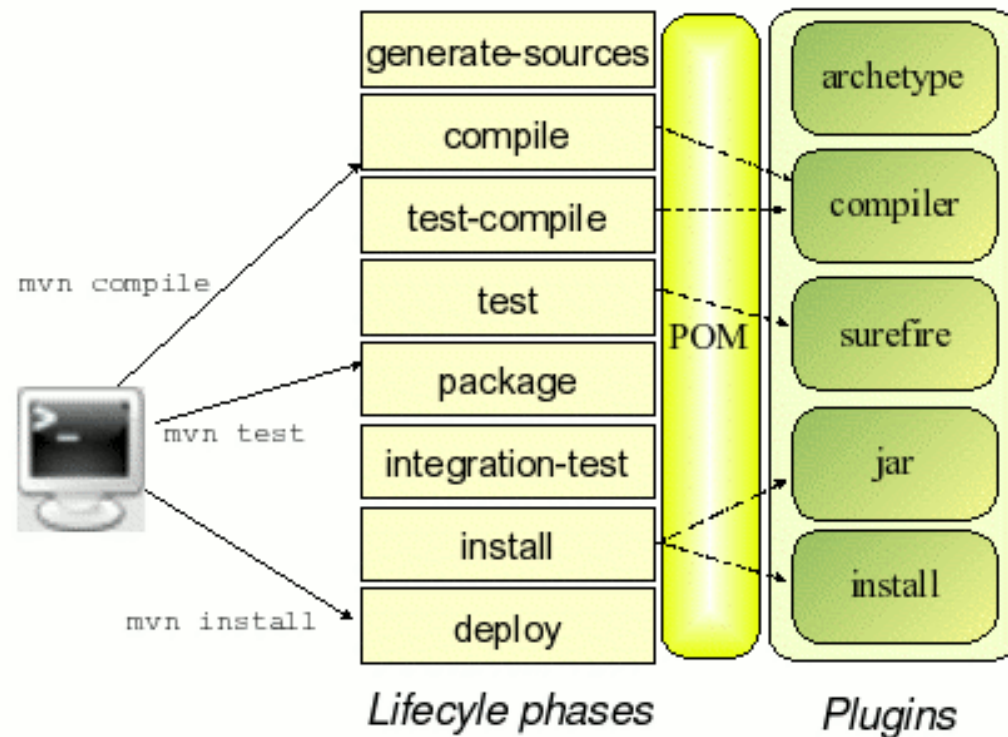


Source : developpez.com



# Installation et paramétrage des outils

## Présentation Maven : arborescence



**Figure 2. Les phases du cycle de vie Maven 2**

Source : developpez.com



- **Créer un projet.**

- La commande `archetype:generate` permet de créer un projet Maven
- C'est par exemple la commande que nous utilisons lors de la création d'un projet Maven Squash TA :
- `mvn archetype:generate -DarchetypeGroupId=org.squashtest.ta -DarchetypeArtifactId=squash-ta-project-archetype -DarchetypeVersion=1.8.0-RELEASE -DarchetypeRepository=http://repo.squashtest.org/maven2/releases`
- Tester la commande depuis cmd (copier/coller ne marche pas)
- Une fois la commande passée, l'arborescence du projet est créée, ainsi que le fichier `pom.xml`



- pom.xml :
  - pom : project object model
  - Est un fichier xml
  - Il décrit le projet et sa mise en place
- Les balises :
  - **project** : c'est la balise racine de tous les fichiers pom.xml
  - **pom.xml.modelVersion** : cette balise indique la version de POM utilisée (obligatoire)
  - **groupId** : cette balise permet d'identifier un groupe qui a créé le projet (fonction d'indexation du projet)
  - **artifactId** : nom des artefacts à construire
  - **packaging** : type de packaging du projet ( ex. : JAR, WAR, EAR, etc.).
  - **version** : version de l'artefact généré par le projet.
  - **name** : nom du projet.
  - **url** : adresse du site du projet.
  - **description** : description du projet.
  - **dependencies** : balise permettant de gérer les dépendances.
  - Plugins → gère les plugins
  - D'autres balises peuvent être utilisées selon les projets



- **Archetype** : un archetype est un template de projet. Permet de respecter les règles de bonnes pratiques.
- **Dépendances** : Une dépendance est un lien vers un artefact spécifique contenu dans un repository.
- **Artefact** : dans Maven, un artefact est un élément spécifique issu de la construction du logiciel (exemple : fichiers .jar, .ear, .zip, .war...)
- **Artifact : groupeid** : identifiant du groupe à l'origine du projet.
- **Artifact : artifactid** : l'artifactid est l'identifiant du projet au sein de ce groupe. Il est souvent utilisé comme nom final du projet.





