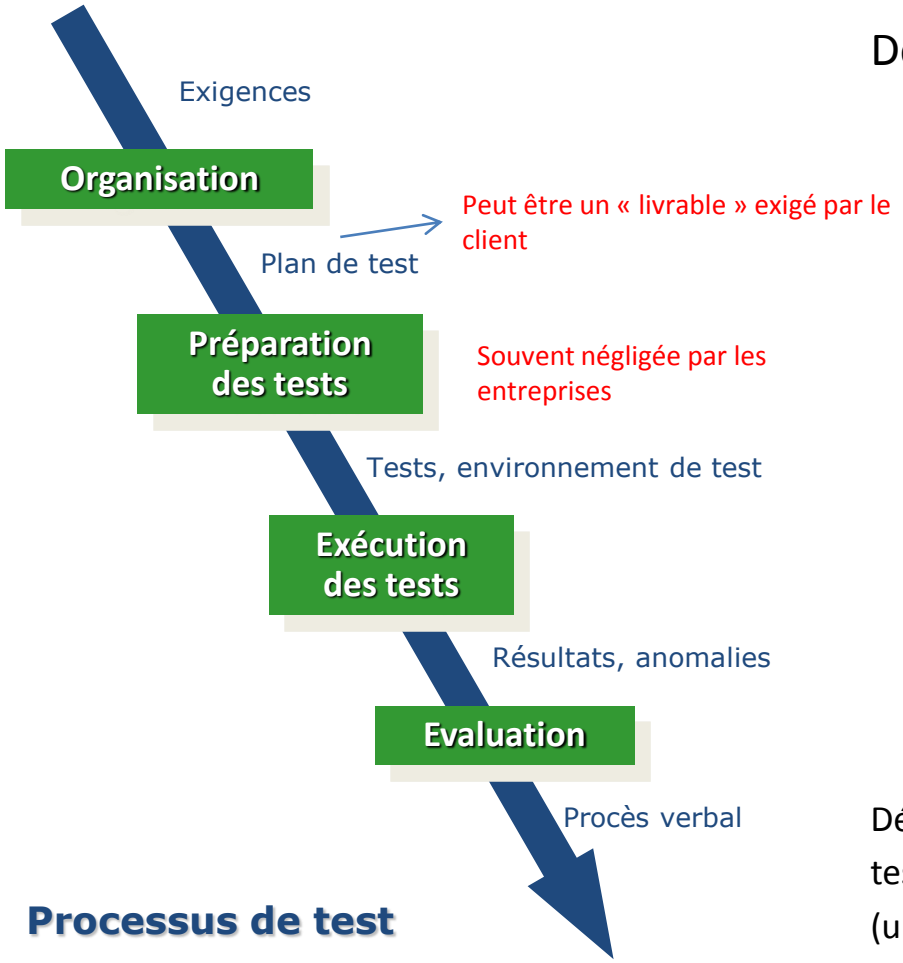
# Processus de test
# Phases et outils

Révision : Septembre 2016

# La démarche classique en validation



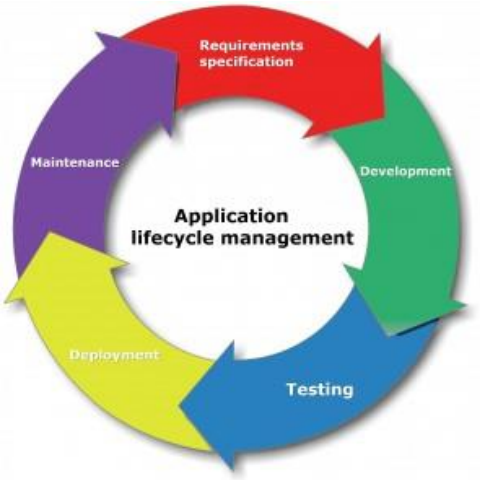

Démarche générale en 4 étapes :

- Phase initiale d'organisation conduisant à produire le plan de test de la phase dès que le contexte est connu
- Phase de préparation des tests débutant dès que les documents de définition sont disponibles
- Phase d'exécution débutant dès la mise à disposition de l'objet sous test
- Phase d'évaluation finale avant livraison

Démarche applicable à chaque phase ou processus de test en ajustant en fonction de la nature des tests (unitaires, intégration, validation …)

