

Mettre en oeuvre une démarche d'intégration continue





- *Le monde d'intégration continue*
- *Méthodes agiles*
- *Gestion des sources*
- *Les tests*
- *Automatisation des tâches*
- *Gestion des configurations*
- *Gestion des artefacts (livraisons)*
- *Le serveur d'intégration continue*
- *La mise en place des métriques*

Monde de l'intégration continue





Pratiques d'ingénierie

- *Le renforcement de la discipline de développement, et de la mise en place rigoureuse de certaines pratiques d'ingénierie, renforce le niveau de la qualité logicielle.*
- *Exemples de pratiques d'ingénierie :*
 - *tests automatisés,*
 - *développement incrémental,*
 - *développement piloté par les tests,*
 - *intégration continue,*
 - *conception simple, refactoring,*
 - *programmation en binôme...*
 - *Méthodes agiles*



Méthodes agiles

- *Simplicité*
- *Légèreté*
- *Orientées participants plutôt que plan*
- *Nombreuses*
 - *XP*
 - *scrum (la plus connue)*
- *Pas de définition unique Mais un manifeste*
- *Cycle itératif avec des releases*



Méthodes agiles

● *Années 90*

- *réaction contre les méthodes classiques*
- *prise en compte de facteurs liés au développement logiciel*

● *Fin années 90*

- *méthodes*
 - *d'abord des pratiques liées à des consultants, puis des livres*
 - *XP, Scrum, FDD, Crystal...*

● *2001*

- *les principaux acteurs s'accordent sur le manifeste « Agile manifesto »*

● *Depuis*

- *projets Agile mixent des éléments des principales méthodes*



Méthodes agiles

- *Février 2001, rencontre et accord sur un manifeste*
- *Mise en place de la « Agile alliance »*
 - *objectif : promouvoir les principes et méthodes Agile*
 - *<http://www.agilealliance.com>*
- *Les signataires privilégient*
 - *les individus et les interactions davantage que les processus et les outils*
 - *les logiciels fonctionnels davantage que l'exhaustivité et la documentation*
 - *la collaboration avec le client davantage que la négociation de contrat*
 - *la réponse au changement davantage que l'application d'un plan*
- *12 principes*



Méthodes agiles - principes

- *Extraits de manifesto (<http://agilemanifesto.org/principles.html>)*
 - *1 - Our highest priority is to satisfy the costumer through early and continuous delivery of valuable software.*
 - *2 - Welcome changing requirements, even late in development. Agile process harness change for the customer's competitive advantage.*
 - *3 - Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
 - *4 - Business people and developers must work together daily throughout the project.*
 - *5- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
 - *6 - The most efficient and effective method of conveying information to and within a development team is face--to--face conversation.*



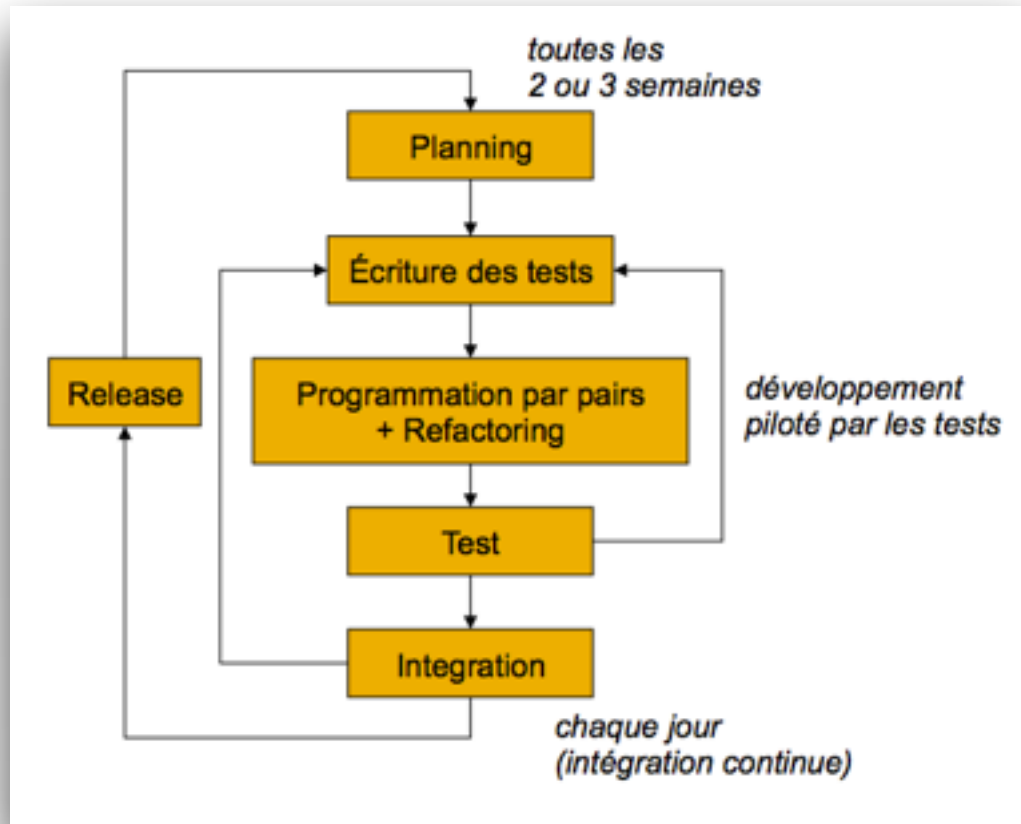
Méthodes agiles - principes

● Extraits de manifesto (<http://agilemanifesto.org/principles.html>)

- 7- Working software is the primary measure of progress
- 8 - Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
- 9 - Continuous attention to technical excellence and good design enhances agility
- 10 - Simplicity – the art of maximizing the amount of work not done – is essential
- 11- The best architectures, requirements, and designs emerge from self-organizing teams
- 12- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly



Méthodes agiles - XP



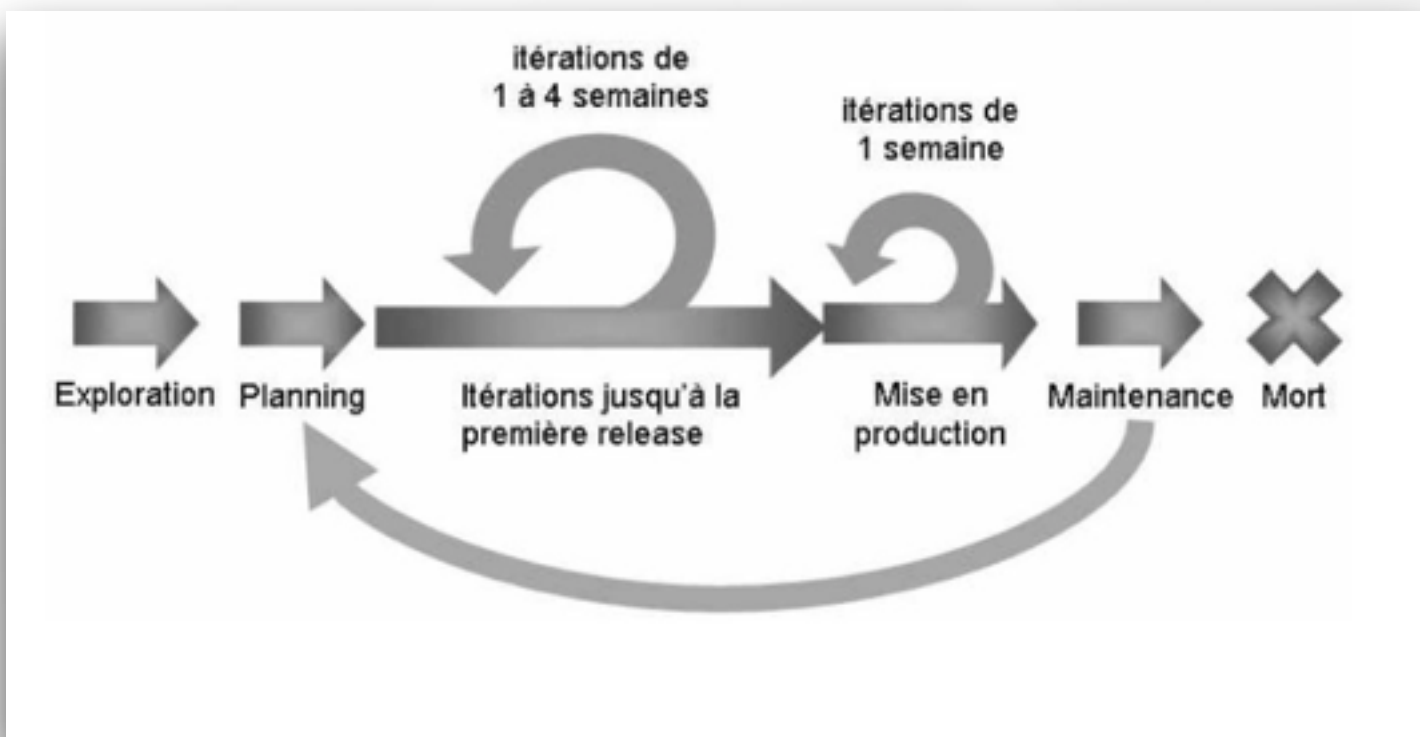
● *L'intégration continue est une composante importante de la méthodologie*



Méthodes agiles - XP



Processus





● *Profils intervenants dans une équipe XP :*

- *Client*
- *Testeur*
- *Manager*
- *Coach*
- *Tracker*
- *Programmeur*



Méthodes agiles - Scrum

● Scrum : mêlée

● Phases

● Initiation / démarrage

● Planning

* définir le système : product Backlog = liste de fonctionnalités, ordonnées par ordre de priorité et d'effort

● Architecture

* conception de haut--niveau

● Développement

● Cycles itératifs (sprints) : 30j

* amélioration du prototype

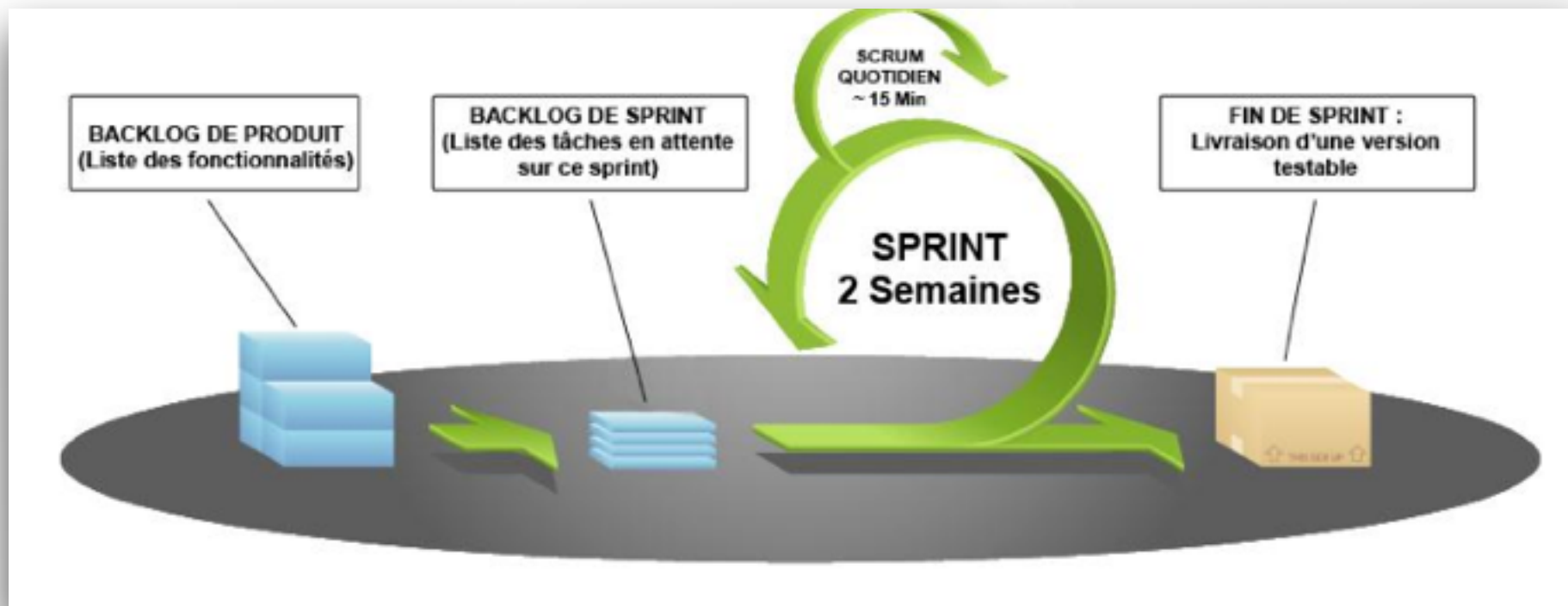
● Clôture

● Gestion de la fin du projet : livraison



Méthodes agiles - Scrum

● Le processus





Méthodes agiles - Scrum

● *Product Backlog*

- *état courant des tâches à accomplir*

● *Effort Estimation*

- *permanente : sur les entrées du backlog*

● *Sprint*

- *itération de 30 jours*

● *Sprint Planning Meeting*

- *réunion de décision des objectifs du prochain sprint et de la manière de les implémenter*

● *Sprint Backlog*

- *Product Backlog limité au sprint en cours*

● *Daily Scrum meeting*

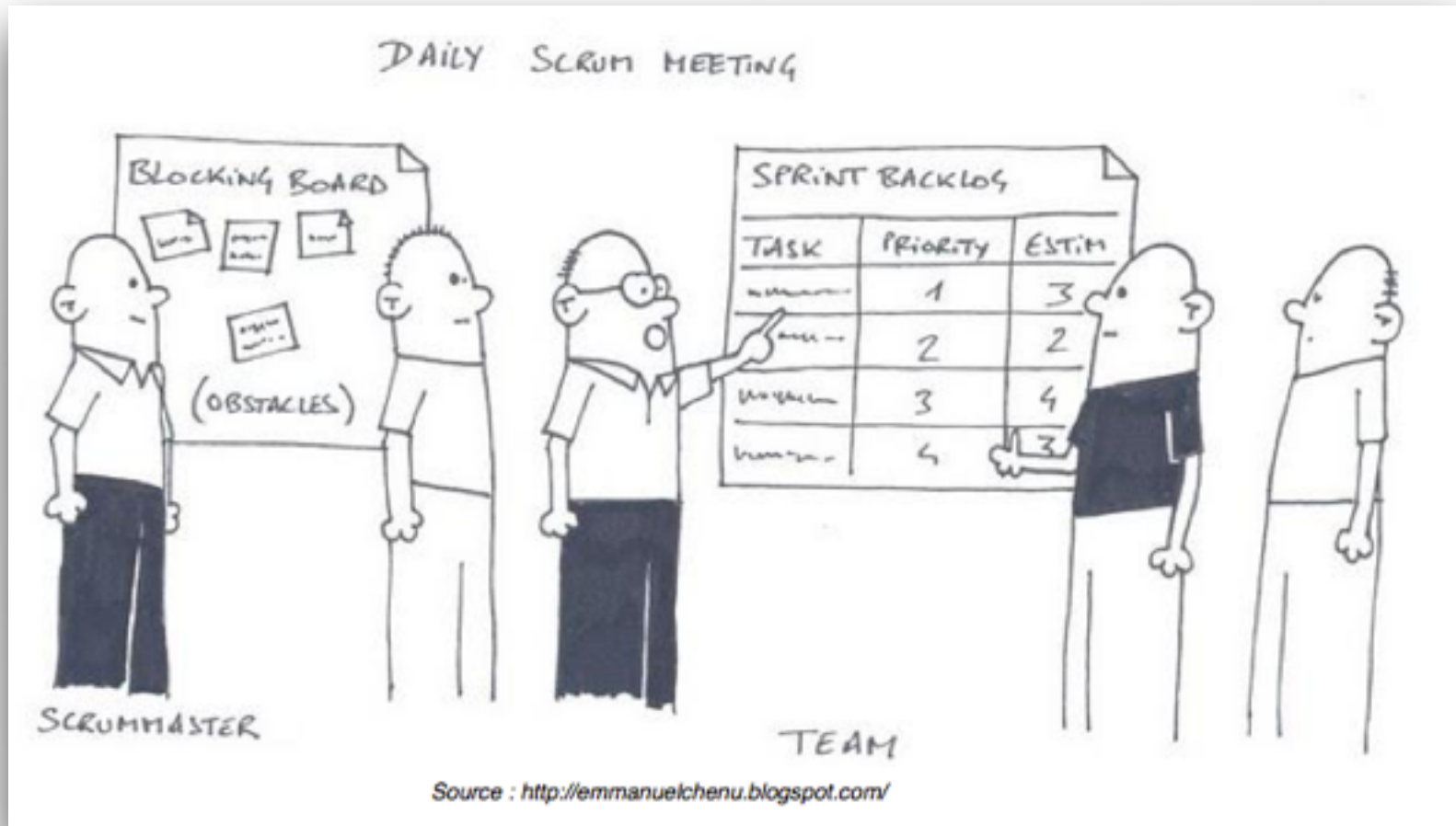
- *ce qui a été fait, ce qui reste à faire, les problèmes*