- Autre exemple :

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/POM/4.0.0" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>fr.afcepf</groupId>
    <artifactId>Yparent</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <artifactId>YWSRest</artifactId>
  <packaging>war</packaging>

  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-api</artifactId>
    </dependency>
    <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-jaxrs</artifactId>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>${project.groupId}</groupId>
      <artifactId>YApiBusiness</artifactId>
      <version>${project.version}</version>
    </dependency>
  </dependencies>
</project>
```

- **Aide Maven** : `mvn --help`
- **Tester un projet Maven**
  - La commande `mvn test` depuis le répertoire où se trouve votre fichier `pom.xml`
- **Générer un site d'information à propos de notre projet Maven**
  - Commande `mvn site`
- **Supprimer le répertoire target**
  - Commande `mvn clean`
- **Compiler, tester et packager (Java)**
  - `mvn compile`
  - `mvn test-compile`
  - `mvn package`
- **Installer un jar**
  - `mvn install`



- **Réaliser l'exercice source developpez.com.**
  - Exercice fourni
  - <http://dcabasson.developpez.com/articles/java/maven/introduction-maven2/#references>



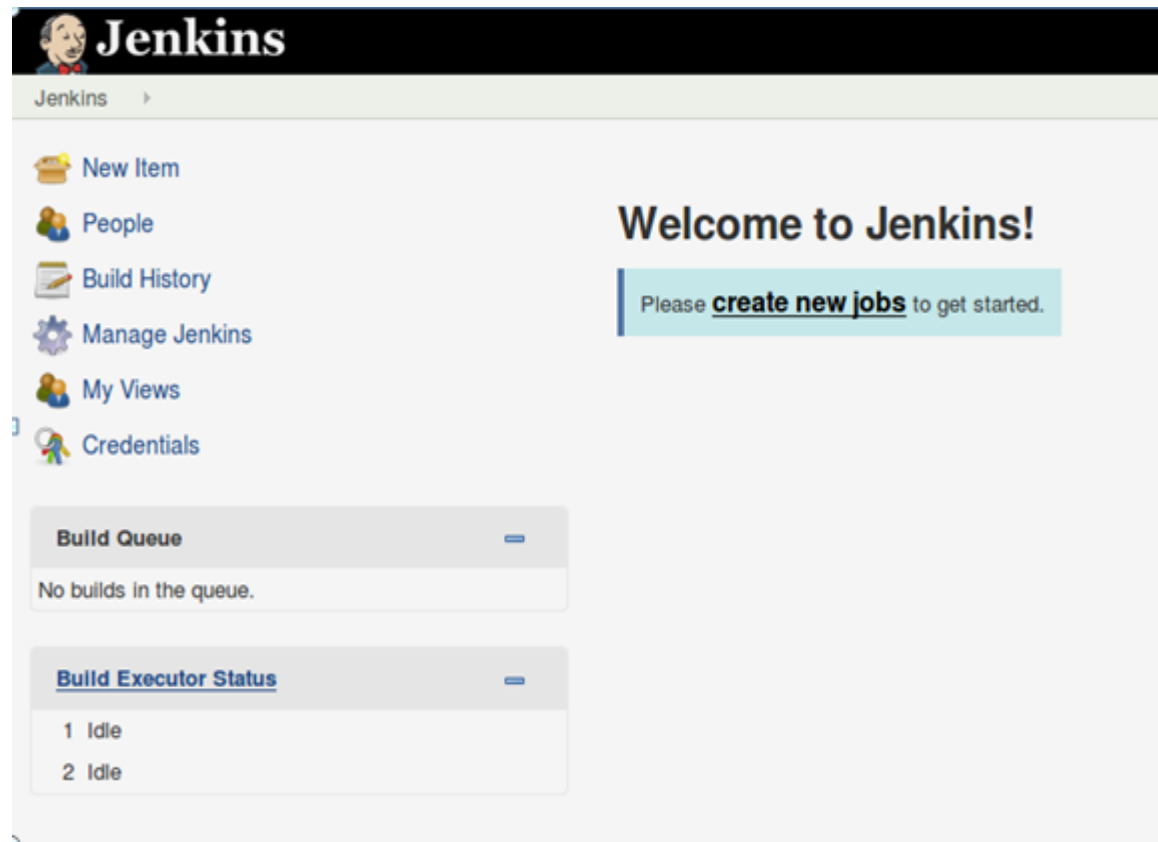
### Partie 3 : Prise en main de Jenkins



# Installation et paramétrage des outils

## Prise en main de Jenkins

- **Premier lancement de Jenkins :**
  - Exercice (5 minutes) : récupérer le mot de passe administrateur et accéder à Jenkins pour la première fois.
  - Vous arrivez sur la page d'accueil suivante :



# Installation et paramétrage des outils

## Prise en main de Jenkins

- Ecran d'accueil de l'application Jenkins (écran admin) (1/3)



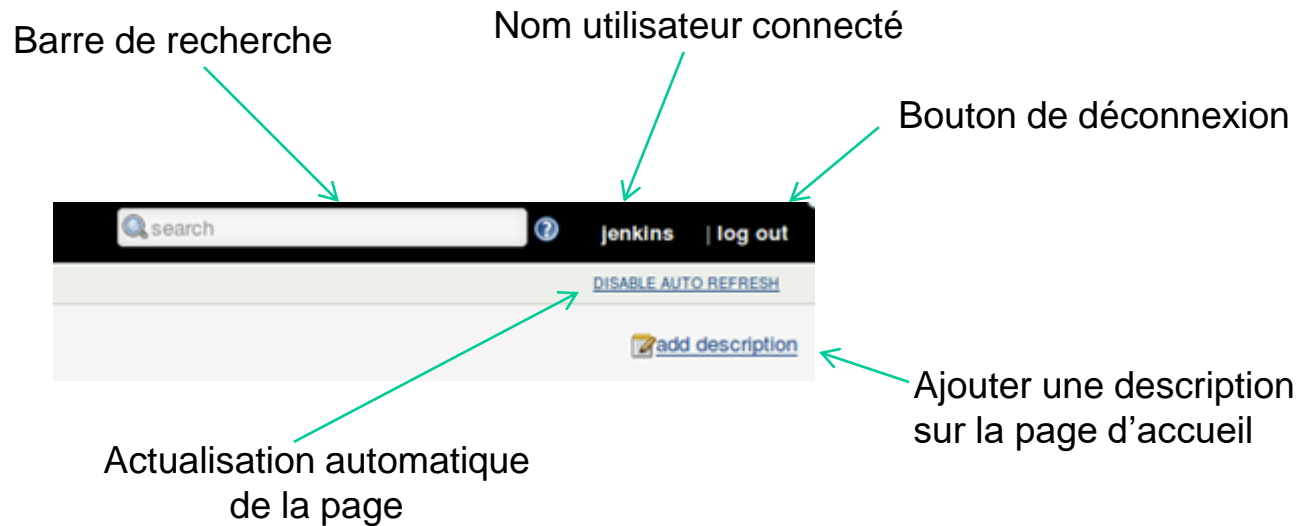
The screenshot shows the Jenkins administration interface. On the left, a sidebar contains several menu items: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'Credentials'. Below these are sections for 'Build Queue' (showing 'No builds in the queue') and 'Build Executor Status' (showing two 'Idle' executors). On the right, a large area displays 'Welcome to Jenkins!' and a message: 'Please create new jobs to get started.'.