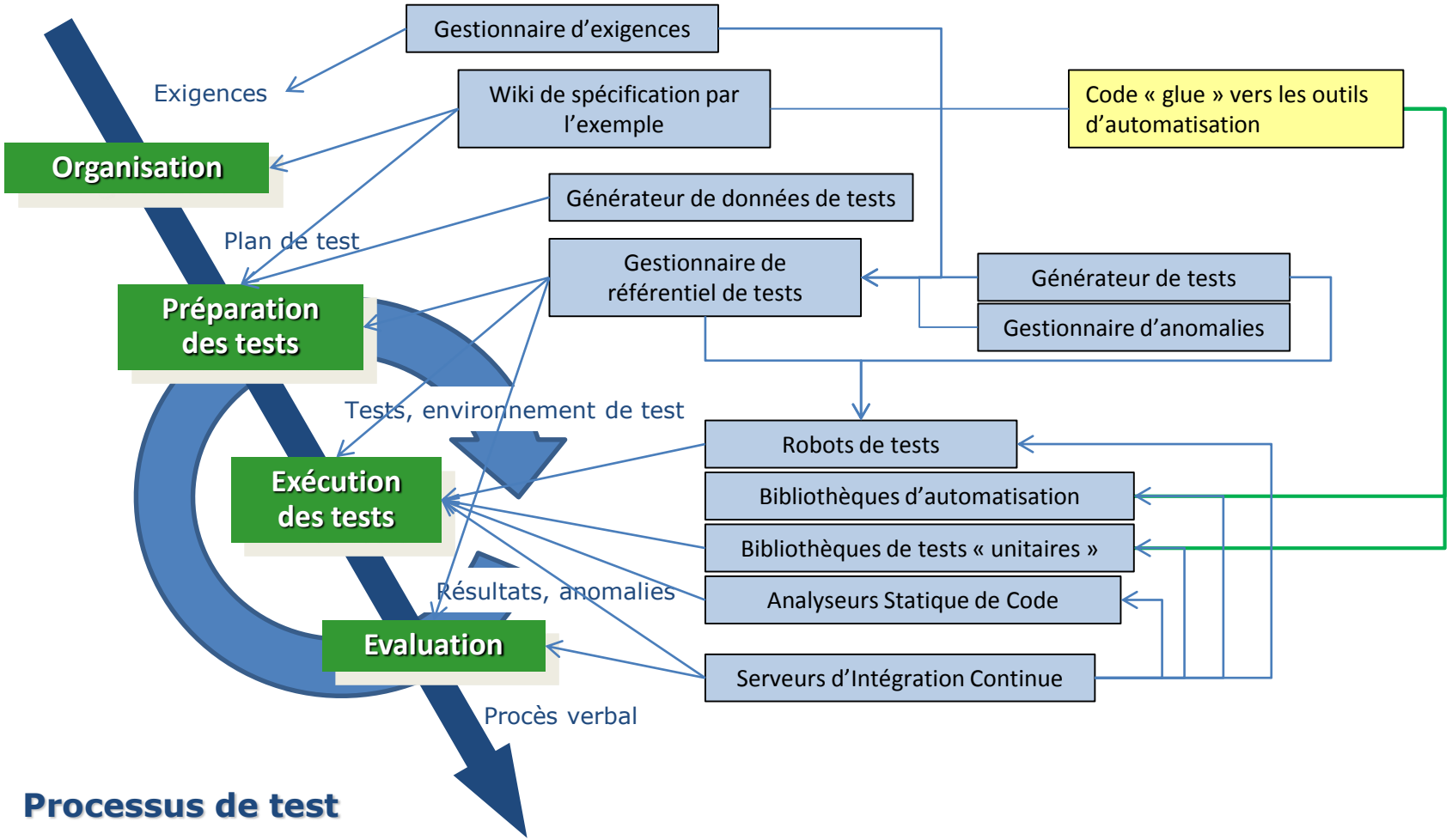
# Quoi automatiser ou outiller ?

- **3 grands axes :**
  - **Automatiser l'exécution complète des tests**
    - Préparation des données
    - Exécution des tests unitaires, intégrations, validations, performance …
    - Génération des automatique des résultats
  - **Outiller/Automatiser la gestion des tests**
    - Traçabilité Exigences <-> Tests
    - Gestion de la bases de tests
    - Gestion des campagnes de tests
    - Gestion des résultats et PREUVES de tests
    - Gestion du projet de tests
    - Gestion des versions /configurations TESTWARE
  - **Outiller/Automatiser les processus connexes aux test**
    - Gestion des exigences
    - Gestion des versions/configurations SOFTWARE
    - Gestion des anomalies