● *Sprint Review Meeting*

- *présentation des résultats du sprint*



Méthodes agiles - Scrum





Intégration continue

- *L'intégration continue est un ensemble de pratiques d'ingénierie*
- *Elle est basée sur l'exécution automatique*
 - *des tests unitaires et d'intégration*
 - *génération des rapports*
 - *les tâches de build*
 - *Le déploiement*
 - *..*



- *Elles sont utilisées en génie logiciel dont l'objectif est :*
 - *de vérifier que les modifications de code source n'induisent pas de régression*
 - *détecter au plutôt les erreurs d'intégration*



● [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*



- *Les principaux avantages d'une telle technique de développement sont :*
 - *le test immédiat des unités modifiées*
 - *la prévention : détection 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*



IC - execution des tests

- *Les outils d'IC permettent d'automatiser l'exécution de tests...*
 - *déclenchés selon différentes politiques*
 - *périodiquement,*
 - *manuellement,*
 - *détection d'un commit SVN...*
 - *classiquement : compilation, analyse de code, tests unitaires*



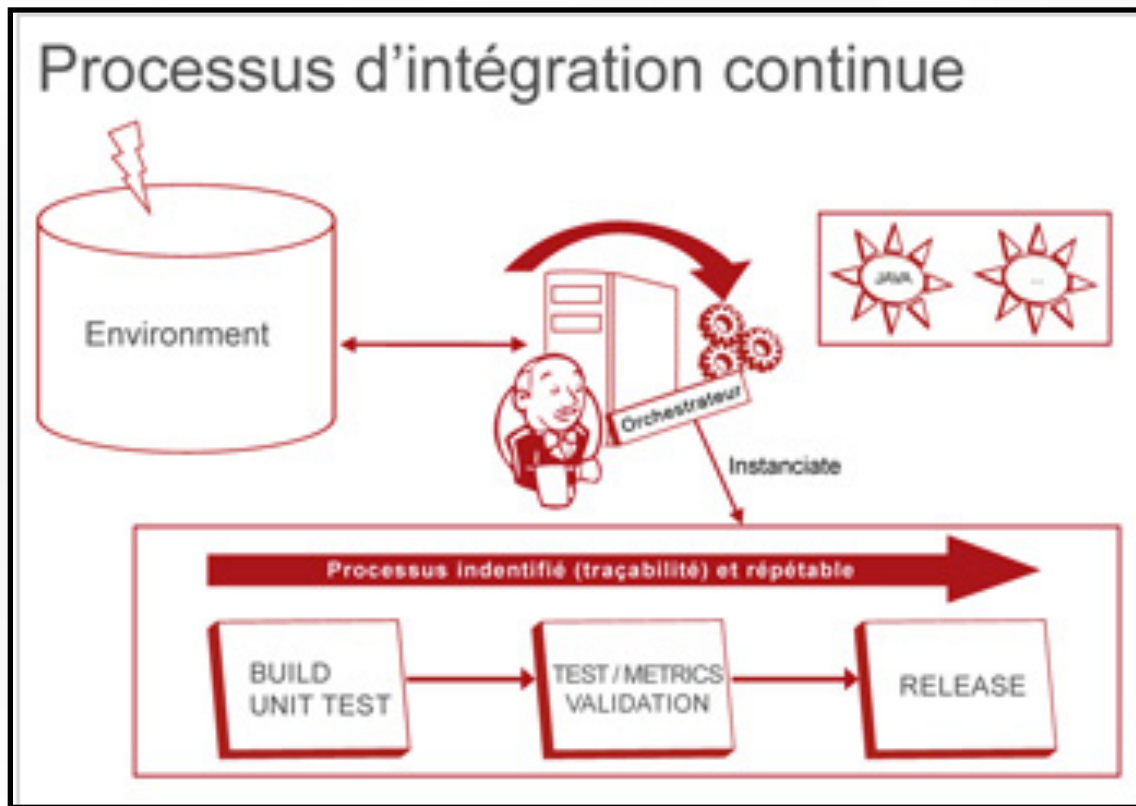
IC - génération des rapports

- *Les outils d'intégration continue permettent la production de rapports*
 - *métrique de qualité*
 - *tendances*
 - *alertes*
 - *...*



Intégration continue

● Processus d'intégration continue





● *Forme simple d'intégration*

- *à la détection des modifications de code, l'outil d'intégration continue procède :*
 - *à la compilation des code source et*
 - *à l'exécution des tests unitaires*
 - *...*



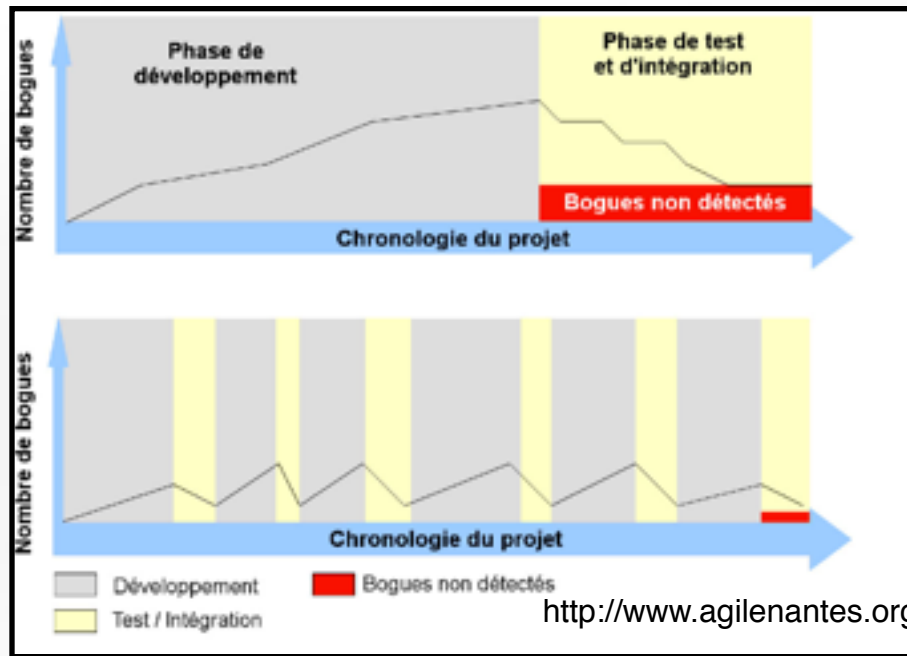
● *Version améliorée d'intégration*

- *Surveillance de la qualité de code*
- *Génération des métriques de couverture de tests*
- *Génération des rapports*
- *Afficher l'état des développements encours*
- *Automatiser le processus de génération des versions stables (releases) : build*
- *Automatiser le processus de déploiement*



Intégration continue

● Une méthode agile 😊



● Principe *plus fréquente,* *moins elle est longue*



● Construire des builds tous les jours...

- *Chargement de la dernière version du projet depuis le gestionnaire de version*
- *Inspection du code (en vue de générer les métriques de qualité)*
- *Génération de la documentation, des rapports, des notes de release (par exemple la Javadoc, rapport Checkstyle)*
- *Construction de la release*
- *Déploiement de l'application sur l'environnement d'intégration*
- *Exécution des tests d'intégration*



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*



- *Un gestionnaire de versions*
 - *SVN, git*
- *Gestionnaire de dépendances, building*
 - *Maven, gradle, ivy,*
- *Tests unitaires*
 - *jUnit, Mockito, Eclemma*
- *Un serveur d'intégration continue*
 - *jenkins,*
- *Repository manager*
 - *Artifactory, Archiva, Nexus*
- *Outils de qualimétrie*
 - *sonar*

DevOps

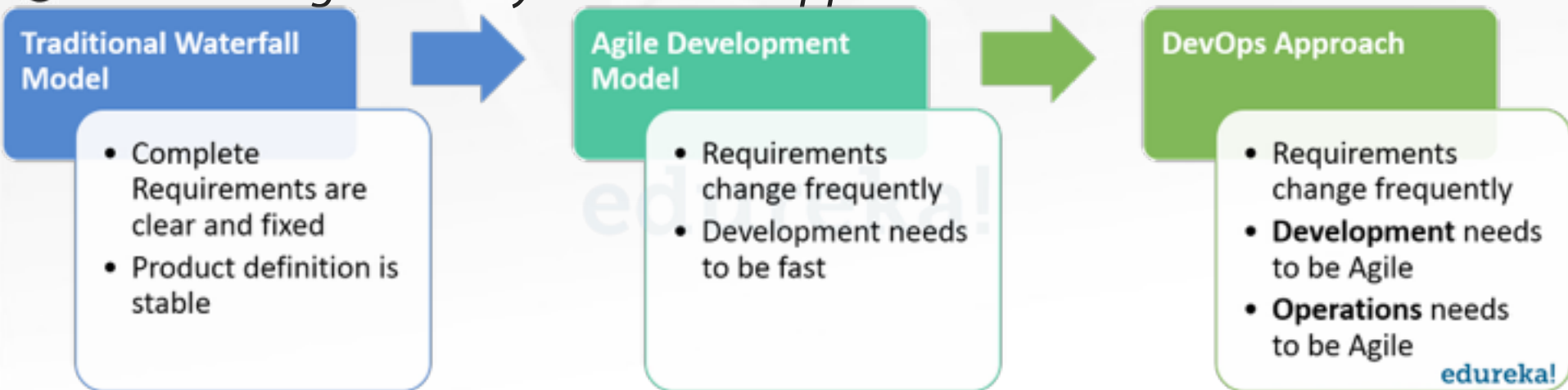




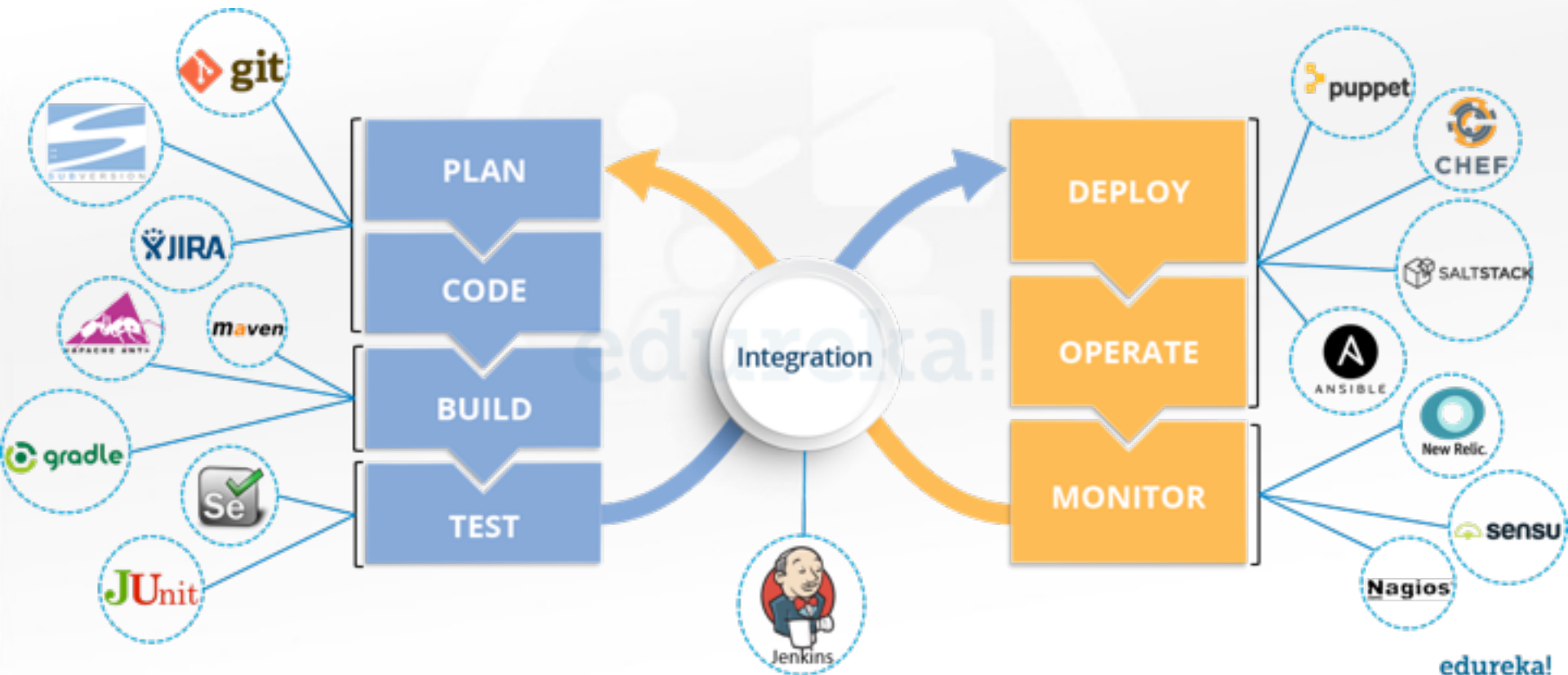
● *DevOps est une approche de développement logiciel qui implique :*

- *développement continu,*
- *tests continus,*
- *intégration continue,*
- *déploiement continu et*
- *une surveillance continue du logiciel*

● *Tout au long de son cycle de développement*



DevOps cible le déploiement et la surveillance continue en production





- «DevOps promotes a set of processes and methods for thinking about communication and collaboration between development, QA, and IT operations.»