Annotations with arrows point from the following text to the interface:

- Nouveau job → New Item
- Liste utilisateur(s) → People
- Historique des jobs → Build History
- Configuration Jenkins → Manage Jenkins
- Vue projet → My Views
- Gestion profils → Credentials
- Jobs en attente → Build Queue
- Descriptif des jobs exécutés depuis l'écran d'accueil → Build Executor Status

Créer un nouveau job Jenkins

- Ecran d'accueil de l'application Jenkins (écran admin) (2/3)



# Installation et paramétrage des outils

## Prise en main de Jenkins

- **Ecran d'accueil de l'application Jenkins après le lancement d'un job et création d'un projet (écran admin) (3/3)**
  - Possibilité de créer de nouvelles vues
  - Ligne du job

The screenshot displays the Jenkins administration page. On the left is a sidebar with navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, and Credentials. The main area shows a 'test description' header with an 'edit description' link. Below this is a table of jobs. The table has columns for 'S' (Status), 'W' (Weather icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. One job, 'test\_maven\_project', is listed with a status of 'S' and a duration of '35 sec'. Below the table are links for 'Icon: S M L' and several RSS feeds. At the bottom, there are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle').

S	W	Name	Last Success	Last Failure	Last Duration
		<a href="#">test_maven_project</a>	46 min - #4	N/A	35 sec

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

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












**Build Queue**  
No builds in the queue.

**Build Executor Status**  
1 Idle  
2 Idle



# Installation et paramétrage des outils

## Prise en main de Jenkins

	<a href="#">Configure System</a> Configure global settings and paths.	→ Panneau de configuration de Jenkins
	<a href="#">Configure Global Security</a> Secure Jenkins; define who is allowed to	→ Panneau de configuration de Jenkins
	<a href="#">Configure Credentials</a> Configure the credential providers and typ	→ Gestion des mots de passe
	<a href="#">Global Tool Configuration</a> Configure tools, their locations and auton	→ Configuration des outils utiliser pour la construction des builds
	<a href="#">Reload Configuration from Disk</a> Discard all the loaded data in memory an	→ Réinitialise les tâches en mémoire
	<a href="#">Manage Plugins</a> Add, remove, disable or enable plugins th	→ Panneau de gestion des plugins
	<a href="#">System Information</a> Displays various environmental informati	→ Informations de l'installation Jenkins sur le serveur d'intégration
	<a href="#">System Log</a> System log captures output from java.	→ Logs Jenkins
	<a href="#">Load Statistics</a> Check your resource utilization and see i	→ Statistiques Jenkins
	<a href="#">Jenkins CLI</a> Access/manage Jenkins from your shell,	→ Liste des lignes de commandes Jenkins
	<a href="#">Script Console</a> Executes arbitrary script for administrat	→ Console de création de scripts Jenkins

# Installation et paramétrage des outils

## Prise en main de Jenkins



### [Manage Users](#)

Create/delete/modify users that can log in to this Jenkins

- Permet de gérer et créer les utilisateurs Jenkins.
  - **Exercice** : créer un nouvel utilisateur (10 minutes)



### [Manage Plugins](#)

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

- Gestion des plugins.
  - Les plugins permettent d'ajouter de nouvelles fonctionnalités à Jenkins
  - Par exemple Maven Integration Plugin permet de créer un nouveau projet Maven. Sans ce plugin, la fonctionnalité n'est pas proposée.
  - Présentation de l'interface plugins :

Updates			
Available			
Installed			
Advanced			
Install	Name ↓	Version	Installed
No updates			

Update information obtained: 3 hr 15 min ago

[Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.

- **Exercice (10 minutes) :**
  - Ajouter dans Jenkins le plugin suivant :



- Mettre à jour les plug-ins si possible



# Installation et paramétrage des outils

## Prise en main de Jenkins



### Global Tool Configuration

Configure tools, their locations and automatic installers.