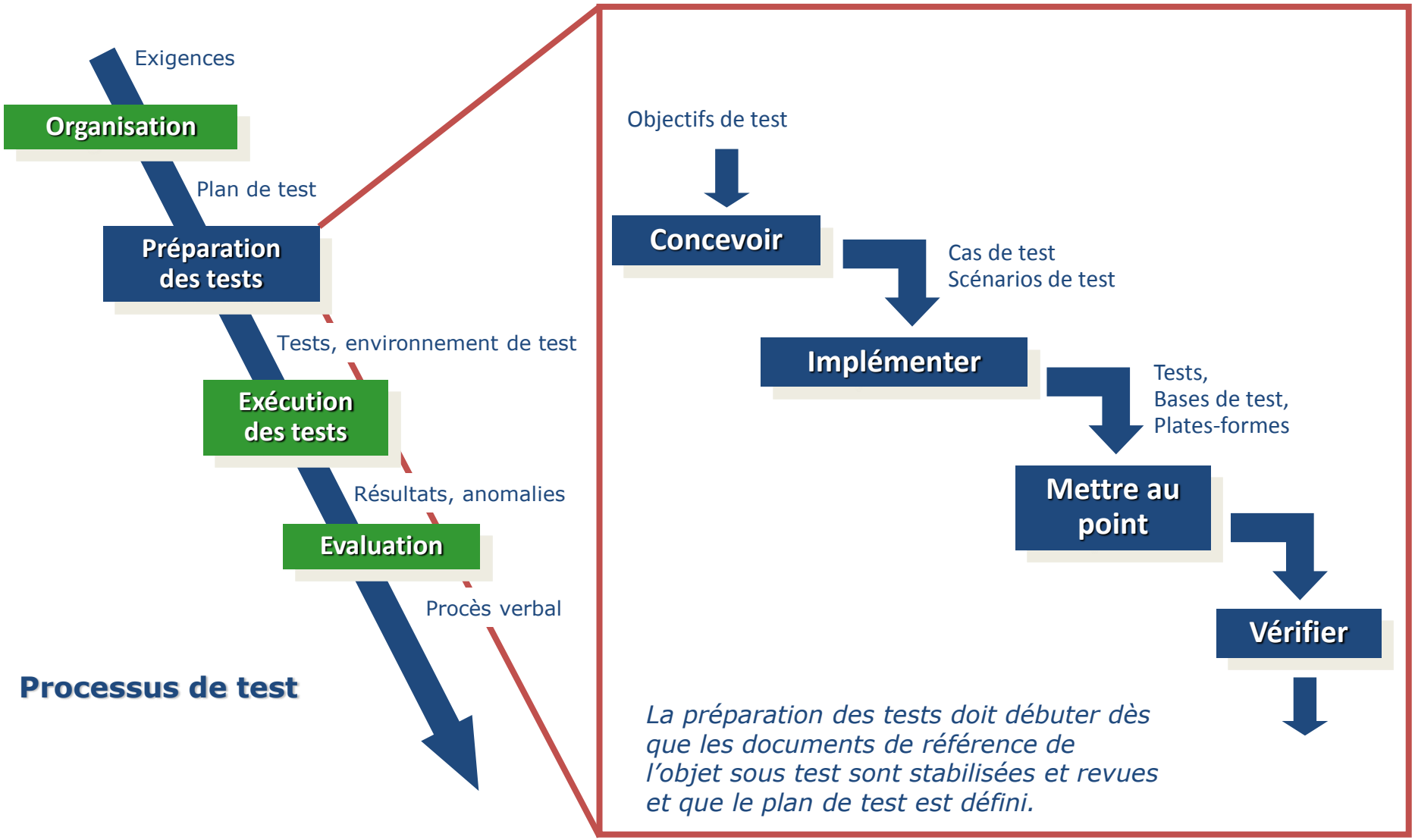



Orchestrer tous les outils pour faciliter la gestion du cycle de vie des applications.

# Les différents types d'outils



# Préparation des tests



# Les gestionnaires de référentiel de tests

## Fonctionnalités

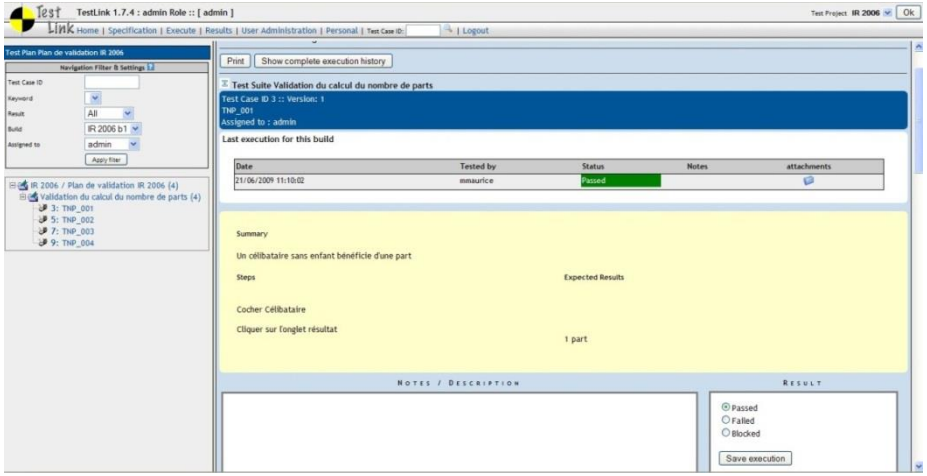
- Gestion des utilisateurs
- Gestion des exigences
- Gestion de l'architecture des tests
- Gestion des tests (tests manuels, scripts de test, bases de test)
- Gestion de la traçabilité
- Version des tests
- Gestion des versions de build
- Préparation des campagnes de test
- Gestion des résultats des campagnes de test
- Couplage avec l'outil de gestion des exigences, des spécifications
- Couplage avec le robot de test
- Couplage avec l'outil de gestion des anomalies
- Reporting

- QA DIRECTOR (COMPUWARE)
- **TESTLINK (OPEN SOURCE)**
- **SQUASH TM (OPEN SOURCE)**
- **ProjeQtOR**
- **ALM (HP) ( Quality Center )**
- RATIONAL QUALITY MANAGER(IBM)
- SILK CENTRAL TEST MANAGER (COMPUWARE)

## Qualités de l'outil

- Collaboratif
- Adaptation méthodologique
- Intégration dans le cycle de vie
- Eventail de couplages
- Multi-projets
- Multi-langues
- IHM
- Indicateurs
- Visualisation graphique

*Vue Testlink*




*Syn. Gestionnaires de testware*

# Création d'un test automatique …

- Depuis un test manuel existant
  - Pourquoi automatiser ce test ?
  - Identifier le test : identifiant de test
  - Identifier les exigences couvertes : identifiants d'exigences
  - **Préparer les données et la base de données de tests**
  - Avec un robot « tout en un »
    - **Enregistrer le test manuel en mode « capture »**
    - **Remanier le code ( initialisation, tests, points de contrôles, finalisation)**
    - **« Variabiliser » avec les assistants de connexion aux données du robot**
  - Avec une bibliothèque des tests unitaires et/ou une bibliothèque de pilotage des navigateurs web
    - **Traduire les pas du test manuel en fonctions informatiques de la bibliothèques**
    - **Utiliser une bibliothèque de tests unitaires adaptées aux langages utilisés pour avoir la structure standard**
      **( initialisation, tests, points de contrôles, finalisation)**
    - **« Variabiliser » si besoin avec des « injecteurs de données »**

# Création d'un test automatique …

- **« From scratch » avec un robot (profil validation finale)**
  - Un expert métier peut enregistrer en mode « capture » **un script pertinent sur des données pertinentes**
  - Il existe des outils qui masquent le robot pour que l'expert métier se retrouve dans un environnement de travail familier.
  - Un automaticien de test remanie ensuite le script obtenu
  - On met en place ensuite la chaîne de traçabilité amont : Script -> ID Test -> Ids Exigences
- **« From scratch » avec une bibliothèque de tests (profil développement)**
  - Fait partie du processus de développement moderne
  - Privilégier le mode « Test First » voire le «Test Driven Development »
  - Peut être néanmoins confié à un automaticien des tests indépendant, à postériori, si il s'agit d'un test de validation.