● <https://en.wikipedia.org/wiki/DevOps>

Gestion des configurations





Gestion des configuration

- *Infrastructure as software*
- *Reproducible setups*
 - *Do once, repeat forever*
- *Scale quickly: Done for one, use on many*
- *Coherent and consistent server setups*
- *Aligned Environments for dev, test, qa, prod nodes*



Gestion des configurations

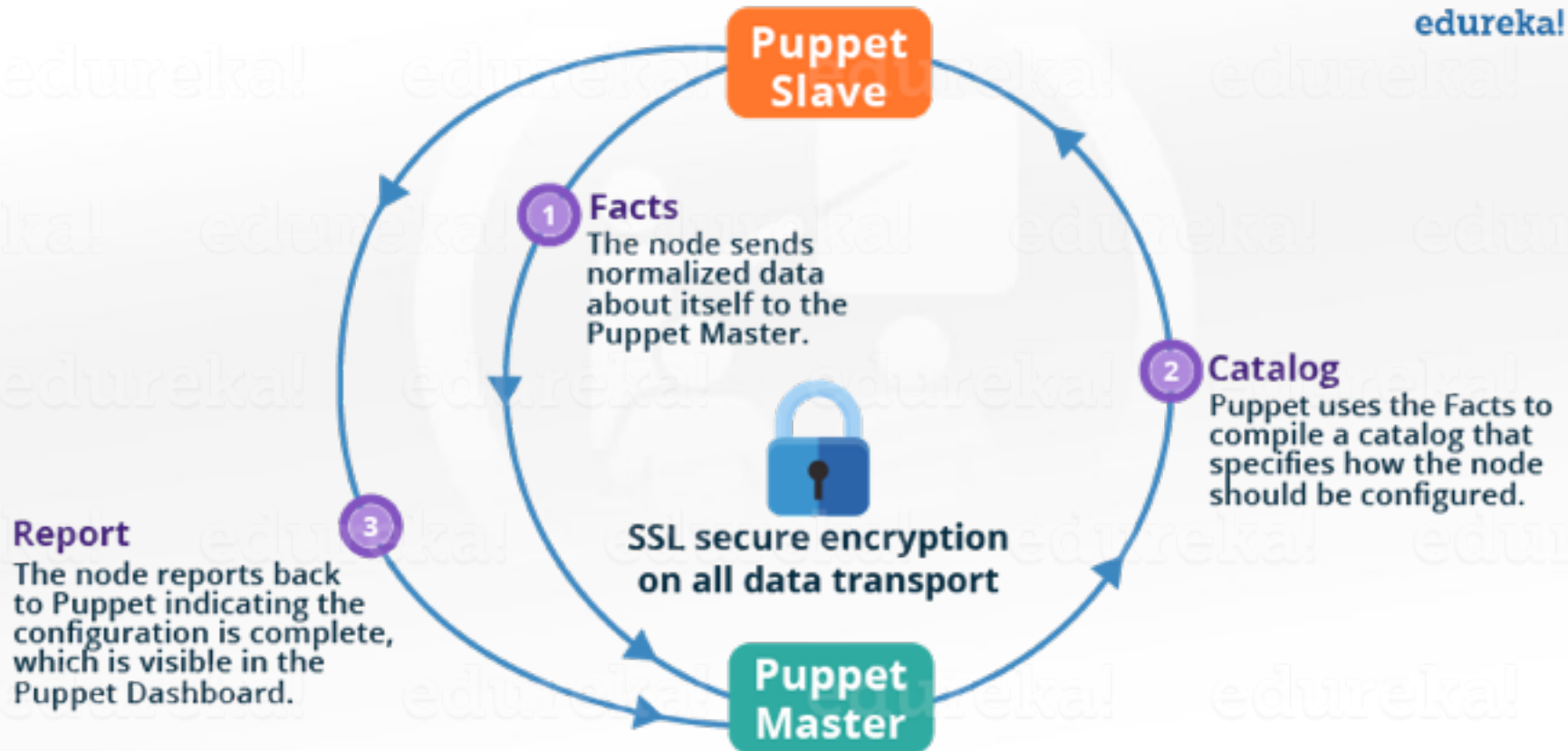
- *Puppet*
- *Chef,*
- *CFEngine,*
- *Salt,*
- *Ansible*



- *Outil de gestion de configuration*
- *Un framework d'automatisation des systèmes*
- *Langage DSL (Domain Specific Language)*
- *Logiciel Open source écrit en Ruby*
- *Compatible plusieurs plateformes*



● Gestion centralisée de la configuration





Puppet : installation

● *Debian, Ubuntu*

- *apt-get install puppet # clients (nodes)*
- *apt-get install puppetmaster # server (master)*

● *RedHat, Centos, Fedora*

- *yum install puppet # clients (nodes)*
- *yum install puppet-server # server (master)*
- *Add EPEL repository or RHN Extra channel*



- *Une ressource est représentée par*
 - *type (package, service, file, user, mount, exec ...)*
 - *A title (how is called and referred)*

```
type { 'title':  
  argument => value,  
  other_arg => value,  
}
```

```
file { 'motd':  
  path    => '/etc/motd',  
  content => 'Tomorrow is another day',  
}
```




Installation des ressources

● *Installation de OpenSSH*

```
package { 'openssh':  
  ensure => present,  
}
```

● *Démarrage d'apache*

```
service { 'httpd':  
  ensure => running,  
  enable => true,  
}
```



Définition des ressources

● *Classes définissent les modèles de ressources*

```
class mysql (  
  root_password => 'default_value',  
  port          => '3306',  
) {  
  package { 'mysql-server':  
    ensure => present,  
  }  
  service { 'mysql':  
    ensure  => running,  
  }  
  [...]  
}
```



● Installation de mysql (à partir d'un module)

```
[root@PuppetMaster ~]# puppet module install puppetlabs-mysql --version 3.10.0
Notice: Preparing to install into /etc/puppet/modules ...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/modules
├─ puppetlabs-mysql (v3.10.0)
├─ puppet-staging (v2.0.1)
└─ puppetlabs-stdlib (v4.13.1)
```

● Ajout de la ressource au manifest(*.pp)

- `include '::mysql::server'`

● Appliquer la modification dans le slave

```
[root@PuppetSlave /]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
```

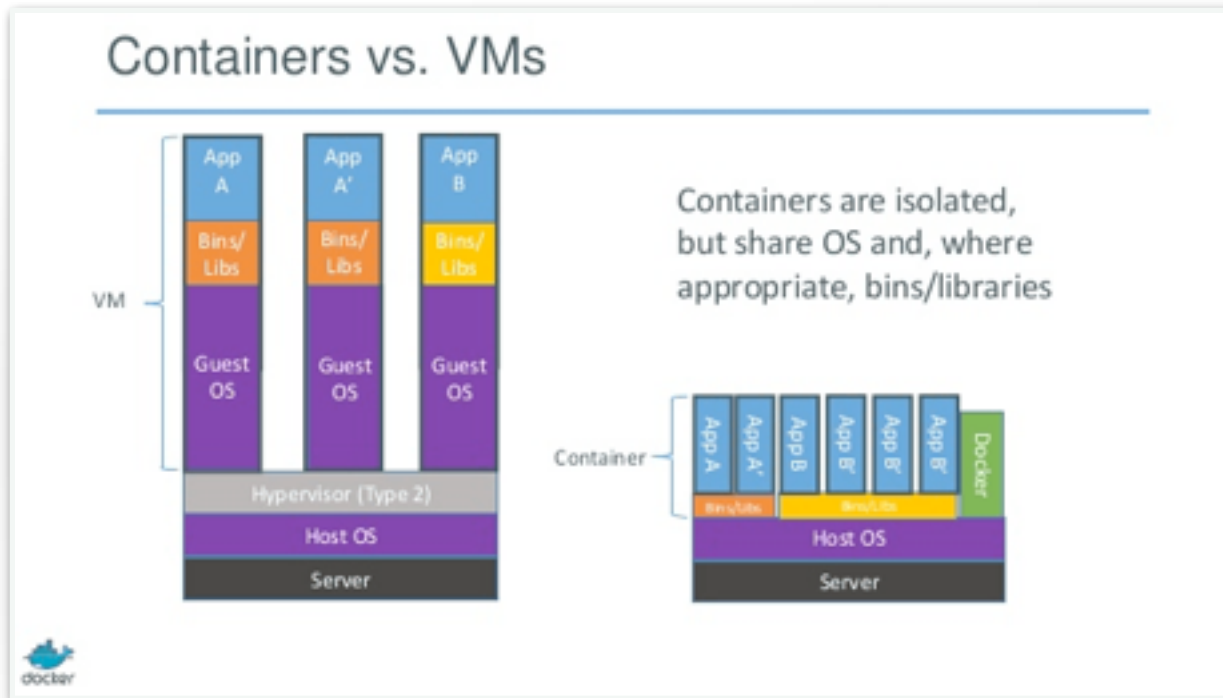
- Les salves contactent le master tous les 30min

Docker





Virtualisation applicative





- *Isoler un processus et ses dépendances dans un conteneur*
- *Imposer des limites de ressources (fichiers, réseaux)*



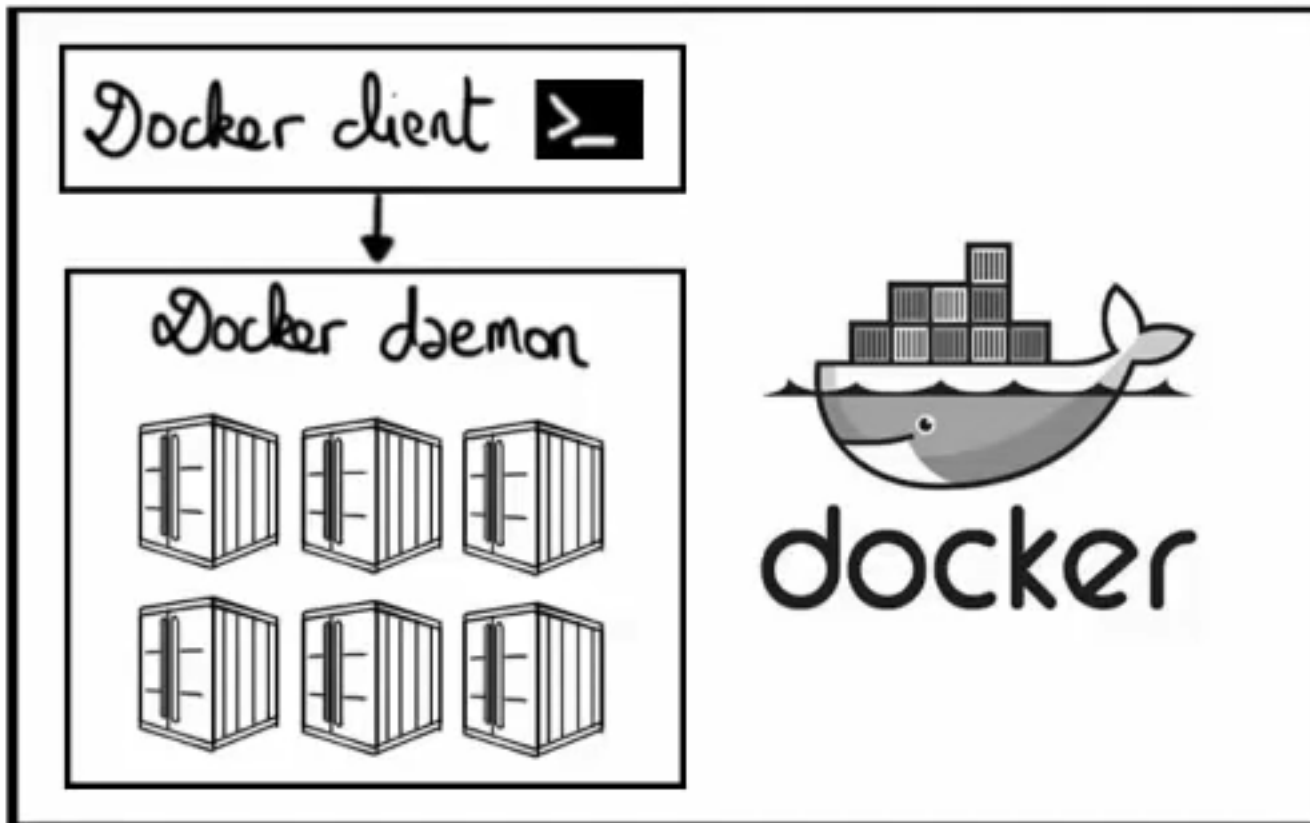
- *Une VM est provisionnée en secondes/minutes (le temps de démarrer l'OS guest)*
- *Un conteneur est lancé en millisecondes*
- *«Saying a container is a light VM is like saying a folder is a light partition» Thomas Maurice*



- *Contrôle et limitation de la consommation des ressources (CPU, RAM, I/O)*
- *Isolation réseau (IP, routage, firewall unique par conteneur)*
- *Isolation filesystem*
- *Isolation des utilisateurs/groupes*
- *Isolation des process*

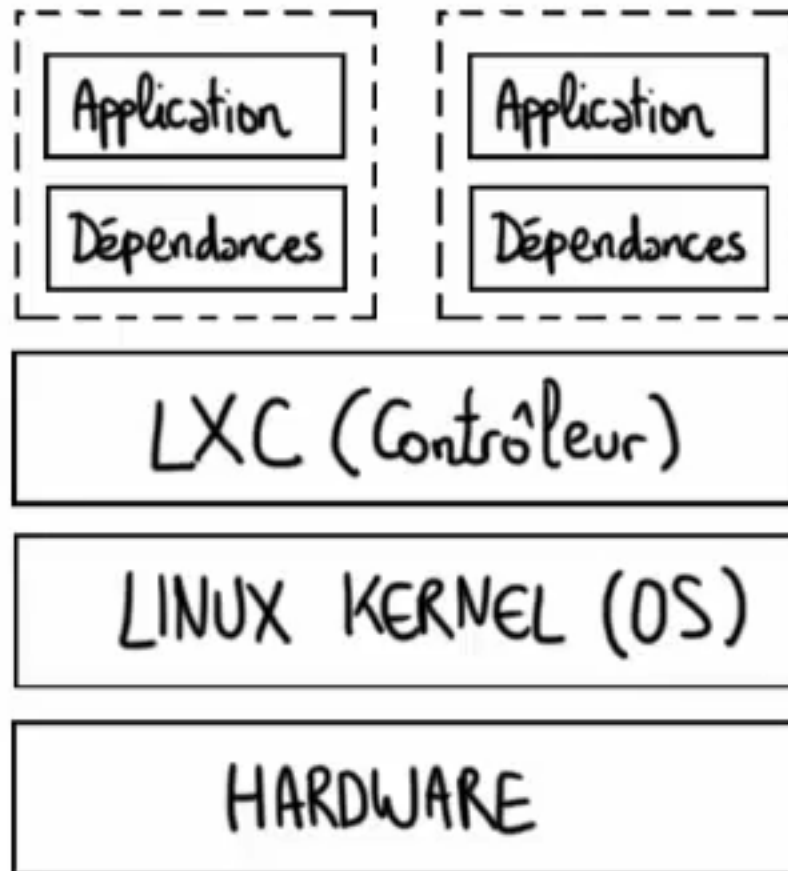


Architecture



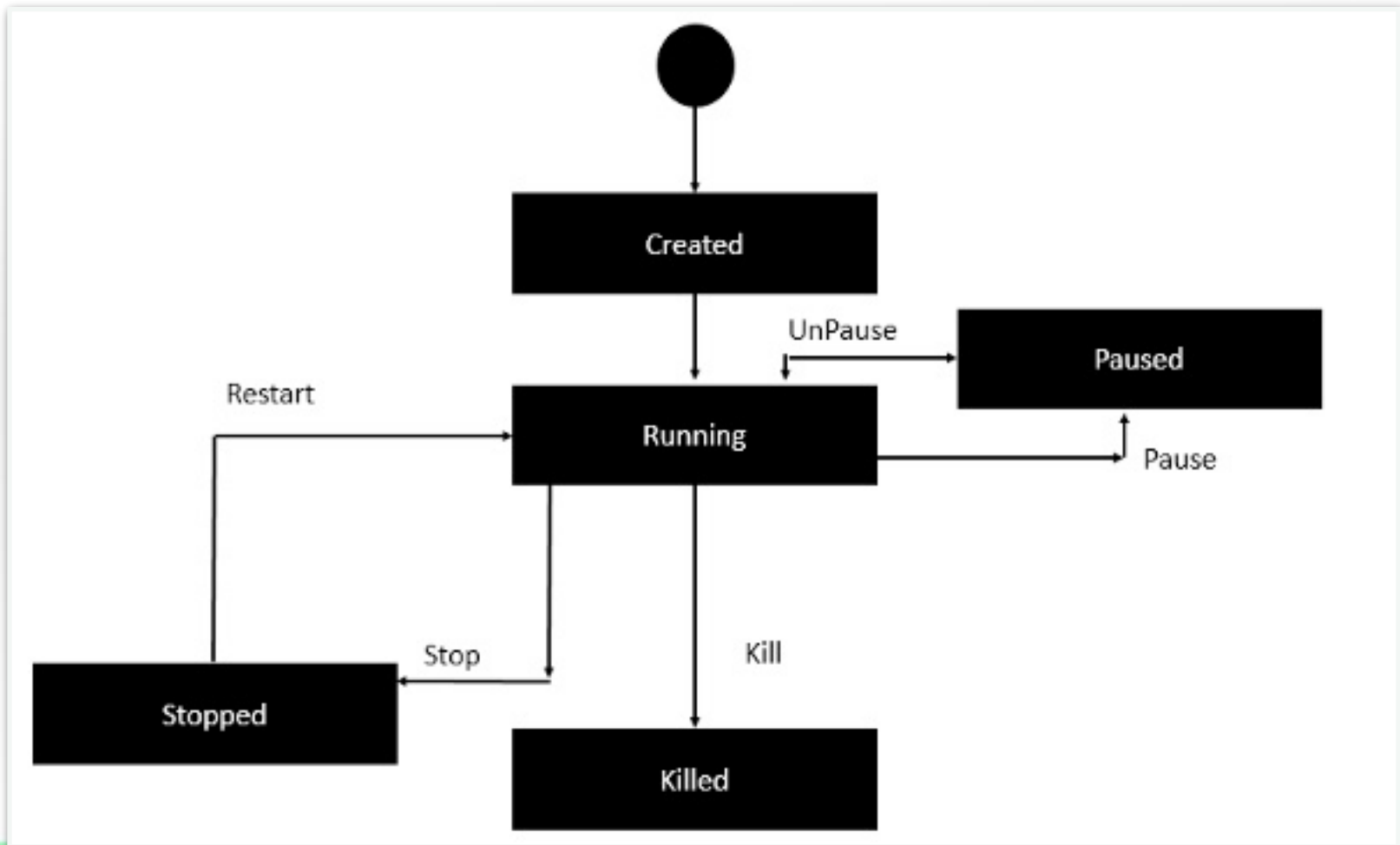


● Architecture





● Cycle de vie d'un conteneur Docker





● Build

- *Permet de construire une image immuable d'une application et de ses dépendances.*
- *Une fois l'image construite, elle garantit la même configuration dans tous les environnements: c'est la même image en dev, en preprod ou en prod.*