- Permet de configurer les outils utilisés par les jobs Jenkins
- Les outils dépendent des plugins précédemment installés
- Vous pouvez par exemple paramétrer votre jdk, vos dépôts git et svn et maven
- Présentation du Global Tool Configuration
- **Exercice (10 minutes) :**
  - Paramétrez vos outils JDK et Maven
  - Pensez à appliquer et valider vos modifications



- Solution de l'exercice 10 (1/2)

### JDK

JDK installations

JDK

Name

jdk8



Install automatically



Install from java.sun.com

Version

Java SE Development Kit 8u121



I agree to the Java SE Development Kit License Agreement

Delete installer

Add Installer



Delete JDK

Add JDK

List of JDK installations on this system



- Solution de l'exercice 10 (2/2)

### Maven

Maven installations

Maven

Name maven

☒ Install automatically



Install from Apache

Version 3.3.9 ▼

Delete Installer

Add Installer ▼

Delete Maven

Add Maven

List of Maven installations on this system



- Menus de nouveau item (job) (change selon les versions et les plugins installés) :
  - Construire un projet freestyle : projet nu où tout doit être paramétré manuellement
  - Construire un projet Maven : projet qui utilise un build Maven
  - Pipeline : permet d'orchestrer plusieurs jobs freestyles → présentation rapide
  - Construire un projet multi-configuration : projet complexe
- Une fois créé, le job apparaît dans la page d'accueil
- Il est possible de facilement supprimer ou modifier un job
- Pour aller sur un job, cliquer dessus



- Création d'un job de test Squash TA - Maven
  - Choisir le type de job souhaité
  - La création d'un job passe par la création d'une configuration globale du job
  - Vous devez savoir ce que vous souhaitez mettre dans votre job
  - Les plugins permettent d'ajouter de nouvelles fonctionnalités (complexe)
  - Exercice (45 minutes) : création d'un projet run test Squash TA
    - Suivre la procédure suivante : <https://sites.google.com/a/henix.fr/wiki-squash-ta/ta-serveur-guide/create-a-job>
    - Vous importerez un projet local créé pour l'occasion
    - Vérifier les paramètres