# Quels tests de validation automatiser ?

- Les tests finaux de validation les plus **longs et fastidieux** à faire en manuel
- Les **tests de non régression**
- Etablir plusieurs critères de sélection notés sur 100 :
  - CF : **Criticité des fonctions** testées
  - SF : **Stabilité des fonctions** testées par le test (Maintenance)
  - CEM : **Complexité d'exécution** manuelle du test
  - PA : **Possibilité d'automatisation** du test (non complexité)
  - …
- En pondérant les critères, établir une formule de classement des tests pour sélectionner les plus pertinents en fonction de vos objectifs.
  - Par exemple : (4 * CF + 2 * SF + 6 * CEM + 6 * PA) / 18

| Domaine fonctionnel | N° fiche scénario | Libelle du scénario | Criticité de la(les) fonction(s) testées coef. : 4 | Stabilité de la fonction coef. : 2 | Complexité d'exécution manuelle coef. : 6 | Possibilité d'automatisation coef. : 6 | NOTE TOTALE |
|---|---|---|---|---|---|---|---|
| | | | Note sur 100 | | | | |
| Requêtes BO | NR_REQUETE_02 | Création et mise à jour des personnes | 100,00 | 100,00 | 40,00 | 66,67 | 76,30 |
| Requêtes BO | NR_REQUETE_03 | Création et mise à jour des titulaires | 100,00 | 100,00 | 40,00 | 66,67 | 76,30 |
| Requêtes BO | NR_REQUETE_05 | Les rejets : khi & kti | 100,00 | 100,00 | 40,00 | 66,67 | 76,30 |
| Modification (PM) | NR_MODIF_PM_05 | Modification d'une fiche Personne (PM) : Titulaire(s) Bancaire(s) Simple(s) | 83,33 | 33,33 | 73,33 | 66,67 | 75,06 |
| Modification (PP) | NR_MODIF_PP_05 | Modification d'une fiche Personne (PP) : Titulaire(s) Bancaire(s) Simple(s) | 83,33 | 33,33 | 73,33 | 66,67 | 75,06 |
| Modification (PP) | NR_MODIF_PP_06 | Modification d'une fiche Personne (PP) : Titulaire(s) Bancaire(s) Groupe(s) | 83,33 | 33,33 | 73,33 | 66,67 | 75,06 |
| Fusion | NR_FUSION_01 | Fusion de 2 personnes | 91,67 | 33,33 | 60,00 | 66,67 | 73,09 |
| Recherche (PM) | NR_RECH_PM_01 | Recherche d'une personne morale | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Recherche (PP) | NR_RECH_PP_01 | Recherche d'une personne physique | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PM) | NR_VISU_PM_01 | Visualisation d'une fiche Personne (PM) : Bandeau | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PM) | NR_VISU_PM_02 | Visualisation d'une fiche Personne (PM) : Signalétique | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PM) | NR_VISU_PM_06 | Visualisation d'une fiche Personne (PM) : Titulaire(s) Bancaire(s) | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PP) | NR_VISU_PP_01 | Visualisation d'une fiche Personne (PP) : Bandeau | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PP) | NR_VISU_PP_02 | Visualisation d'une fiche Personne (PP) : Signalétique | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PP) | NR_VISU_PP_06 | Visualisation d'une fiche Personne (PP) : Titulaire(s) Bancaire(s) Simple(s) et Groupe(s) | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Recherche | NR_RECH_00 | Recherche d'une personne : test des critères | 66,67 | 100,00 | 33,33 | 77,78 | 67,90 |
| Visualisation (PM) | NR_VISU_PM_05 | Visualisation d'une fiche Personne (PM) : Déontologie et suivi | 58,33 | 100,00 | 33,33 | 77,78 | 65,43 |
| Visualisation (PP) | NR_VISU_PP_05 | Visualisation d'une fiche Personne (PP) : Déontologie et suivi | 58,33 | 100,00 | 33,33 | 77,78 | 65,43 |
| Visualisation (PM) | NR_VISU_PM_08 | Visualisation d'une fiche Personne (PM) : Individus | 66,67 | 66,67 | 33,33 | 77,78 | 64,20 |
| Visualisation (PP) | NR_VISU_PP_08 | Visualisation d'une fiche Personne (PP) : Individus | 66,67 | 66,67 | 33,33 | 77,78 | 64,20 |
| Requêtes BO | NR_REQUETE_01 | Contrôle de cohérence | 58,33 | 100,00 | 40,00 | 66,67 | 63,95 |
| Requêtes BO | NR_REQUETE_04 | Flux d'individus | 58,33 | 100,00 | 40,00 | 66,67 | 63,95 |
| Administration | NR_ADMIN_01 | Vérification des privilèges utilisateurs | 41,67 | 66,67 | 60,00 | 72,22 | 63,83 |
| Visualisation (PM) | NR_VISU_PM_04 | Visualisation d'une fiche Personne (PM) : MIF & Profil de risque | 50,00 | 100,00 | 33,33 | 77,78 | 62,96 |

# Préparation des données de tests

- **Préparation de la base de données de tests**
  - Copie de la base de production
  - Mécanique d'extraction
    - D'une partie des données
    - D'une partie des tables

- **Les données portant sur l'alimentation des tests et les oracles de tests**
  - Techniques de conception des tests ( limites, classes d'équivalences, valeurs invalides )
  - Stockage « embarqué » dans le code des tests, dans des fichiers externes, voir en base de données

- **Les problèmes courants**
  - Anonymisation des données (CNIL) ( parfois trop compliqué )
  - Rajeunissement et vieillissement des données ( Important dès la conception du système )
  - La réservation des données pour éviter que deux tests touchent aux mêmes données en parallèle.