- *«Build. Ship. Run. Any app. Anywhere.»*



- *Ship : Hébergement de l'image dans un:*
 - *annuaire publique*
 - *ou privée*
- *Annuaire publique: certains peuvent écrire, tout le monde peut lire*
- *Annuaire privée: certains peuvent écrire, certains peuvent lire*



- *Run : Démarrer l'application dans un conteneur:*
 - *Isolation de l'application des autres processus*
 - *Isolation réseau*
 - *Isolation filesystem*
 - *Isolation des users/groups*
 - *Isolation des process*
 - *Limitation de ressources (RAM, CPU, I/O)*



● *Any app*

- *Binaire go compilé statiquement*
- *Elasticsearch*
- *Python*
- *bash*
- *C++*
- *Application graphique*



● *Anywhere*

● *Docker est installable sur*

- *Linux x64/ARM*
- *MacOS*

● *Tout type de serveur:*

- *Serveur dédié*
- *VM*
- *RaspberryPi*



● *Image Docker*

- *Une image Docker est un empilement de couches non modifiables (read-only layers) dans un système de fichier*
- *Chaque de ces couches contient un ensemble de modifications apportées par rapport à la couche précédente*
- *Une image est non modifiable*
- *Une image ne possède pas d'état*
- *Une image est utilisée pour lancer un conteneur*
- *Une image peut être utilisée comme image de base d'une autre image*



- *Layer 10: Déclaration du port TCP exposé dans le conteneur*
- *Layer 9 : Définition du script lancé au démarrage du conteneur*
- *Layer 8 : Installation des dépendances de l'application*
- *Layer 7 : Définition de `/usr/src/app` comme répertoire courant*
- *Layer 6 : Copie du code de l'application dans `/usr/src/app`*
- *Layer 5 : Chown récursif de `/usr/src/app` pour `myappuser:myappgroup`*
- *Layer 4 : Création d'un utilisateur/groupe `myappuser:myappgroup`*
- *Layer 3 : Création du répertoire `/usr/src/app`*
- *Layer 2 : Installer dépendances système*
- *Layer 1 : Image de base: Python 3.5*



● Dockerfile

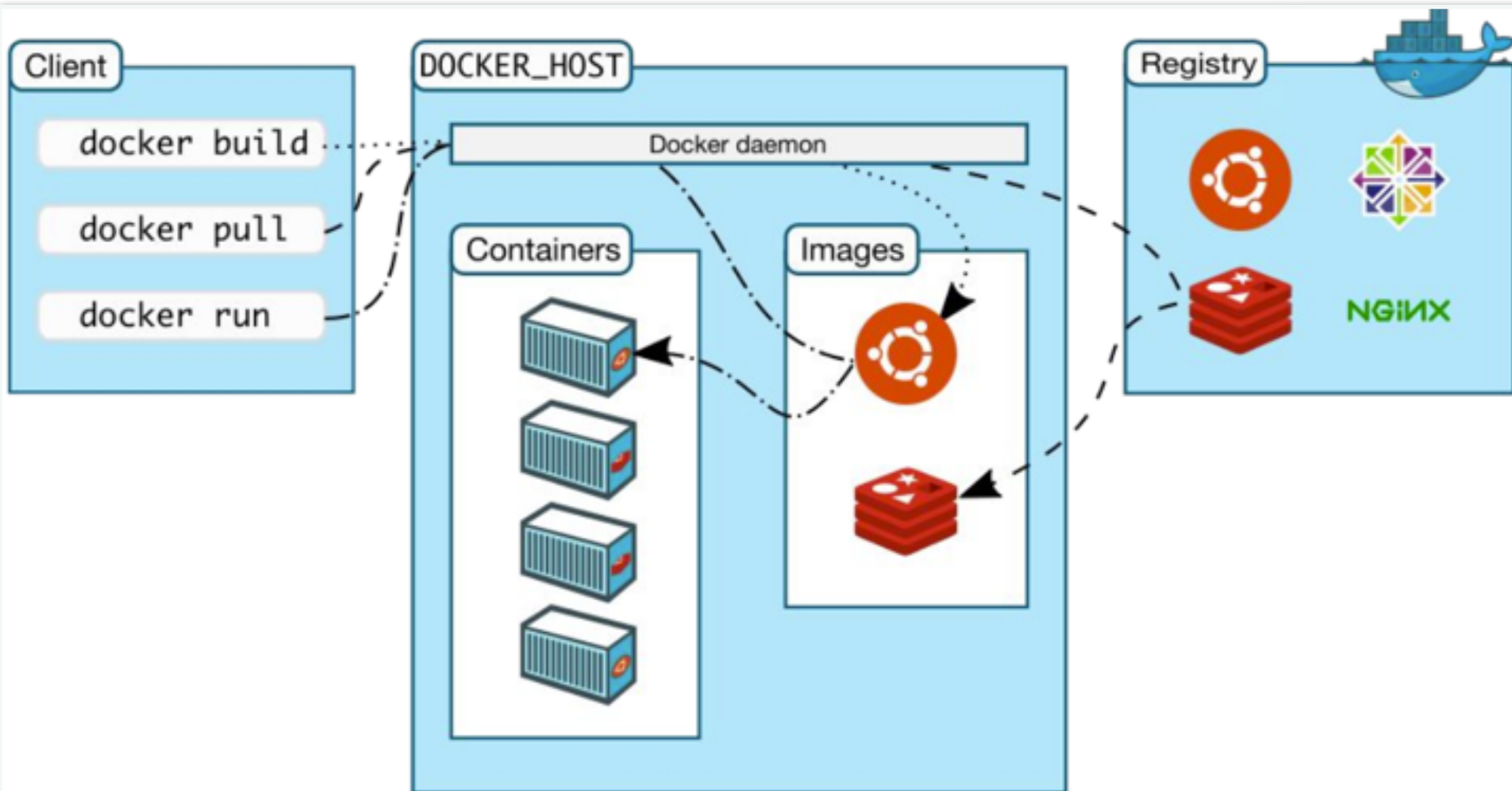
```
FROM python:3.5
RUN apt-get update && apt-get install -y libffi-dev
RUN mkdir -p /usr/src/app
RUN groupadd myappgroup && \
    useradd --create-home --home-dir /home/myappuser --uid 4242 -g \
        myappgroup myappuser
RUN chown -R myappuser:myappgroup /usr/src/app
COPY . /usr/src/app
WORKDIR /usr/src/app
RUN pip install requirements.txt
ENTRYPOINT docker-entrypoint.sh
EXPOSE 5000
```



● Conteneur

- *Empilement d'une image et d'une couche modifiable (writable layer) non persistée*
- *La couche modifiable contient les modifications apportées par l'utilisateur et l'application*
- *Lancé via docker run*

Docker



Gestion des sources





Définition

- *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*



● A l'origine (70-80)

- *Gestion manuelle des artefacts*
- *Démarche complexe, fastidieuse et souvent source d'erreurs*
- *Nécessité d'un logiciel gérant le référentiel des modifications*



● *Logiciel de gestion de versions*

- *Assure le stockage chronologique des fichiers*
- *Conserve l'intégralité des versions d'un fichier ou d'un répertoire*
- *Retrouve les révisions d'une ressource*
 - *fichier,*
 - *image...*



- *Logiciel de gestion de versions est généralement constitué d'un dépôt (local, distant) :*
 - *contenant toutes les versions,*
 - *de copies de travail*
 - *contenant les modifications d'un utilisateur qui seront ensuite incluses dans le dépôt.*



- *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*



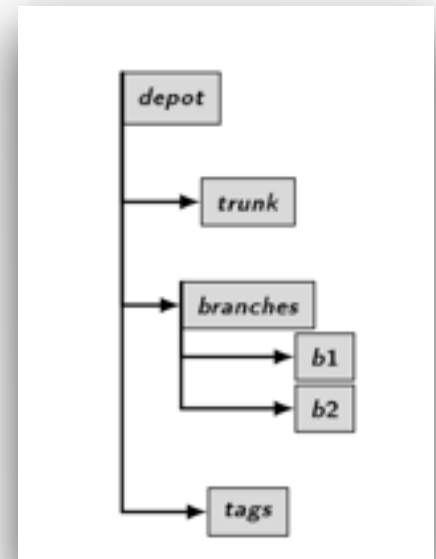
- *Disposer de 2 (ou plusieurs) versions de travail (branches)*

- *correction des bugs en parallèle au développement de nouvelles fonctionnalités*

- *Disposer d'un (ou plusieurs) dépôt(s) de référence*



- *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,*
 - *à fusionner après une divergence.*



- *Les tags sont des marques symboliques*
- *Ils permettent de définir les versions du projet*
- *Ils servent à marquer les jalons de livraison importants*
- *Ils permettent de marquer l'état des artefacts du projet*

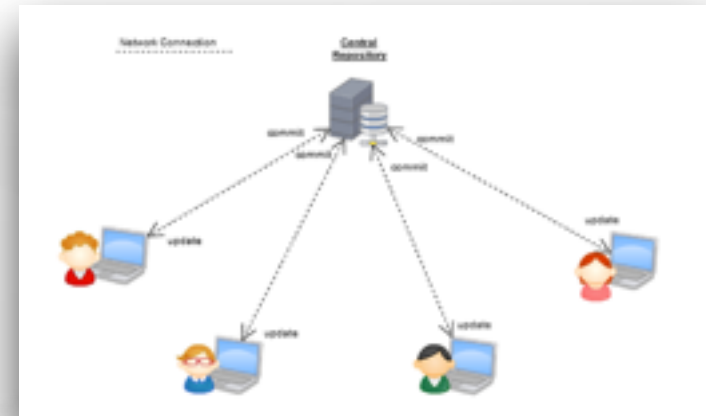


Les différentes familles d'outils

● 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:

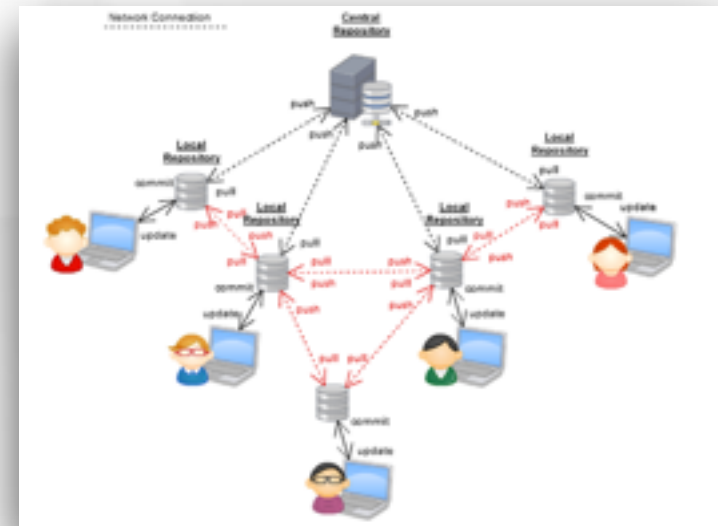
● Exemple : CVS, SVN





Les différentes familles d'outils

- *Version décentralisée*
 - *Plusieurs dépôts coexistent*
 - *Nécessité de synchronisation*
 - *Nécessité pour les utilisateurs de définir des règles de fonctionnement*
- *Exemple :*
 - *git, mercurial, darcs, svn*





● CVS correspond : Conccurent Version System

- Il s'agit du 1er outil de gestion open source de versions concurrentes
- Il fonctionne en mode Client-Serveur en mode centralisé
- Intégré aux différents outils de développement : Eclipse, NetBeans
- Site officiel : cvs.nongnu.org
- Actuellement en perte de vitesse:
 - Renommage et copie non géré
 - Commit non atomique



● *Subversion*

- *conçu pour remplacer CVS*
- *projet initié en 2000 par CollabNet*
- *le 14 février 2010, SVN est devenu officiellement un logiciel de la Fondation Apache*





● *Il enrichi les fonctionnalités de CVS*

- *Renommer et copie des sources*
- *Commit atomique*
- *Branches et tags par copie*
- *Méta-données (permissions)*



● Principales commandes SVN

Commande	Signification
add	Déclare l'ajout d'une nouvelle ressource pour le prochain commit.
checkout (co)	Récupère en local une version ainsi que ses méta-données depuis le dépôt.
cleanup	Nettoie la copie locale pour la remettre dans un état stable.
commit (ci)	Enregistre les modifications locales dans le dépôt créant ainsi une nouvelle version.
copy	Copie des ressources à un autre emplacement (localement ou dans le dépôt).
delete	Déclare la suppression d'une ressource existante pour le prochain commit (ou supprime directement une ressource du dépôt).
diff	Calcule la différence entre deux versions (permet de créer un patch à appliquer sur une copie locale).
lock	Verrouille un fichier.
log	Donne les messages de commit d'une ressource.
merge	Calcule la différence entre deux versions et applique cette différence à la copie locale.
move	Déclare le déplacement d'une ressource.
resolved	Permet de déclarer qu'un conflit de modifications est résolu.
revert	Revient à une version donnée d'une ressource. Les modifications locales sont écrasées.
status (st)	Indique les changements qui ont été effectués.
switch	Bascule sur une version/branche différente du dépôt.
update (up)	Met à jour la copie locale existante depuis la dernière version disponible sur le dépôt.
unlock	Retire un verrou.