    - Essayer d'exécuter votre premier build (marche rarement du premier coup)
  - Retour en groupe sur le projet créé
    - Les plugins utilisés
    - Les paramètres
    - L'environnement global du projet





# Installation et paramétrage des outils

## Prise en main de Jenkins

Gérer un job depuis la page d'accueil

Ajouter un job

Onglets de rangement

Ligne du job

Modifier le job

Répertoire qui contient  
le projet

Exécuter le job

Supprimer le job

Configurer le job

Accès dépôts

The screenshot shows the Jenkins job management page. At the top, there are tabs for 'Tous' and 'test', with a '+' button to add a new job. Below the tabs is a table with columns: 'S' (Status), 'M' (Maintenance), 'Nom du projet ↓' (Project Name), and 'Dernier succès' (Last Success). The table contains four rows of jobs. A context menu is open over the first row, showing options: 'Modifications', 'Répertoire de travail', 'Lancer un build', 'Supprimer Maven project', 'Configurer', and 'Modules'. Green arrows point from the text labels on the left to specific elements in the interface.

S	M	Nom du projet ↓	Dernier succès
		<a href="#">test</a>	S. O.
			1 h 48 mn - <a href="#">#4</a>
			S. O.
			S. O.

Context menu options:

- Modifications
- Répertoire de travail
- Lancer un build
- Supprimer Maven project
- Configurer
- Modules

- Planifier des jobs
- Orchestrer des jobs



- Exercice final (1 heure) : Importer votre projet sélénium dans Jenkins et le lancer



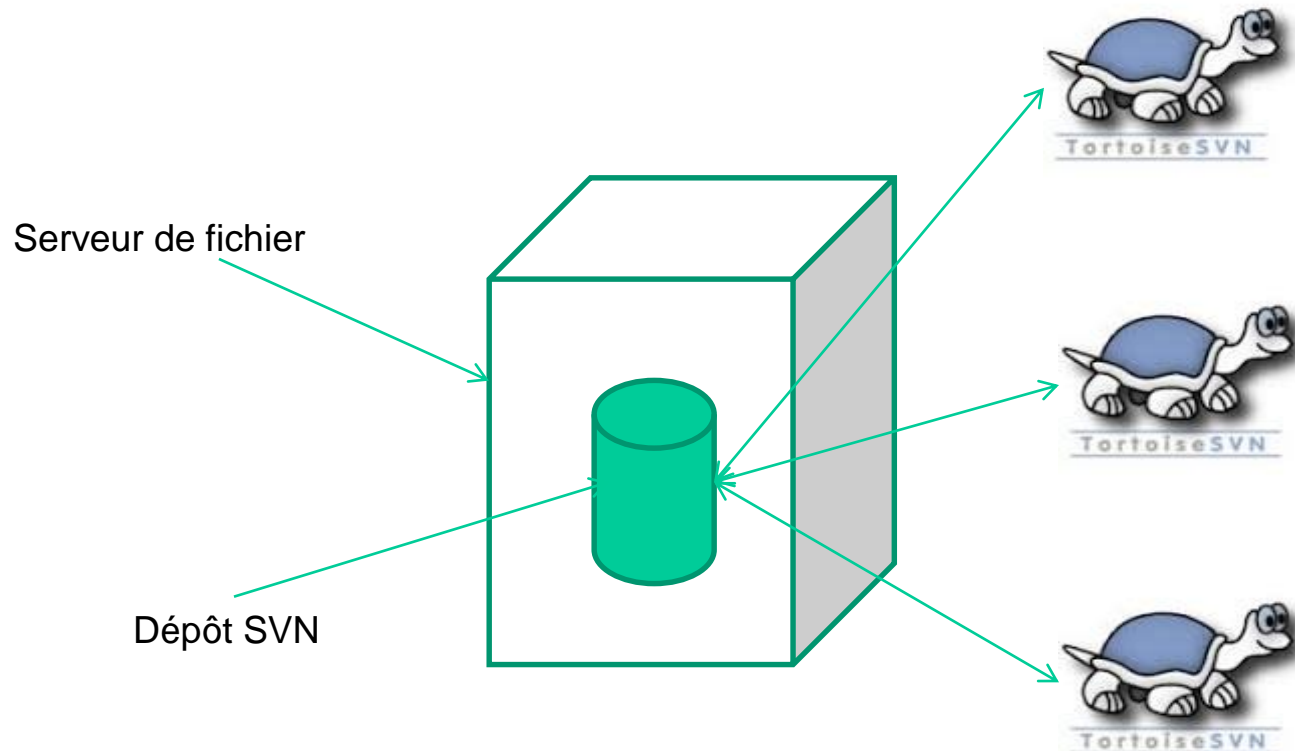
### Partie 4 : Introduction à SVN



# Installation et paramétrage des outils

## SVN : présentation de Subversion (1/2)

- Principe de fonctionnement : Subversion utilise une base de données centrale qui contient tous vos fichiers sous contrôle de version avec leur historique complet.
- Autres technologies Git, Mercurial, Bazaar, CVS

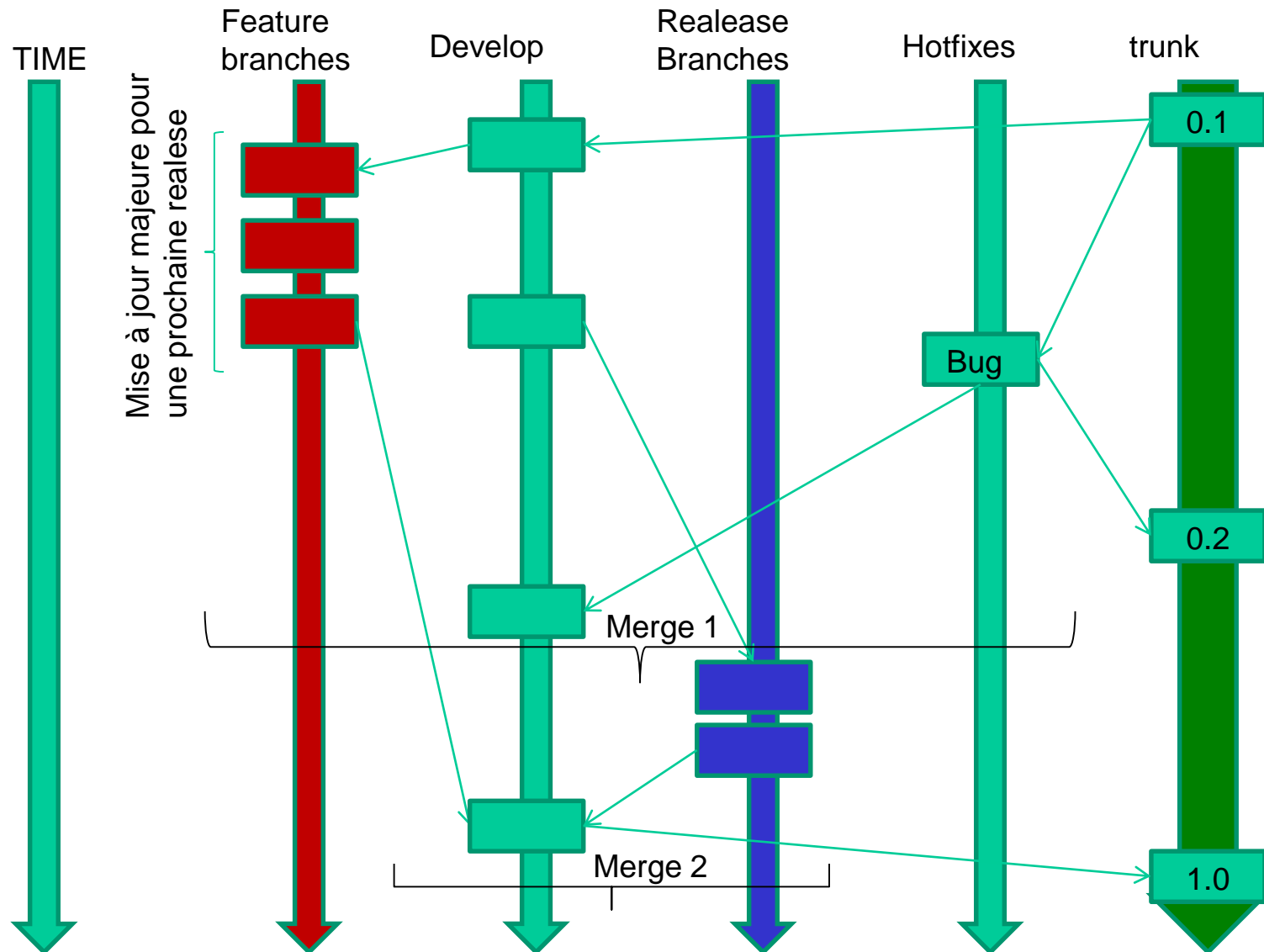


- En entreprise, votre poste sera muni d'un client Subversion comme Tortoise SVN
- Votre dépôt se trouvera sur un serveur de fichiers
- L'utilisation de Subversion (ou une technologie analogue) demande de la rigueur et une organisation de groupe.