# Outils pour la préparation des données de tests

- Le générateurs de données
  - DTM Data Generator
  - Mockaroo en ligne
- Les «batchs» de commandes (Extraction SQL …)
- **Les robots de tests pour alimenter la base de données de tests par l'IHM**
- Les outils spécialisés dans l'extraction et la gestion des données
  - Jailer (Open Source)
  - IBM Optim Data Management
- Les générateurs de tests complets
  - Model Based Testing avec Certify It et Matelo
  - Algorithmes spécialisés (valeurs limites, valeurs interdites)

# Générateurs de données

## Fonctionnalités

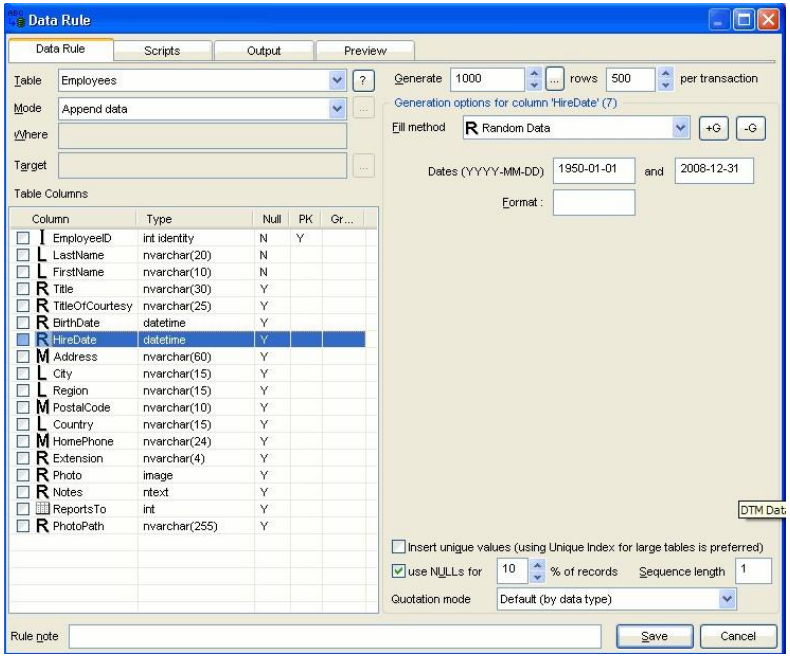
- Extraction de données
- Formatage des données de test selon expressions régulières, règles d'intégrité référentielle
- Génération aléatoire de vecteurs de test
- Anonymisation des données
- Gestion des environnements

- **DATA GENERATOR (OPEN SOURCE)**
- **DTM DATA GENERATOR**
- **OPTIM TEST DATA MANAGEMENT (IBM)**
- **DATAGEN (E-NAXOS)**
- **SPAWNER (OPEN SOURCE)**
- **MOCKAROO (Online/Free en version de base)**

## Qualités de l'outil

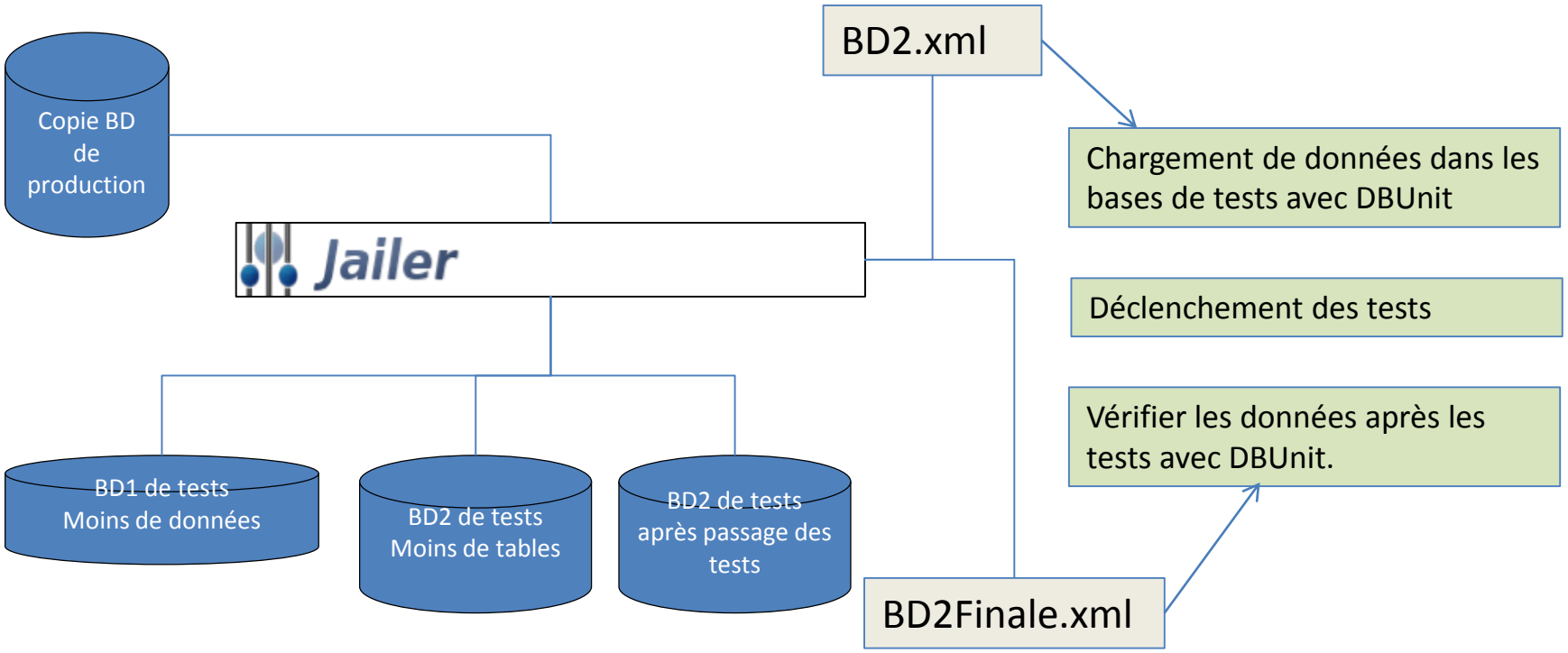
- Multiplicité des formats (types de données, CSV, HTML, XML …)
- Interfaces ODBC
- IHM