● Quelques bonnes pratiques

- *Testez avant de valider (commit) une modification*
- *Mettre à jour le référentiel local avant la validation (commit)*
- *Assurez la synchronisation avec le dépôt central autant que possible*
- *Créez une nouvelle branche spécifique pour corriger les bugs éventuels*



● Quelques bonnes pratiques(suite)

- *Mettre à jour le référentiel local avant toute modification*
- *Apportez des commentaires détaillés aux mise à jour*
- *Envoyer les modifications des sources au moins une fois par jour.*
- *Marquer dans le dépôt les versions des sources livrées.*
- *Mettre régulièrement ses sources à jour*

Les tests





Les tests - Intérêts

- *Les tests sont un élément indispensable à l'obtention d'un logiciel de qualité*
- *Ils garantissent que l'application répond bien aux besoins du client*
- *Ils garantissent que l'application ne comporte pas d'anomalies*
- *Ils garantissent que les évolutions ne génère pas de régressions*



Les différents types de tests

● *Il existe différents types de tests*

- *Tests unitaires*
- *Tests d'intégration*
- *Tests fonctionnels*
- *Tests de non regression*
- *Tests de charge*



Tests fonctionnels

- *Les tests fonctionnels ont pour objectifs :*
 - *valider que le logiciel répond aux attentes du client*
 - *garantir la satisfaction des clients*
- *Les tests sont réalisés dans un environnement le plus proche possible de l'environnement utilisateur cible*
- *Cette phase de test est primordiale et représente le minimum à mettre en œuvre pour un logiciel de qualité*



● *Ils s'occupent de valider l'interface entre l'utilisateur et l'application :*

- *Interface graphique*
- *Les commandes (script)*
- *Les données (interopérabilité)*
- *Les services*



Tests fonctionnels - démarches

- *Il existe plusieurs démarches pour réaliser les tests fonctionnels :*
 - *Démarche basée sur l'élaboration d'un référentiel de tests qui seront exécutés par n'importe quel testeur*
 - *Démarche basée sur équipe dédiée*



Tests fonctionnels - démarches

- *La démarche du référentiel consiste à structurer, organiser et référencer les tests afin d'être en mesure de connaître clairement les tests permettant de valider une fonctionnalité et de les rejouer facilement, de manière déterministe, par n'importe quel testeur*
- *Elle présente les avantages suivants :*
 - *La reproductibilité des tests,*
 - *Mutualisation des cas de tests,*
 - *Association fonctionnalité/cas de test*
 - *Degré faible de formation*



Tests fonctionnels - démarches

- *Une démarche basée sur une équipe dédiée peut s'avérer plus efficace sans l'usage d'un référentiel*
- *Les testeurs peuvent se comporter comme des utilisateurs réels et donc peuvent détecter des anomalies non prévues.*
- *Le processus de validation repose alors sur les personnes et donc l'effort de formation est plus important*



Tests unitaires

- *Les tests unitaires, concernent les éléments de base d'une application à savoir :*
 - *les méthodes,*
 - *les classes ou interfaces*
 - *Composants*
- *Ils permettent de détecter et corriger les erreurs à faible coût*



- *Les tests unitaires utilisent couramment des Mocks «bouchons »*
- *Ils sont nécessaires à toute stratégie de qualité*
 - *Ils permettent de détecter et isoler les anomalies*
 - *Ils permettent de détecter au plutôt les anomalies et ainsi de limiter leurs impact*



Tests d'intégration

- *L'objectif est d'identifier les dysfonctionnements lors de l'intégration des différents modules d'une application ou l'utilisation des services tiers*
- *Tester les implémentations réelles à la place des Mocks*
- *Ils permettent de*
 - *créer une version complète et cohérente d'un logiciel*
 - *garantir le bon fonctionnement du logiciel dans l'environnement cible*



Tests de charge

- *L'objectif est garantir la robustesse de l'application en analysant son comportement*
- *Les tests de charge simulent, via des injecteurs, le comportement d'un nombre défini d'utilisateurs*



- *Ils se basent sur des logiciels*
 - *simulant les actions des utilisateurs et*
 - *avec des jeux de données proches de données de production*
- *Il est recommandé de les conduire le plutôt pour détecter au plus vite les différents types d'erreurs.*
 - *de conception, d'implémentation ou de dimensionnement de l'infrastructure.*



Test 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*



Test de non régression

- *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*

Framework JUnit





● *Procédure de test classique d'une application Java*

- *Evaluer les expressions du programme dans un débogueur (pdb)*
- *Evaluer les résultat affichés sur la sortie standard*

● *Les limites de la démarche :*

- *Elles requièrent un jugement humain pour analyser les résultats*
- *Un seul test peut être exécuté à la fois*
- *Trop d'informations affichés sur la sortie standard ce que rend l'interprétation des résultat illisibles*



- *Junit : un framework permettant de réaliser des tests unitaires*
 - *Pas besoin d'interprétation humaine des résultats*
 - *De nombreux tests peuvent être exécutés simultanément*



- *Les résultats des tests doivent être facilement utilisables*
 - *Succès => très synthétique*
 - *Echec/Erreur => détaillé*
- *Distinction entre*
 - *Échec => anticipé*
 - *Erreur => imprévue*
- *Si une méthode peut renvoyer une ou plusieurs exceptions :*
 - *Une méthode de test pour le cas normal*
 - *Une méthode de test par exception potentiellement levée.*



Réalisation d'un test

- *Etapes à suivre pour réaliser un test avec JUnit < 4*
 - *Créer une sous classe de TestCase*
 - *Implémenter les méthodes testXxx()*
 - *Pour vérifier une condition, utiliser la méthode assertXxx()*



Réalisation d'un test

- *Etapes à suivre pour réaliser un test avec JUnit ≥ 4*
 - *Créer une classe de test (qui n'étend aucune classe parente)*
 - *Implémenter les méthodes de tests (void sans paramètres) en les annotant avec l'annotation `@Test`*
 - *Pour vérifier une condition, utiliser les méthodes `assertXxx()`*



● Exemple: calcul de la surface d'un cercle

```
@Test // seulement en JUnit >=4
    public void testCalculerSurface() {
        int s = 4;
        int result = cercleI.calculerSurface( param);
        assertEquals(result, s );
    }
```

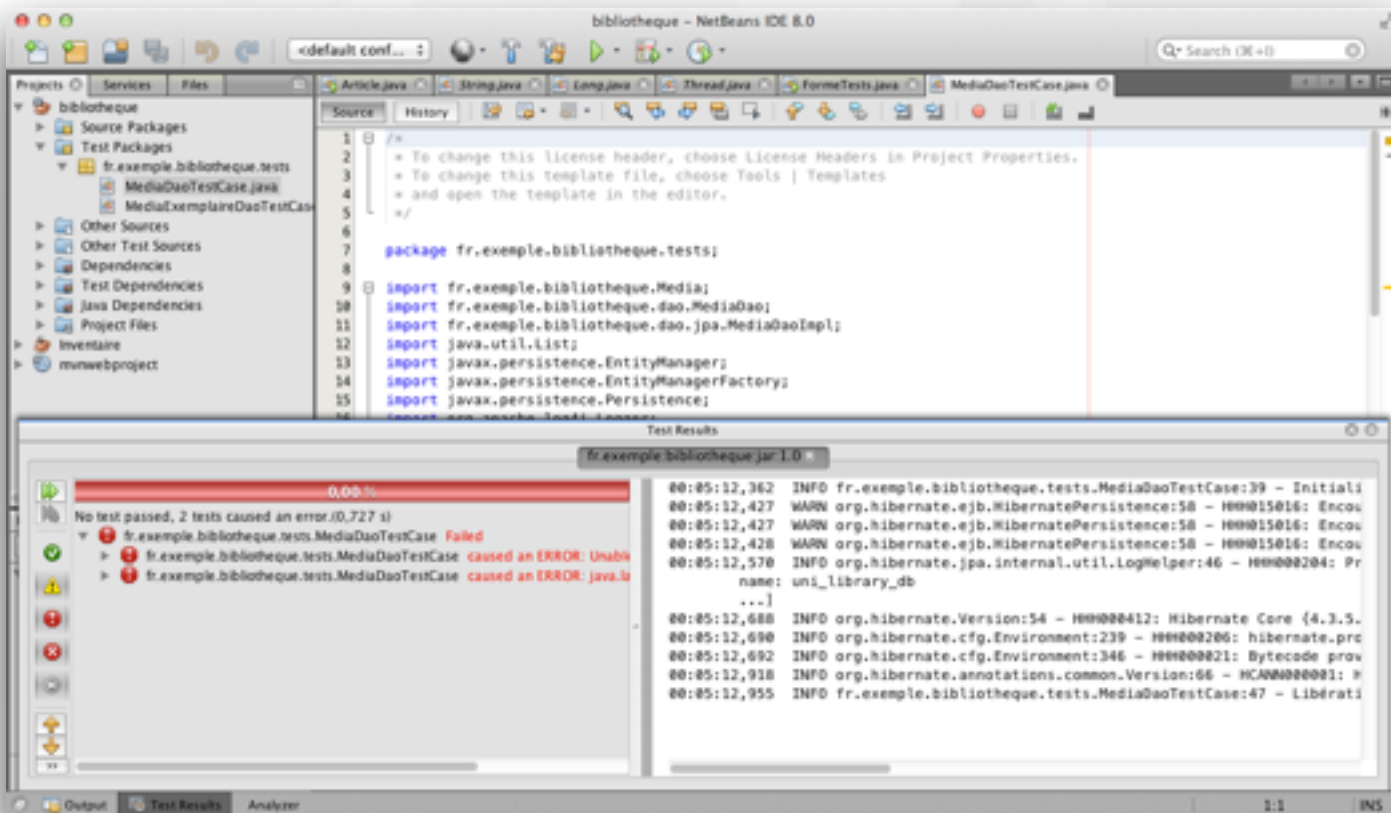


- *Plusieurs procédures existent pour le lancement des tests unitaires :*
 - *Le développeur sélectionne un test ou un ensemble de tests et les exécute dans L'IDE*
 - *Le framework ant d'apache*
 - *Le framework Maven d'apache*
 - *Intégration continue*



Execution des tests

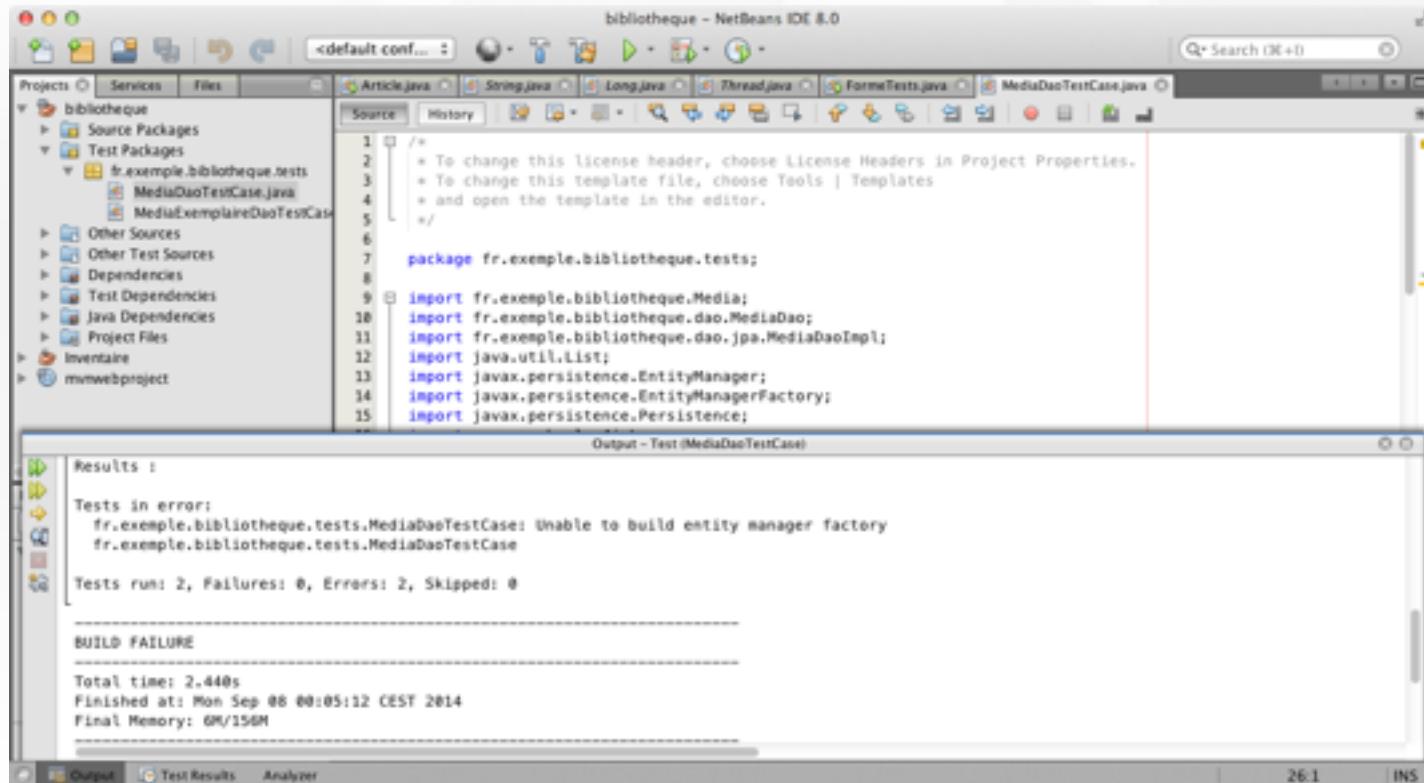
- La figure ci-dessous présente la perspective d'exécution des tests unitaires en NetBeans





Exécution des tests

- La figure ci-dessous présente le rapport de l'exécution des tests avec maven





- *Lancement via l'intégration continue*
 - *Les tests sont exécutés automatiquement par un serveur d'intégration continue à intervalle régulier*
 - *Les résultats sont historisés et accessibles à tout le monde via une interface web.*



- *Le plutôt les tests seront mis en œuvre, plus vite une éventuelle erreur sera détectée et plus rapidement les correctifs pourront être mis en œuvre en minimisant leur impact.*
- *Exemple d'erreurs*
 - *Erreur de conception,*
 - *Erreur d'implémentation ou*
 - *le dimensionnement de l'infrastructure*



- *Les tests augmentent de 30% la charge d'un projet*
- *Il existe des moyens pour*
 - *augmenter la productivité des tests et de développement*
 - *de faire des tests l'élément central du développement des logiciels.*



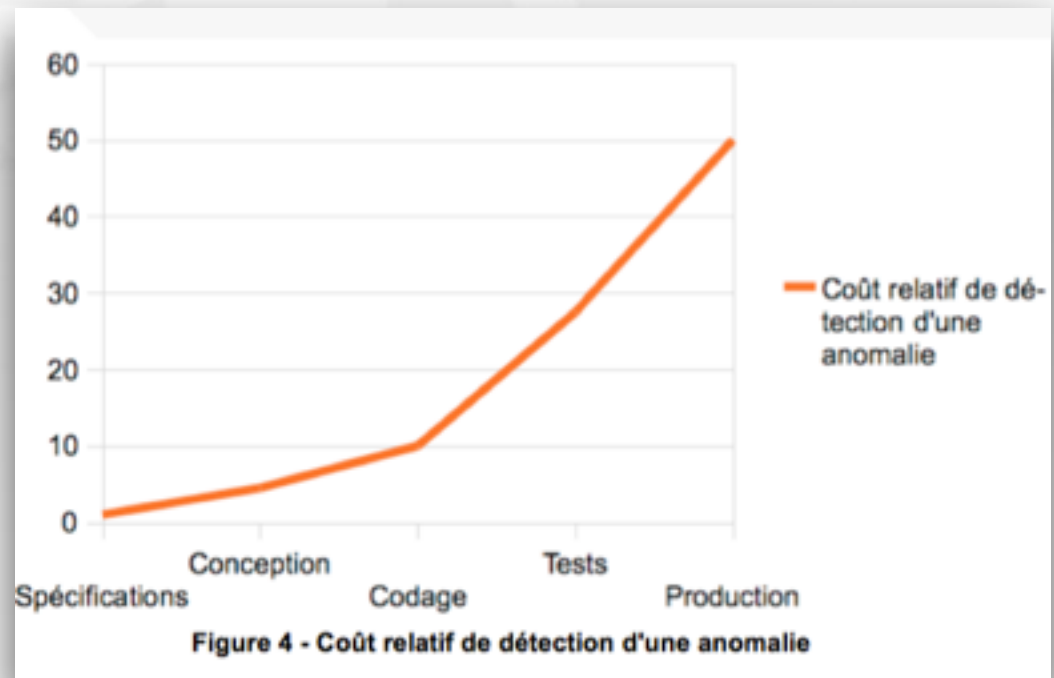
● Selon G. J. Myers :