- Tous vos enregistrements dans le (ou les) dépôts seront tracés, annotés et datés.
- Subversion permet de travailler à plusieurs, en plusieurs temps sur un même projet, ou de fusionner des projets ...



# Installation et paramétrage des outils

## SVN : Cas d'utilisation de subversion



# Installation et paramétrage des outils

## SVN : Structure de votre dépôt SVN

- **Trunk** : Représente l'axe principal de développement de votre projet
- **Branches** : Représente le ou les axes de développement parallèles de votre projet
- **Tag** : Marque dans le tronc des versions abouties de votre de développement
- **Revision** : Chaque action dans SVN implique un commit de la part du développeur. Chaque commit, commenté et daté, fabrique un historique et une possibilité de « rollback » sur votre projet. La dernière révision est appelée HEAD. Les autres révisions sont numérotées.
- **Les commentaires** sont obligatoires pour toutes actions qui affectent le dépôts



- **Checkout** : descendre un projet vers votre copie de travail
- **Commit** : envoyer votre copie de travail vers votre dépôt SVN
- **Merge** : fusionner des travaux différents dans un même dépôt
- **Update** : mettre à jour votre copie de travail depuis un dépôt cible
- **Clean up** : nettoyer votre copie de travail
- **Switch** : changer de branches dans votre dépôt



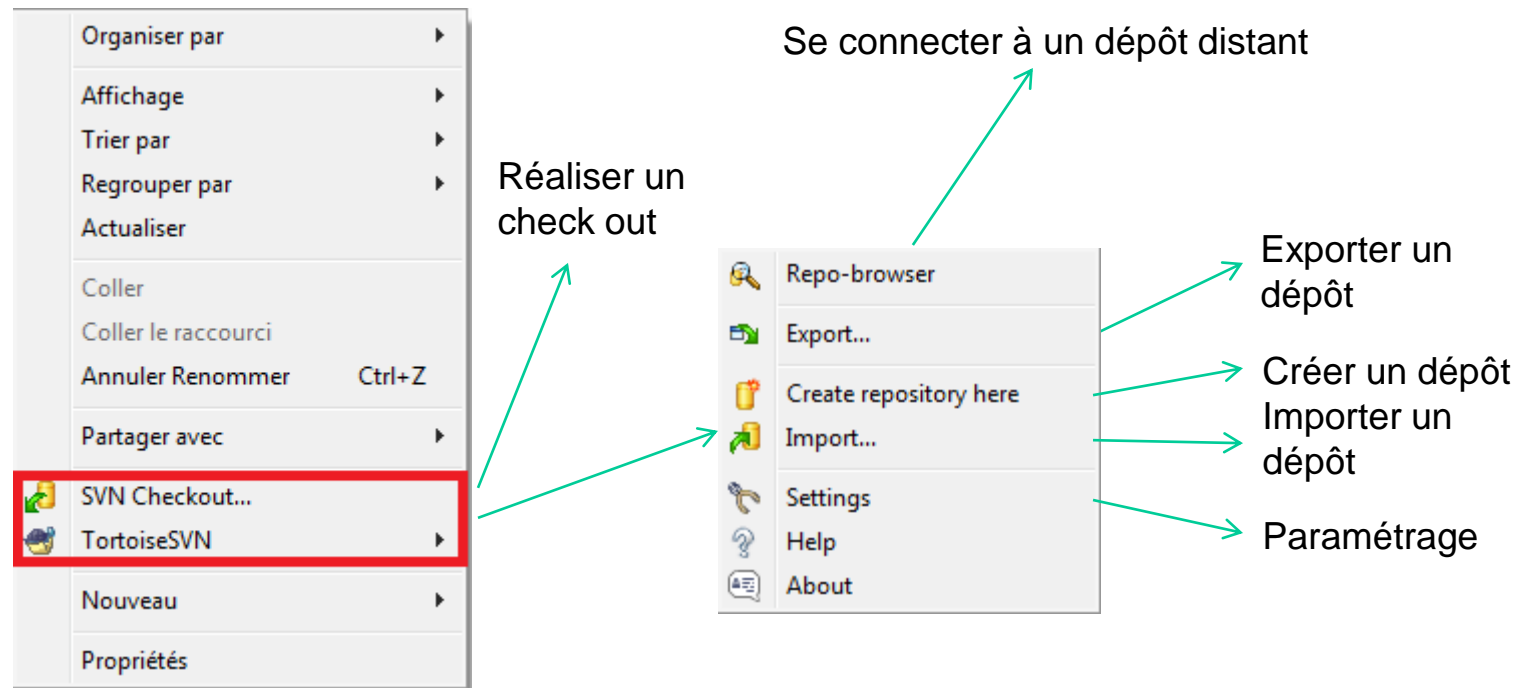
- Deux clients libres (principaux) peuvent être utilisés avec SVN (d'autres clients existent)