*Vue DTM Data Generator*



# Extraire des données avec Jailer

- Open source et supports de nombreux SGBD
- Extraction de sous ensembles de données et de structures
- Sauvegarde d'états des données au formats DBUnit.
- Comparaison et rechargement des états des données / à la base de test



# IBM Optim Data Managment

Extraction
Anonymisation
Réservation
Vieillissement

## The bottom line: Managing test data nets real business value

Production data doesn't stand still—and neither should test data. Organizations need test data management solutions that are designed to accommodate changing test requirements.

**Support a complete test data management strategy with IBM InfoSphere Optim solutions**

The IBM InfoSphere® Optim™ Test Data Management solution offers comprehensive test data management capabilities for creating rightsized, fictionalized test databases that accurately reflect end-to-end business processes. InfoSphere Optim software scales to meet development and testing requirements across multiple applications, databases, operating systems and hardware platforms. It also helps facilitate modern software delivery models—including agile development—by making test data continuously accessible to testers and developers so they can quickly meet test requirements (see Figure 1).

The InfoSphere Optim Test Data Management solution helps improve application quality and delivery efficiency by:

- Reducing costs by intelligently creating and subsetting realistic test data from complete business objects
- Reducing risk by masking sensitive information
- Speeding delivery of test data through refresh capabilities

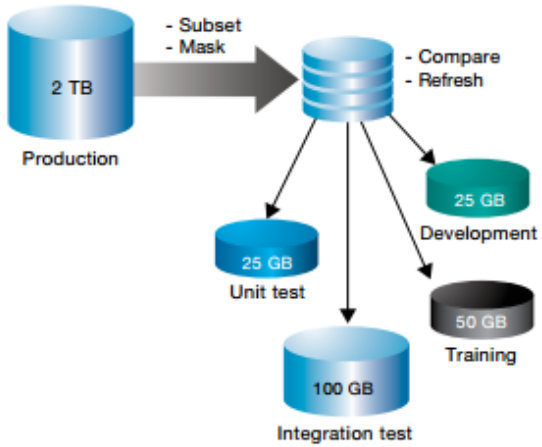
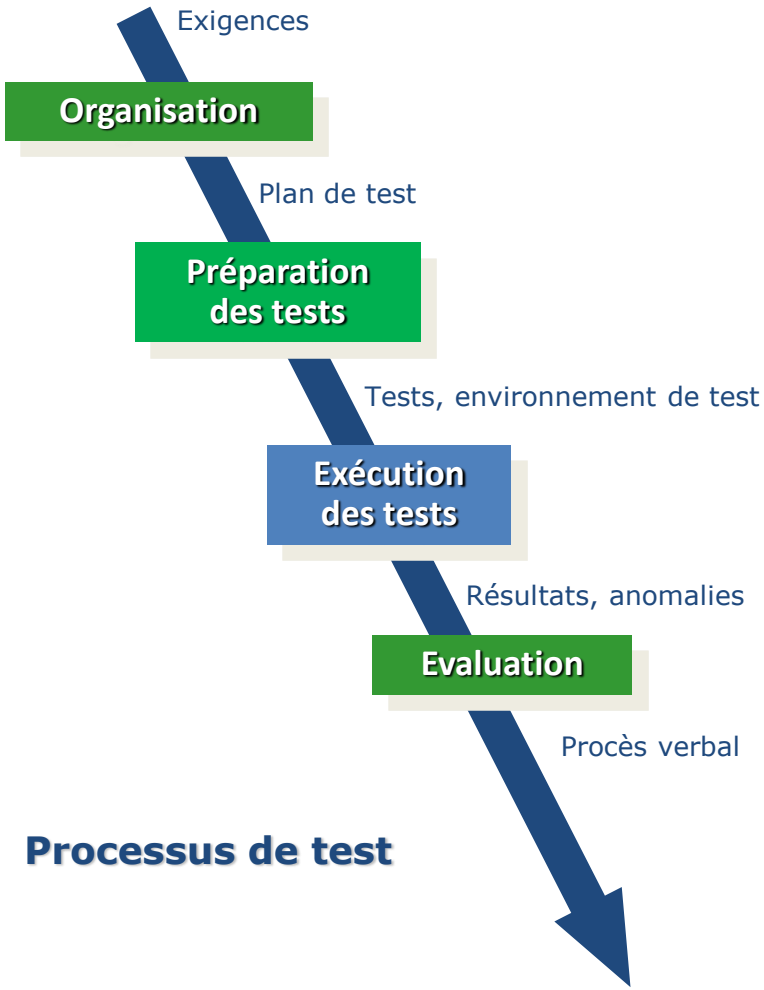



Figure 1: InfoSphere Optim software enables users to create referentially intact, rightsized and secure test databases.