- *un bon test est un test qui essaye d'identifier une anomalie ou un défaut (The Art of Software Testing, 1979).*
- *Un test élaboré pour trouver des bugs a une valeur, qu'il en trouve ou non.*
- *Un test fait pour "réussir / passer" n'a pas de valeur.*



Qualité des tests

- *La qualité des tests dépend de la précocité de leur mise en oeuvre*
- *Plus une anomalie est détectée tôt, plus elle sera facile à corriger*





Qualité des tests - couverture

- *Que valent mes tests, quantitativement et qualitativement ?*
- *Quelle est leur couverture technique et fonctionnelle ?*
- *Les tests doivent représenter ce que les utilisateurs vont réaliser*
- *L'aspect couverture peut concerner*
 - *couverture du code*
 - *couverture fonctionnelle*



Qualité des tests - couverture

● Couverture du code

- *mesurer, de manière automatique, la couverture de code réalisé par la campagne de tests*
- *mesurer le pourcentage de code traversé par les tests par rapport à l'ensemble du code écrit*

● *Elle permet d'identifier au plus tôt les éventuels manques des tests unitaires*



Qualité des tests - couverture

- **Couverture fonctionnelle :**

- *Il s'agit de s'assurer que l'on a testé toutes les fonctionnalités, règles métier du logiciel*

- **Il faut maintenir un référentiel comportant :**

- *d'un côté les différents cas d'utilisation, règles métier, fonctionnalités du logiciel et*
 - *de l'autre les scénarios de tests*

Test Driven Development





Test Driven Development

- *Cette démarche à pour objectif de répondre aux questions suivantes :*
 - *Comment tester le plus tôt possible une application?*
 - *Avec une couverture maximale ?*



Test Driven Development

- *La méthode consiste à définir les tests en amont de la phase de réalisation*
- *Pour chaque fonctionnalité, il faut définir les tests appropriés permettant de valider son bon fonctionnement*
- *Le principe est de créer le test avant la définition de la mise en oeuvre de la nouvelle fonctionnalité*
 - *le test doit échouer avant l'implémentation*
 - *le test doit aboutir après l'implémentation*



Test Driven Development

- *Les avantages de la démarche sont multiples:*
 - *Préciser les différents scénarios en rapport avec un comportement réel attendu*
 - *Meilleure conception : la conception émerge des tests et des cas concrets d'utilisation*
 - *La couverture des tests est maximale*
 - *Facilite le refactoring et la réalisation des tests de non régression*
 - *Améliore la qualité des applications : tout traitement testé est par la suite effectif et réutilisable*

Automatisation des tâches





- *Plusieurs tâches peuvent être automatisées afin d'améliorer la productivité*
 - *La compilation*
 - *L'exécution des tests*
 - *Le déploiement*
 - *La génération de la documentation*
 - *L'analyse du code*
 - *...*



Automatisation des tests

- *Minimise le surcoût des tests unitaires ou fonctionnels*
- *Permet de valider une fonctionnalité ou l'ensemble de l'application de manière automatique*
- *S'assurer que les modifications apportées n'ont pas engendré de régression dans le logiciel*
- *Détecter au plus tôt une anomalie et par conséquent d'en limiter l'impact et le coût*



Automatisation des tests

- *Le coût de l'élaboration des scripts d'automatisation sont rapidement rentabilisés grâce :*
 - *Le gain en terme de non régression*
 - *La précocité des détections d'anomalies*
- *On s'assure ainsi à moindre frais qu'un correctif n'engendre pas de défaillance annexe sur le système*
- *L'automatisation est un investissement, qui devient obligatoire si l'on fait des livraisons sur des cycles courts.*
 - *l'automatisation est rentable, en charge de tests, à partir de la 4ème campagne*