- Les deux clients sont utilisables en ligne de commande et graphiquement



# Installation et paramétrage des outils

## SVN : Utilisation de Tortoise SVN

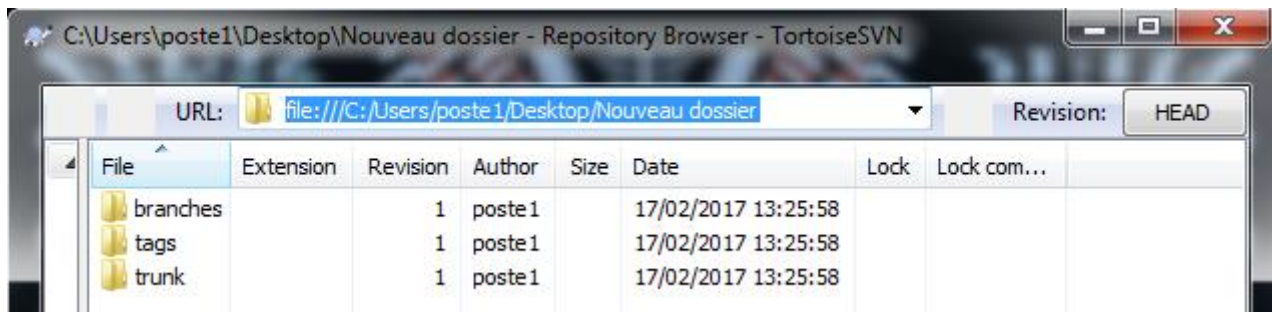
- Vous pouvez utiliser Tortoise SVN graphiquement
- Présentation de l'interface graphique de Tortoise SVN :



# Installation et paramétrage des outils

## SVN : Utilisation de Tortoise SVN

- Créer un dépôt :
  - Créer le répertoire du dépôt
  - Cliquer sur « Create repository here »



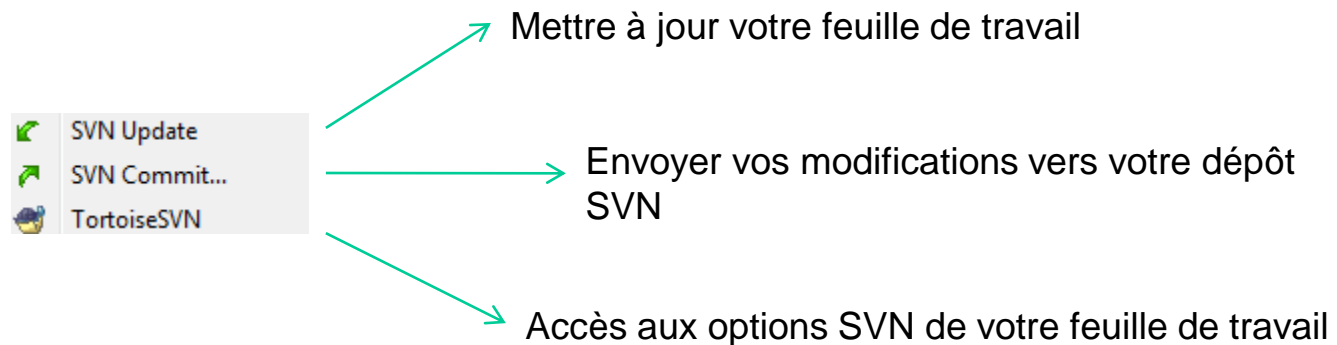
- Repo-Browser
  - Affiche la structure de votre dépôt
- Checkout :
  - Créer le répertoire de réception
  - Toutes actions sous SVN doivent être justifiées par un message explicatif (obligatoire)



# Installation et paramétrage des outils

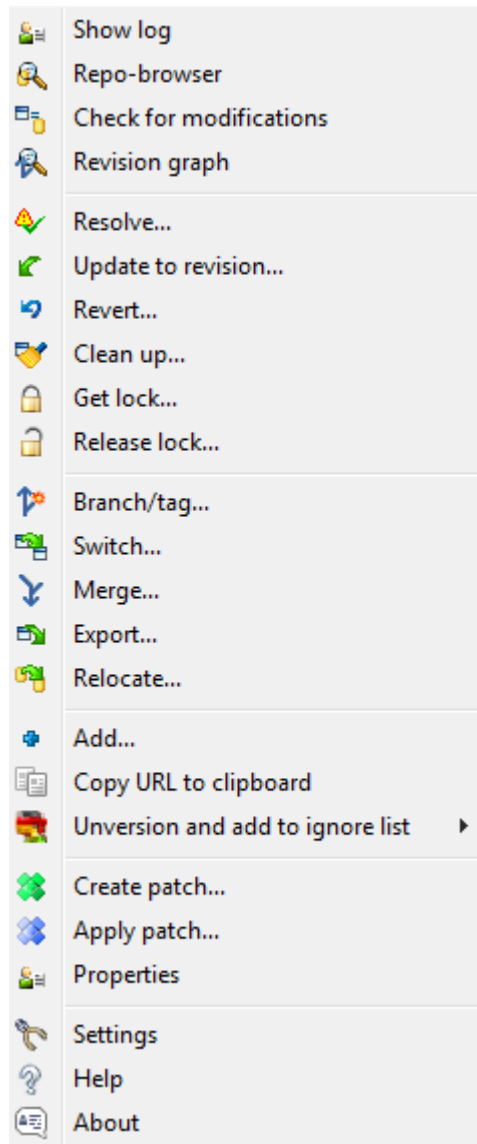
## SVN : Utilisation de Tortoise SVN

- Une fois un dépôt importé dans un répertoire de votre choix, ce répertoire devient votre feuille de travail SVN



# Installation et paramétrage des outils

## SVN : Utilisation de Tortoise SVN



- Exemple manipulation :



- Exercice 1 : Télécharger et installer Tortoise SVN (20 minutes)
  - Créer un dépôt dans le répertoire de votre choix
  - Importer vos tests automatisés dans votre dépôts
  - Créer votre copie de travail
  - Modifier un test et mettre à jour le dépôt SVN
  - Créer un autre répertoire et importer votre projet. Que remarquez-vous?