9

Un exemple d'utilisation chez Bouygues Telecom : anonymisation-bouygues-telecom.pdf

# Exécution des tests




- Exécution des tests
- Récupération des résultats
- Gestion des anomalies

# Exécution des tests


- **Les outils autonomes**
  - Robots de tests
  - xUnit
  - Bibliothèques spécialisées dans l'automatisation des tests
    - Robot Framework (Google)
    - Selenium Web Driver
  - Frameworks propriétaires
  - Batchs, lignes de commandes

- **Les outils ALM d'intégration et de collaboration**
  - Serveur d'intégration continue
  - Gestionnaire de référentiel des tests
  - Mircosoft Team Test Foundation
  - …

# Piloter par les constructeurs

- **Les constructeurs ant, maven, gradle, grunt, MS Build, …**
  - **Idée reçue** : ils ne sont pas mono langage !
    - Nombreux cas de projets PHP gérés avec **ant** écrit en java
  - Beaucoup de balises ou d'actions mis à disposition
    - Balises de déclenchement des outils de tests
    - Balises de récupération et de transformation des résultats
  - Facilitent grandement la manipulation des fichiers, et répertoires ainsi que les connections avec les gestionnaires de versions des sources.
  - Portables sur différents OS




# Ant avec Java et PHP

```xml
<target name="test" depends="compiletest" description="run the tests " >
  <junit printsummary="yes" fork="yes" haltonfailure="yes">
    <formatter type="plain"/>
    <batchtest fork="true">
      <fileset dir="${test}">
        <include name="**/*Test*.java"/>
      </fileset>
    </batchtest>
    <classpath refid="${project.classpath}"
  </junit>
```

Lancement intégré de JUnit

```xml
<target name="phpcpd-ci"
        depends="prepare"
        description="Find duplicate code using PHPCPD and log
 <exec executable="${toolsdir}phpcpd">
  <arg value="--log-pmd" />
  <arg path="${basedir}/build/logs/pmd-cpd.xml" />
  <arg path="${basedir}/src" />
 </exec>
</target>

<target name="phpunit"
        depends="prepare"
        description="Run unit tests with PHPUnit">
 <exec executable="${toolsdir}phpunit" failonerror="true">
  <arg value="--configuration"/>
  <arg path="${basedir}/build/phpunit.xml"/>
 </exec>
</target>
```

Lancement de l'exécutable phpunit

# Piloter par les robots de tests

- Lancement depuis le robot
  - En phase de mise au point ou vérification des tests
  - Sur des suites instables

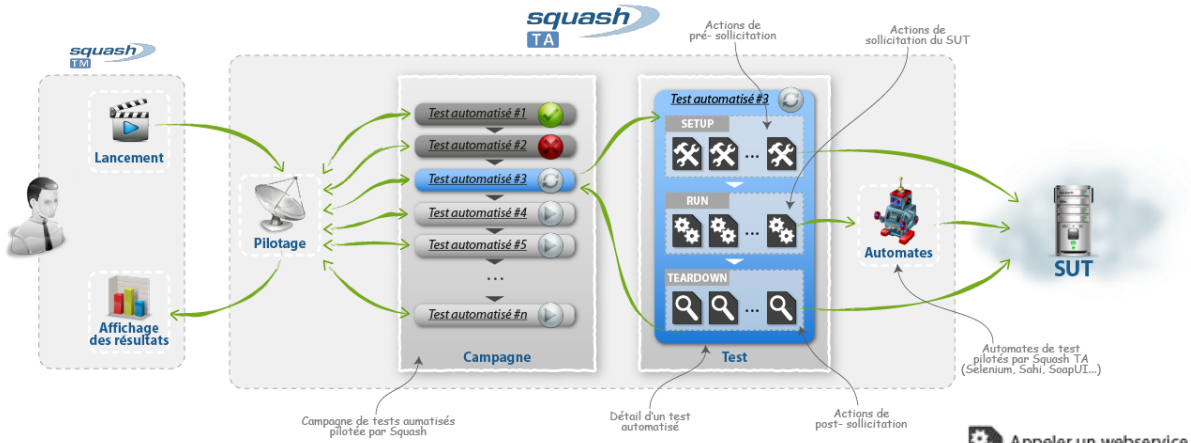

- TEST COMPLETE (SMART BEAR)
- SELENIUM (OPEN SOURCE)
- QUICK TEST PRO (HP)
- RATIONAL ROBOT (IBM)
- SIKULI(OPEN SOURCE)
- JMETER (OPEN SOURCE)
- LOADUI(OPEN SOURCE)
- RANOREX (RANOREX Software)
- TESTPARTNER (COMPUWARE)

- Lancement du robot en mode silencieux
  - Mode ligne commande ( option –silent )
  - Depuis un batch ou un constructeur

- Plateforme d'exécution
  - Certains robots produisent des exécutables
  - Des plateformes d'exécution locales ou distantes peuvent être utilisées (Ranorex)


# Pilotage des robots avec Squash TA

- Squash TA est basé sur Jenkins …




Permet de « profiter » de toute l'infrastructure de Jenkins et de son moteur d'orchestration

Squash TA peut être lui-même piloté par Squash Test Management (Gestionnaire de référentiel de tests) et ainsi consolider les résultats plus facilement.