Outils d'automatisation

Maven





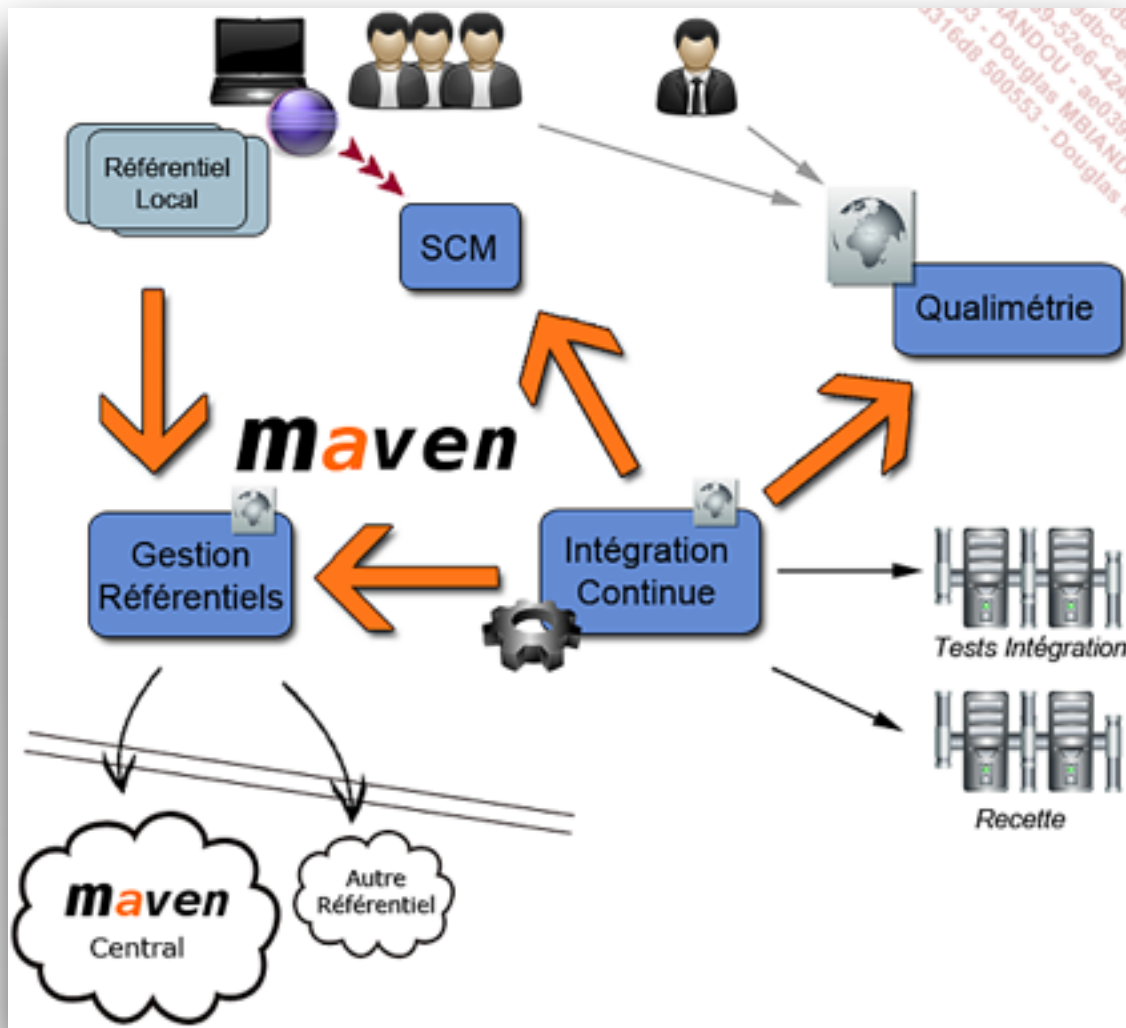
● *La notion de build*

- *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*

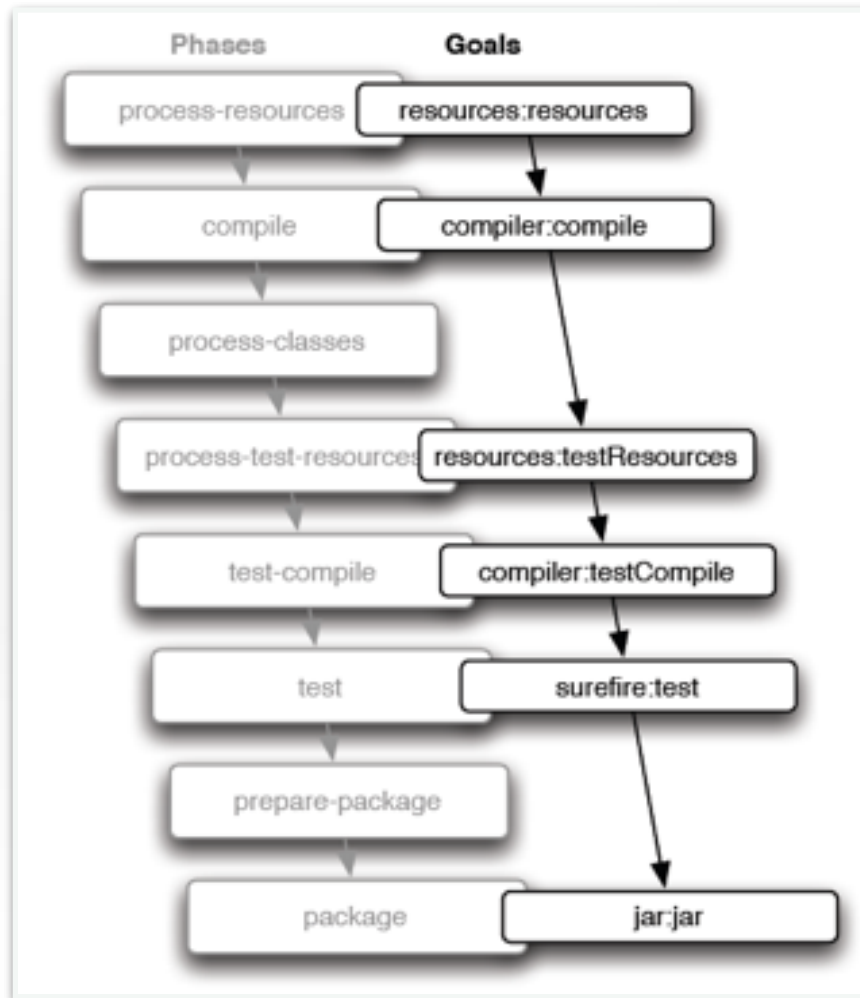


Architecture Maven





Phases , plugins et goals



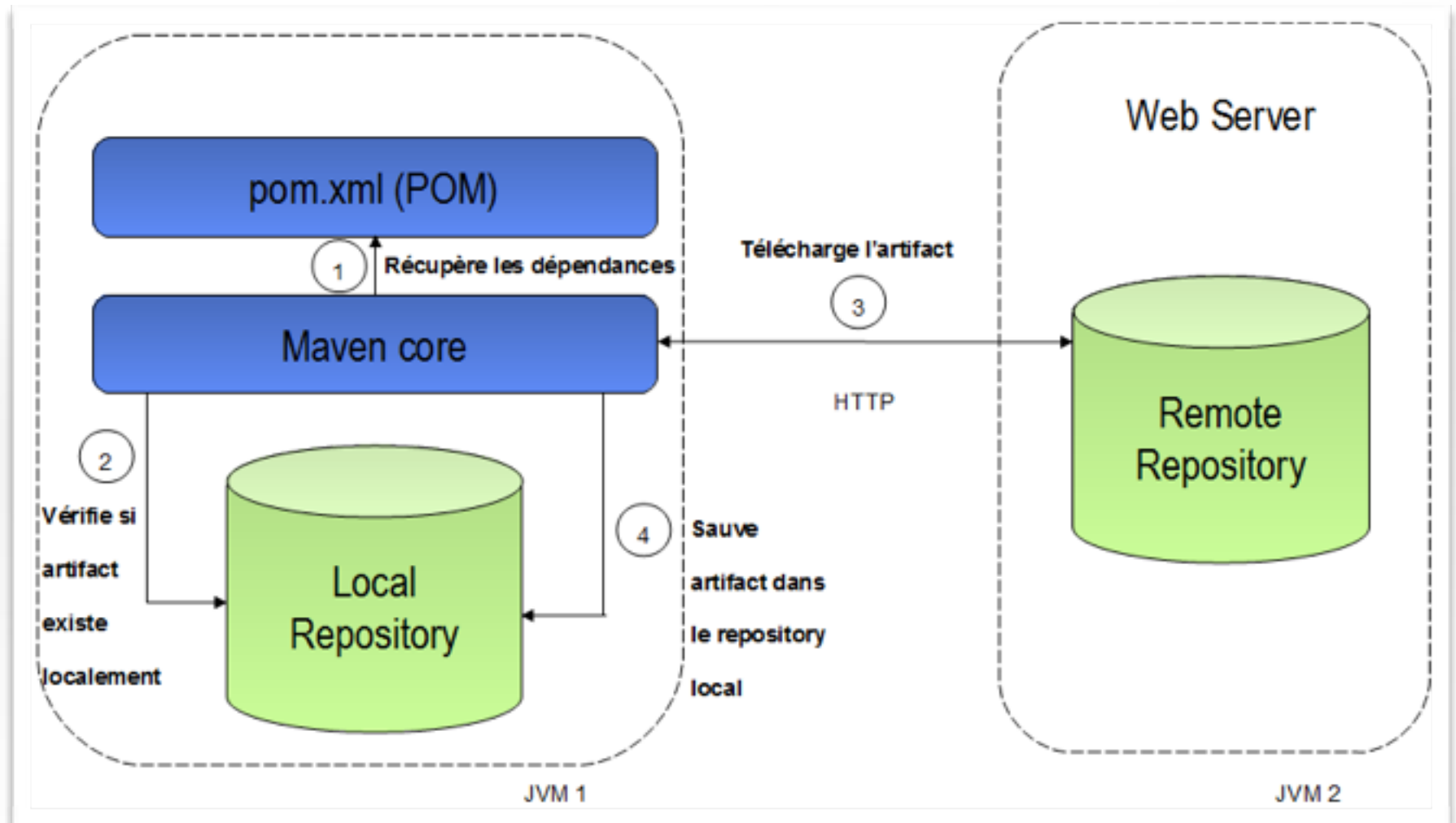


Maven - La gestion des dépendances

- *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*



Repository Remote / Local





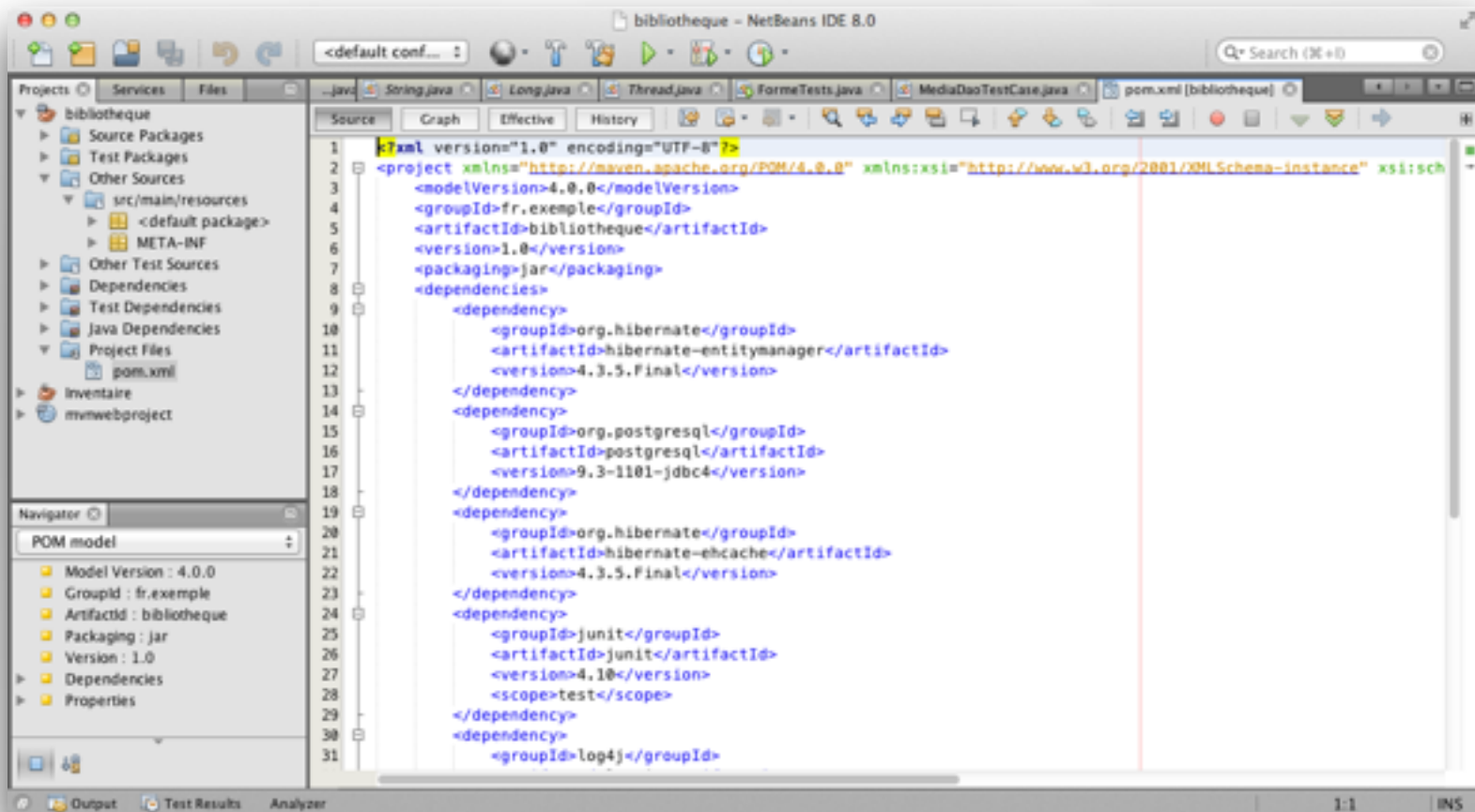
Maven - La gestion des dépendance

- *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

● Déclaration de quelques dépendances

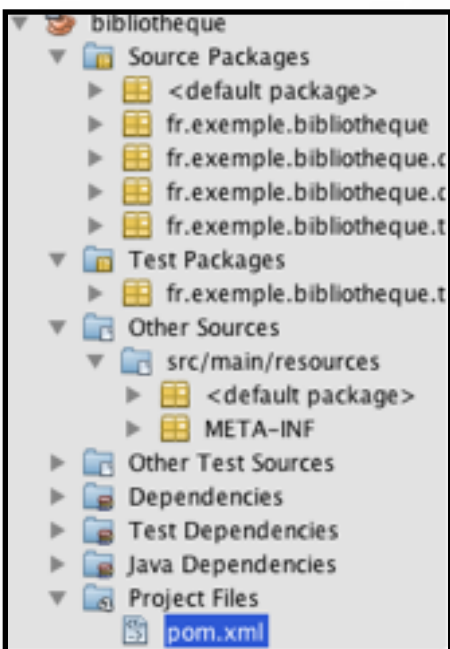




Maven - structure de projet

● Organisation des sources

● Maven propose d'avoir 4 répertoires de sources :



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 - mise en oeuvre

- *Utiliser Maven consiste à*
 - *Créer un projet de type maven*
 - *Renseigner les dépendances dans le fichier pom.xml à la racine du projet.*
 - *Utiliser un plugin maven si ce n'est pas déjà le cas (Q4E, MIA, m2Eclipse) avec eclipse*



Maven - mise en oeuvre

● *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 - mise en oeuvre

- *Les informations à renseigner sont les suivantes*
 - *un groupId,*
 - *un artifactId,*
 - *une version*
 - *un type de livrable (war, ear, jar)*
 - *la définition du build*
 - *un ensemble de dépendances*
 - *les plugins*



Maven - mise en oeuvre

- *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.*



Intégration de JUnit avec maven

- Ajoutez la dépendance de junit dans le fichier de configuration de maven POM :

```
<dependency>  
  <groupId>junit</groupId>  
  <artifactId>junit</artifactId>  
  <version>4.4</version>  
</dependency>
```



Maven et les webtools

- *Lorsque maven est utilisé avec les webtools :*
 - *bien penser à inclure maven en tant que j2ee module dependencies dans Eclipse, sans quoi les jar récupérés par maven ne sont pas pris en compte par les webtools.*

Gestion des configurations avec maven





- *Le Filtrage permet d'intégrer des variables dynamiques dans les ressources Maven*
- *Exemples :*
 - *log4j.properties, db.properties, hibernate.cfg.xml, context.xml*

```
log4j.rootLogger=${log.level}, CONSOLE_APP  
log4j.appender.console=org.apache.log4j.ConsoleAppender  
log4j.appender.console.layout=org.apache.log4j.PatternLayout  
log4j.appender.console.layout.ConversionPattern= %d %-4r %-5p %c %x - %m%n
```



- *L'activation se fait dans le fichier de configuration (pom)*

```
<build>
  . . .
  <resources>
    <resource>
      <directory>src/main/resources</directory>
      <filtering>true</filtering>
    </resource>
  </resources>
  . . .
</build>
```

- *mvn package -Dlog.level=DEBUG*



- *Les profils permettent de définir des configuration dynamiques*
- *Les profils permettent de créer des variations dans le cycle de construction, afin de s'adapter à des particularités, comme*
 - *Des construction pour des plateformes différentes*
 - *Tester en utilisant différentes DB*
 - *Référencer un Système de fichiers local*
 - *Etc...*
- *Les profils sont déclarés dans des entrées du POM*
 - *Ils peuvent être « activés » de différentes manières*



- *Les profils pourront être définis à deux endroits :*
 - *Le fichier settings.xml de la directory .m2 (repository)*
 - *Le POM lui-même*
- *Dans un de ces fichiers, vous pouvez définir les éléments suivants :*
 - *Repositories, pluginRepositories, dependencies, plugins, modules, reporting*
 - *dependencyManagement, distributionManagement*



● 3 procédure pour activer un profil :

- le sélecteur `-P` de la commande `mvn`

✱ `mvn -Pprofile1,profile2 install`

- via la section `activeProfile`, qui référence les profiles définis dans `settings.xml`, voir ci-dessous à droite
- via un « déclencheur » ou trigger, qui, si vérifié, activera le profile déclaré.

```
<profile>
  <id>profile1</id>
  ...
  <activation>
    <property>
      <name>environment</name>
      <value>test</value>
    </property>
  </activation>
</profile>
```

```
<settings>
  ...
  <profiles>
    <profile>
      <id>profile1</id>
      ...
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>profile1</activeProfile>
  </activeProfiles>
  ...
</settings>
```



Les profils

```
<profiles>
  <profile>
    <id>test</id>
    <properties>
      <db.url>jdbc:mysql://mon_serveur_de_test/db_test</db.url>
      <db.driver>com.mysql.jdbc.Driver</db.driver>
      <db.username>user1</db.username>
      <db.password>user1pwd</db.password>
    </properties>
    <activation>
      <property>
        <name>test</name>
      </property>
    </activation>
  </profile>
  <profile>
    <id>dev</id>
    <properties>
      <db.url>jdbc:mysql://localhost/db_dev</db.url>
      <db.driver>com.mysql.jdbc.Driver</db.driver>
      <db.username>root</db.username>
      <db.password />
    </properties>
    <activation>
      <property>
        <name>dev</name>
      </property>
    </activation>
  </profile>
</profiles>
```

Le serveur d'intégration continue

Jenkins





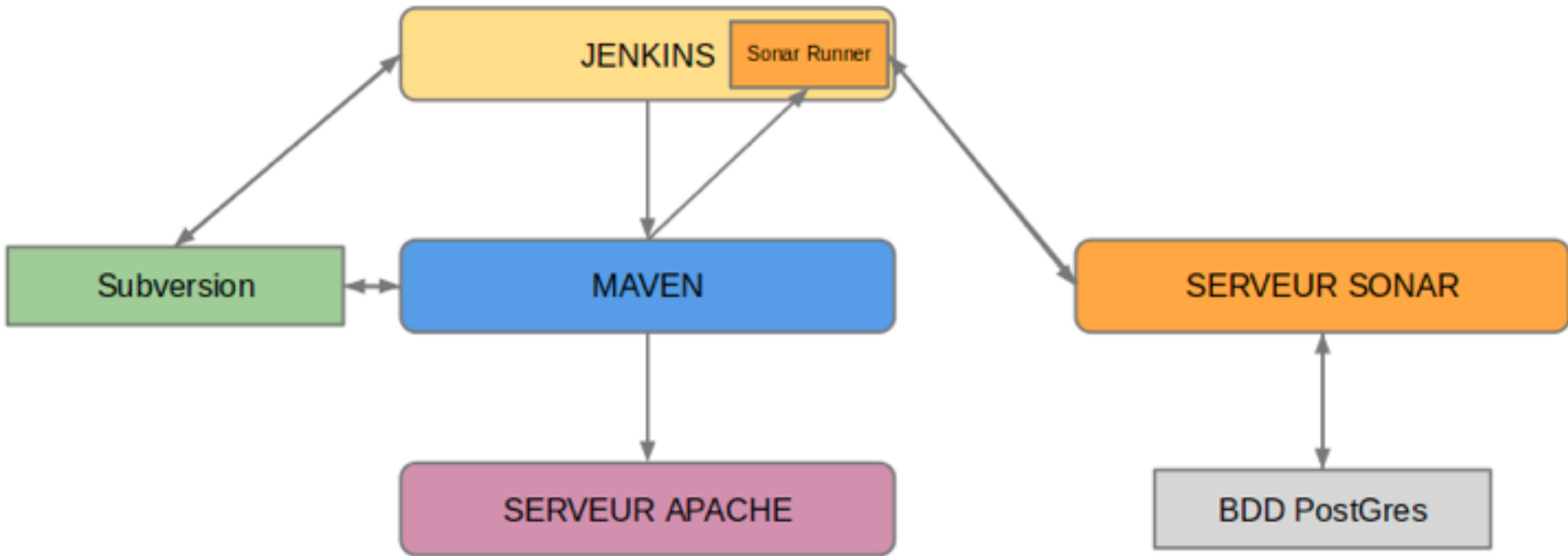
- *Jenkins anciennement appelé Hudson est un outil d'intégration crée par Kohsuke Kawaguchi*
 - *open source*
 - *Simple à utilisé*
 - *Ecrit en Java*
 - *Extensible (plugins)*
 - *Communauté très dynamique*
 - *Une Release tout les 3mois*
 - *Interface user friendly*



- *2004 : Initiation de projet Hudson en tant que loisir*
- *2008 : Sun charge l'auteur à se consacrer au projet en plein temps*
- *2010 : Meilleur outil d'IC*
- *2011 : projet renommé Jenkins*



Jenkins - architecture





● *Pré-requis*

- *Outil de contrôle de version : SVN, git*
- *Outil de gestion de tâches : ant, maven*

● *Téléchargement*

- *<http://jenkins-ci.org/>*

● *Installation Debian*

- *source <http://pkg.jenkins-ci.org/debian/>*
- *service : `/etc/init.d/jenkins`*



● System Debian

- `wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -`
- `sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'`
- `sudo apt-get update`
- `sudo apt-get install jenkins`



- *Procédure d'installation avec Tomcat*
 - *Installation de JDK*
 - *Configuration des variables d'environnement*
 - *JAVA_HOME*
 - *PATH*
 - *Installation de Tomcat*
 - *Déploiement de package war de jenkins dans le repertoire de déploiement webapps*
- *L'url :*
 - *<http://localhost:8080/jenkins>*



Jenkins - Page d'accueil

Jenkins

Jenkins ▶

 [New Job](#)

 [People](#)

 [Build History](#)

 [Manage Jenkins](#)

Build Queue

No builds in the queue.

Build Executor Status

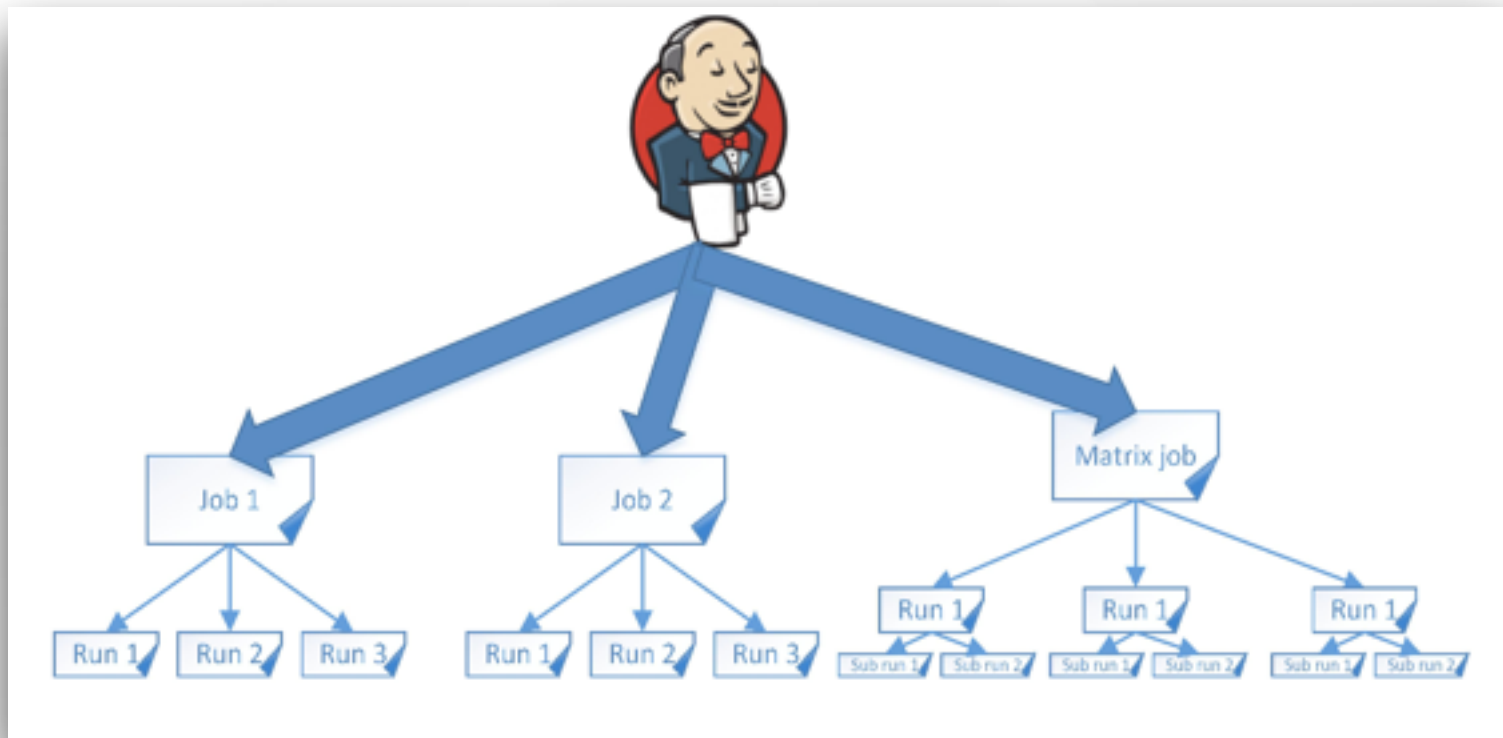
#	Status
1	Idle
2	Idle

 [Help us localize this page](#)

Welcome to Jenkins! Please [create new jobs](#) to get started.