- Exercices 2 : Réaliser des modifications sur votre projet et mettre à jour votre dépôt SVN. Manipuler SVN.
- Exercice 3 : Connecter votre dépôt SVN sur votre projet Jenkins





### Partie 5 : Introduction à Nexus (support annexe)



# Annexes



- 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. Comme Hudson évoluait au cours des années, il était de plus en plus utilisé par les équipes de Sun pour leurs propres projets. Début 2008, Sun reconnaissait la qualité et la valeur de cet outil et demandait à Kohsuke de travailler à plein temps sur Hudson, commençant ainsi à fournir des services professionnels et du support 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 apparurent entre la communauté de développeurs d'Hudson et d'Oracle, dont la source était les problèmes de l'infrastructure Java.net, aggravées par la politique d'Oracle au sujet de la marque Hudson. Ces tensions révélaient de profonds désaccords sur la gestion du projet par Oracle. En effet, Oracle voulait orienter le projet vers un processus de développement plus contrôlé, avec des livraisons moins fréquentes, alors que le cœur des développeurs d'Hudson, mené par Kohsuke, préférait continuer à fonctionner selon le modèle communautaire ouvert, flexible et rapide qui avait si bien fonctionné pour Hudson par le passé. (source : Guide complet Jenkins)



- En Janvier 2011, la communauté des développeurs Hudson vota pour renommer le projet en Jenkins. Par la suite ils migrèrent le code originel d'Hudson vers un nouveau [projet Github](#) et y poursuivirent leur travail. La grande majorité des développeurs du coeur et des plugins leva le camp et suivit Kohsuke Kawaguchi et les autres principaux contributeurs dans le camp de Jenkins, où le gros de l'activité de développement peut être observée aujourd'hui.
- Après ce fork, la majorité des utilisateurs suivit la communauté des développeurs Jenkins et passa à Jenkins. Au moment où ces lignes sont écrites, les sondages montrent qu'environ 75% des utilisateurs d'Hudson sont passés à Jenkins, 13 % utilisent encore Hudson et 12% utilisent Jenkins et Hudson ou sont en cours de migration vers Jenkins.
- Cependant, Oracle et Sonatype (la compagnie derrière Maven et Nexus) ont poursuivi leur travail à partir du code d'Hudson (qui est maintenant lui aussi hébergé chez GitHub à <https://github.com/hudson>), mais selon un axe différent. En effet, les développeurs de Sonatype se sont concentrés sur des modifications dans l'architecture, notamment sur l'intégration avec Maven, sur le framework d'injection de dépendances, et sur l'architecture des plugins. (source : Guide complet Jenkins)



- **Exercice 1 (facultatif)** : Depuis l'application VirtualBox, créer une nouvelle machine virtuelle Ubuntu (système Linux). (50 minutes)
- La machine doit :
  - Avoir suffisamment d'espace disque (il est conseillé de définir directement une taille de mémoire conséquente)
  - Avoir suffisamment de ressource machine
  - Avoir un nom et mot de passe simple
  - Les utilisateurs de Windows 10 peuvent directement créer une machine Ubuntu
  - Ne pas mettre à jour la machine virtuelle (chez un client peut représenter une faille de sécurité)



- **Exercice 2 (facultatif)** : Monter votre environnement Tomcat en vous aidant de la documentation Tomcat/Ubuntu (internet). Pour rappel, Tomcat est un conteneur web. Il intégrera l'application Jenkins (1 heures)
  - Afin de vérifier la bonne installation de serveur d'application, vérifiez vos logs (Tomcat → catalina.out)
  - Systématiquement, avant de passer à l'étape suivante, assurez-vous que tout fonctionne parfaitement.
- **Exercice 3 (facultatif)** : Vérifier et paramétrer votre environnement Java. Paramétrer votre JAVA\_HOME (/etc/environment). Vérifiez votre version de Java (java – version) (15 minutes)



- **Exercice 4 (facultatif) :** En utilisant le support Jenkins Ubuntu, installez votre application Jenkins dans le serveur Tomcat. (1h20)
  - Vous n'utiliserez pas le dépôt apt-get
  - Veillez à ce que Jenkins appartienne à l'utilisateur et groupe Tomcat
  - Adaptez les commandes d'installation à votre version de Tomcat (Tomcat7)
  - En suivant la documentation, vérifiez vos variables d'environnement
  - Depuis les logs Tomcat (catalina.out), vérifiez que l'application Jenkins se déploie correctement
  - Tester une première connexion depuis l'URL suivante `http://localhost:8080/jenkins`. Trouvez la solution pour la première authentification
  - Choisissez les plugins dont vous avez besoin.
  - Vous créerez deux scripts (en shell), un script de lancement de la servlet Jenkins et un script de déploiement de Jenkins. Vos scripts doivent avoir une gestion d'erreur et des logs (obligatoire).