● Structure des tâches





Jenkins - Paramétrage

JDK

JDK installations

⌵

JDK

Name

JAVA_HOME

☐ Install automatically

List of JDK installations on this system

Ant

Ant installations

⌵

Ant

Name

☒ Install automatically

⌵

Install from Apache

Version

List of Ant installations on this system

Maven

Maven installations

⌵

Maven

Name

☒ Install automatically

⌵

Install from Apache

Version



Jenkins - administration

Administrer Jenkins (Jenkins) - Google Chrome

192.168.0.14:8080/manage

Jenkins

ACTIVER LE ENTRAÎNEMENT AUTOMATIQUE

Administrer Jenkins

- [Configurer le système](#)
Configurer les paramètres généraux et les chemins de fichiers.
- [Rechercher 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.
- [Gérer 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.
- [Log système](#)
Le log système capture le texte Java, net.J, logging relative à Jenkins.
- [Diagnostics d'utilisation](#)
Vérifiez l'utilisation des ressources et décidez si vous avez besoin d'ordinateurs supplémentaires pour vos builds.
- [Ligne de commande d'API Jenkins](#)
Accéder au 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.
- [À propos de Jenkins](#)
Affiche les informations de version et de licence.
- [Primer le système](#)
Créer d'ordinateur de nouveaux builds, afin que le système puisse se lancer.

Page générée: 3 août 2012 19:27:25 [Jenkins ver. 1.404](#)



Jenkins - Paramétrage

● Paramétrage de proxy

The screenshot shows the Jenkins Update Center interface in a Google Chrome browser. The page title is "Update Center [Jenkins] - Google Chrome". The address bar shows the URL "192.398.0.34:9000/pluginManager/advanced". The Jenkins logo is visible in the top left corner. The main content area is titled "Configuration du proxy HTTP" and contains four input fields: "Serveur", "Port", "Nom d'utilisateur", and "Mot de passe". Below these fields is a "Soumettre" button. The next section is "Soumettre un plugin", which includes a text input field for the plugin path, a "Fichiers" dropdown menu, and another "Soumettre" button. The final section is "Update Site", which has a text input field for the update site URL and a "Soumettre" button. At the bottom right, there is a "Dernière mise à jour: 2 y a 2 h 4 min" label and a "Vérifier maintenant" button. The footer of the page indicates "Page générée: 3 août 2013 22:35:02" and "Jenkins ver. 1.434".

Update Center [Jenkins] - Google Chrome

192.398.0.34:9000/pluginManager/advanced

Jenkins

Jenkins - Gestion des plugins

Retour au tableau de bord

Administrer Jenkins

Mises à jour Disponibles Installées **Avancé**

Configuration du proxy HTTP

Serveur

Port

Nom d'utilisateur

Mot de passe

Soumettre un plugin

Indiquez ici le chemin vers un plugin à ajouter à ce serveur Jenkins

Fichiers: Aucun fichier choisi

Update Site

URL:

Dernière mise à jour: 2 y a 2 h 4 min

Page générée: 3 août 2013 22:35:02 [Jenkins ver. 1.434](#)



Jenkins - Sécurité



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								
	Administer	Read	Run	Scripts	Upload	Plugins	Configure	Update	Center	Configure	Delete	Create	Disconnect	Connect	Create	Delete	Configure	Read	Discover	Build	Workspace	Cancel	Delete	Update	Create	Delete	Configure	Read	Tag
Anonymous	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 user	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

User/group to add:

☐ Project-based Matrix Authorization Strategy

☐ Prevent Cross Site Request Forgery exploits



● Page d'authentification

The screenshot shows the Jenkins login interface. It features a 'User:' label followed by a text input field, a 'Password:' label followed by a password input field, and a checkbox labeled 'Remember me on this computer'. Below these is a 'log in' button. At the bottom, there is a link that says 'Create an account if you are not a member yet.' To the left of the login form, there is a vertical sidebar with several gray rectangular buttons.

● Lien de création d'un nouveau compte



Jenkins - gestion des rôles

● Jenkins offre plusieurs rôles pour la gestion des habilitations des utilisateurs

- Delete
- Configure
- Read
- Discover

Project roles									
Role	Pattern	Job							
		Delete	Configure	Read	Discover	Build	Workspace	Cancel	ViewStatus
Anonymous	[LCIM]?-.*	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



Jenkins - Plugins

- *L'outil est extensible à l'aide d'une offre riche de plugins*

The screenshot shows the Jenkins Update Center interface. The browser address bar indicates the URL `192.168.0.14:8080/pluginManager/`. The page title is "Jenkins" and the breadcrumb is "Jenkins > Gestion des plugins". On the left sidebar, there are links for "Retour au tableau de bord" and "Administrer Jenkins". The main content area has tabs for "Mises à jour", "Disponibles", "Installés", and "Avancé". The "Mises à jour" tab is active, showing a table of available updates. The table has columns for "Nom", "Version", and "Installée". Three plugins are listed: "CVS Plugin", "Subversion Plugin", and "SSH Slaves plugin". Each plugin entry includes a brief description. Below the table is an "Installer" button and a note: "Cette page liste les mises à jour des plugins en cours d'utilisation."

	Nom	Version	Installée
	CVS Plugin This bundled plugin integrates Jenkins with CVS version control system.	1.5	1.2
	Subversion Plugin This plugin adds the Subversion support (via SVNKit) to Jenkins.	1.29	1.28
	SSH Slaves plugin This plugin allows you to manage slaves running on Unix machines over SSH.	0.18	0.17

[Installer](#)


Cette page liste les mises à jour des plugins en cours d'utilisation.





Jenkins - Définition d'une tâche


Jenkins


Jenkins > All >

 [New Job](#)

 [People](#)

 [Build History](#)

 [Manage Jenkins](#)

 [My Views](#)

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Job name


☒ **Build a free-style software project**
This is the central feature of Jenkins. Jenkins

☐ **Build a maven2/3 project**
Build a maven 2/3 project. Jenkins takes adv

☐ **Build multi-configuration project**
Suitable for projects that need a large number

☐ **Monitor an external job**
This type of job allows you to record the exec

☐ **Copy existing Job**
Copy from

 [Help us localize this page](#)



Jenkins - rapport de tests

● Configuration des rapports de tests dans Jenkins

Build

Root POM

Goals and options

[Advanced...](#)

Build Settings

☐ E-mail Notification

Post-build Actions

☐ Archive the artifacts

☐ Aggregate downstream test results

☐ Build other projects

☐ Deploy artifacts to Maven repository

● Jenkins traite n'importe quel rapport de tests compatible xUnit



Jenkins - rapport de tests

● Publier les résultats de test xUnit

☒ Publish testing tools result report ?

Boost Test Library Pattern

Fail the build if test results were not updated this run ☒

Delete temporary JUnit files ☒

Delete

Add ▾

- Custom Tool
- NUnit
- MSTest
- Boost Test Library
- UnitTest
- Free Pascal Unit
- PHPUnit

container

ed build on other projects

?

?

?

Mise en place des métriques





Génération des rapports

- *La tâche site de maven permet la génération de site web*

The screenshot shows a Maven project website for 'consumerBanking'. The page has a sidebar with 'Project Documentation' links and a main content area with sections for Project Summary, Project Information, Project Organization, and Build Information.

consumerBanking
Last Published: 2012-07-11

Project Documentation

- Project Information
- About
- Continuous
- Integration
- Dependencies
- Issue Tracking
- Mailing Lists
- Plugin Management
- Project License
- Project Summary**
- Project Team
- Source Repository

Project Summary

Project Information

Field	Value
Name	consumerBanking
Description	-
Homepage	http://maven.apache.org

Project Organization

This project does not belong to an organization.

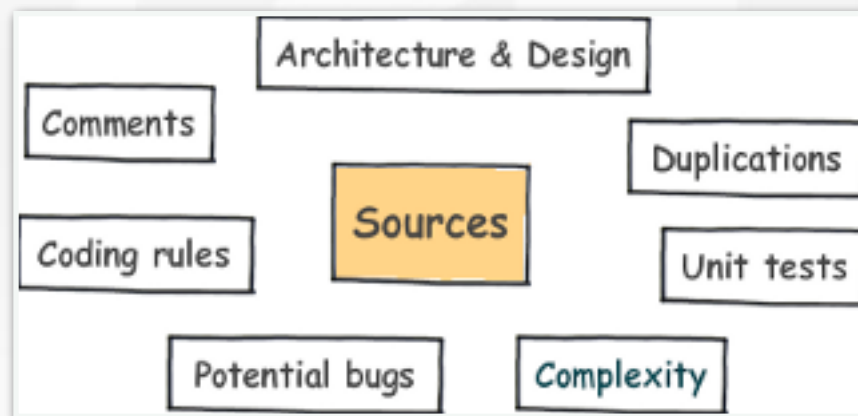
Build Information

Field	Value
GroupId	com.companyname.bank
ArtifactId	consumerBanking
Version	1.0-SNAPSHOT
Type	jar

© 2012



- *Maven offre plusieurs plugins qui permettent l'amélioration de la qualité de code*





Normes et règles

- *Les normes et standards de développement permettent de s'accorder sur un ensemble de pratiques*
- *Elles permettent d'assurer la bonne compréhension du code par l'ensemble des développeurs qu'ils soient présents sur le projet dès le démarrage ou dans le futur*



- *Il existe différents type de règles :*
 - *conventions de nommage*
 - *Simplicité du code*
 - *Documentation*
 - *Code mort*
 - *Manque d'abstraction*
 - *Ressources non libérées*
 - *Traitement des exceptions et gestion des traces*



- *Un grand nombre de règles de développement existent et peuvent être vérifiées automatiquement*
 - *La présence de commentaire*
 - *les règles de nommage...*
- *Exemples d'outils :*
 - *checkstyle,*
 - *PMD,*
 - *Findbugs...*



Mesures de qualité - checkstyle

- *Checkstyle est un outil d'analyse statique pour Java.*
- *A l'origine conçu pour faire respecter un ensemble de normes de codage hautement configurable, Checkstyle permet aussi maintenant de vérifier les mauvaises pratiques de codage, ainsi que le code trop complexe ou dupliqué.*
- *Checkstyle est un outil polyvalent et flexible qui devrait avoir sa place dans n'importe quelle stratégie d'analyse de code basé sur Java*



Mesures de qualité - checkstyle

● *Voici la déclaration du plugin maven checkstyle*

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <version>2.4</version>
      <configuration>
        <configLocation>
          src/main/config/company-checks.xml
        </configLocation>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```



Mesures de qualité - PMD

- *PMD est un autre outil populaire d'analyse statique.*
- *Il se focalise sur les problèmes potentiels de codage comme :*
 - *le code non utilisé ou sous-optimisé,*
 - *la taille et la complexité du code, et les bonnes pratiques de codage.*
- *Certaines règles typiques intègrent :*
 - *« Empty If Statement », « Broken Null Check »,*
 - *« Avoid Deeply Nested If Statements,*
 - *« Switch Statements Should Have Default », et « Logger Is Not Static Final »*



Mesures de qualité - checkstyle

● *Voici la déclaration du plugin maven checkstyle*

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <version>2.4</version>
      <configuration>
        <configLocation>
          src/main/config/company-checks.xml
        </configLocation>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```



Mesures de qualité - FindBugs

- *FindBugs est un puissant outil d'analyse de code qui vérifie le byte code des applications afin de trouver :*
 - *des anomalies potentielles,*
 - *des problèmes de performances,*
 - *ou des mauvaises habitudes de codage*



Mesures de qualité - FindBugs

● *Voici la déclaration du plugin maven FindBugs*

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>findbugs-maven-plugin</artifactId>
      <version>2.3.1</version>
      <configuration>
        <effort>Max</effort>
        <xmlOutput>true</xmlOutput>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

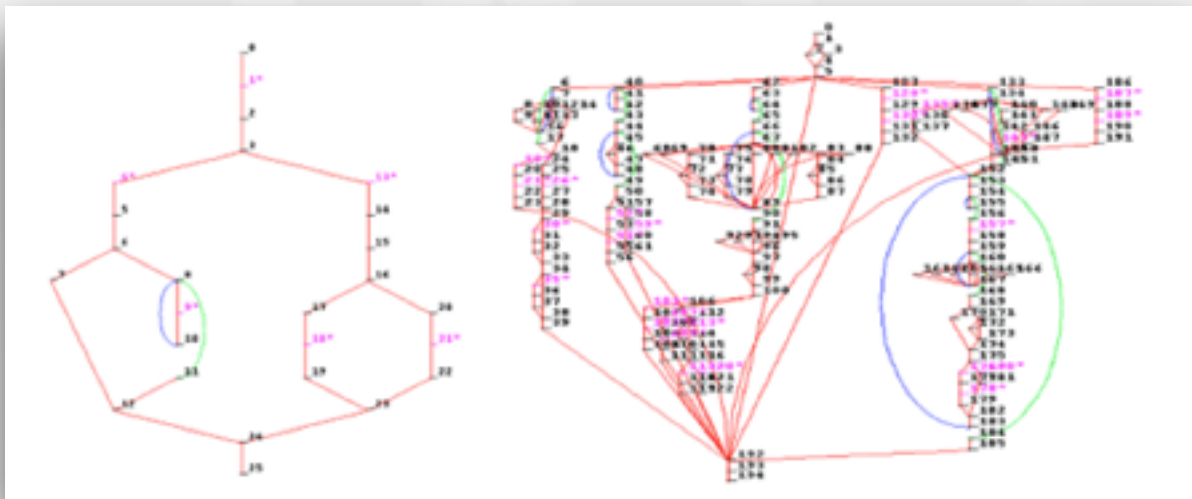


- *Métriques concernant la simplicité du code*
 - *le nombre de lignes de code d'une méthode ou d'une classe*
 - *La complexité cyclomatique mesure le nombre de chemins*



Mesures de qualité

- *La complexité cyclomatique développée par Thomas McCabe,*
 - *Au delà de 7 chemins, il devient difficile de comprendre, maintenir et tester le code*





Mesures de qualité

Les outils de consolidation permettent d'avoir une vision d'ensemble cohérente de la qualité des applications (exemple sonar)





Amélioration continue

- *Chaque bug doit être perçu comme le point de départ de l'amélioration du processus de développement dans son ensemble*
- *A l'identification d'un bug, il faut s'interroger sur les raisons de l'anomalie ?*
 - *Spécifications erronées, incomplètes ?*
 - *Couverture des tests trop faible ?*
 - *Code trop complexe, conception perfectible ?*
 - *Paramétrage incomplet en production ?*
- *Un test doit être rajouté au référentiel pour pouvoir le détecter dans les prochaines itérations*

Synthèse





● *Vision développeur*

- *Réduction du temps de correction des bogues*
- *Intégration des modifications régulièrement*
- *La portion de code à débbugger est faible lors de la détection des anomalies*



● *Vision développeur*

● *Amélioration du travail collaboratif*

- *La fin du 'Ca marche sur mon poste pourtant !'*

- *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*

● *L'IC encourage des bonnes habitudes de test*



Intégration continue

● *Vision de chef de projet*

- *L'intégration continue 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*



Intégration continue

● *Vision de chef de projet*

● *Pilotage / Suivi*

- *L'intégration continue 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*



● *Pour aller plus loin*

- <http://martinfowler.com/articles/continuousIntegration.html>
- <https://hudson.dev.java.net/>
- <http://sonar.codehaus.org/>
- <http://maven.apache.org/>
- <http://www.agilenantes.org/wp-content/uploads/2010/02/Presentation-IC-agileNantes1.pdfs>
- http://jerome.lenaou.free.fr/wp-content/uploads/Gestion_de_projet.pdf
- <http://pagesperso.lina.univ-nantes.fr/~prie-y/ens/11-12/SIMA/pdf/CM-processus-agile.pdf>