Remerciement à SquashTest.org et à la société Henix

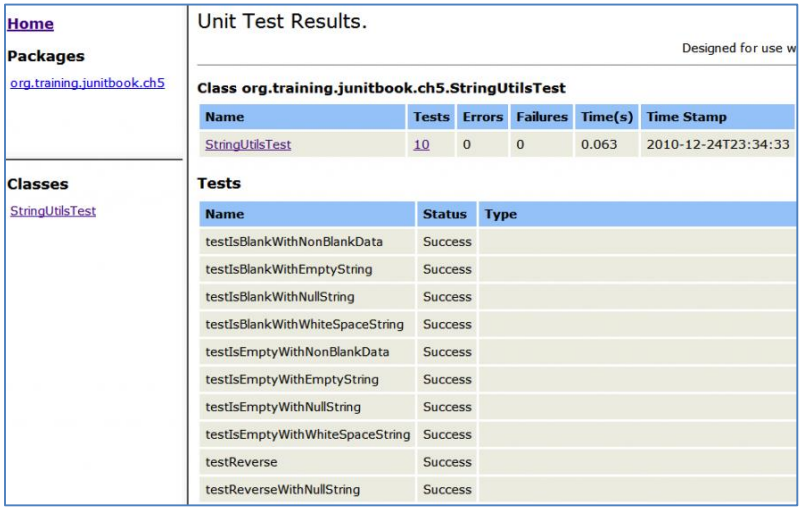
# Génération des rapports de tests

- **Utiliser les formats propriétaires lorsqu'ils suffisent**

- **Problématique : se ramener à des formats XML connus**
  - Logs propriétaires des robots -> XML
  - Utilisation de techniques de transformation
    - XSLT
    - « Moulinette » propriétaire

- **Générations de rapport HTML**
  - Utilisation des tâches constructeurs type ant, maven, gradle pour générer des rapports HTML à partir de fichiers XML standards

  - XSLT : XML -> HTML



Home

Packages

org.training.junitbook.ch5

Classes

StringUtilsTest

Unit Test Results.

Designed for use w

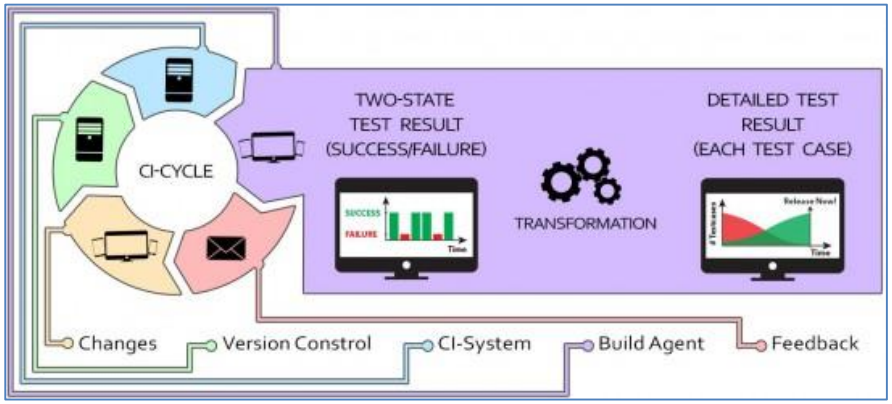
Class org.training.junitbook.ch5.StringUtilsTest

| Name | Tests | Errors | Failures | Time(s) | Time Stamp |
|---|---|---|---|---|---|
| StringUtilsTest | 10 | 0 | 0 | 0.063 | 2010-12-24T23:34:33 |

Tests

| Name | Status | Type |
|---|---|---|
| testIsBlankWithNonBlankData | Success | |
| testIsBlankWithEmptyString | Success | |
| testIsBlankWithNullString | Success | |
| testIsBlankWithWhiteSpaceString | Success | |
| testIsEmptyWithNonBlankData | Success | |
| testIsEmptyWithEmptyString | Success | |
| testIsEmptyWithNullString | Success | |
| testIsEmptyWithWhiteSpaceString | Success | |
| testReverse | Success | |
| testReverseWithNullString | Success | |

# Exemple : Robot Ranorex -> xUnit




```
<junitreport todir="${outputdir}">
    <fileset dir="${jrdir}">
        <include name="TEST-*.xml"/>
    </fileset>
    <report todir="${outputdir}/html"
        styledir="junitreport"
        format="frames">
        <param name="key1" expression="value1"/>
        <param name="key2" expression="value2"/>
    </report>
</junitreport>
```

On peut également utiliser la tâche **junitreport de ant**

myreport.rxlog

```
<xsl:for-each select="//activity[((@testcasename) or (@type='test case setup') or (@type='t

  <xsl:choose>

  <xsl:when test="parent::activity[@testcasename]">
      <!-- Do not report Setup/Teardown if this container is child of test case because tes
           Just only report global Setup/Teardown -->
  </xsl:when>
  <xsl:otherwise>

      <xsl:element name="testcase">
        <xsl:choose>
          <xsl:when test="@testcasename">
            <xsl:attribute name="classname">
              <xsl:text>RanorexTestReport.</xsl:text>
              <xsl:value-of select="@testcasename"/>
            </xsl:attribute>
          </xsl:when>
          <xsl:otherwise>
            <xsl:choose>
              <xsl:when test="(@type='test case setup') or (@type='test case teardown')">
                <xsl:attribute name="classname">
                  <xsl:text>RanorexTestReport.Setup/Teardown</xsl:text>
                </xsl:attribute>
```

Fichier XSL

myreport.xml
au format xUnit

Permet à jenkins (Par exemple) de consolider les résultats des tests provenant de logiciels propriétaires

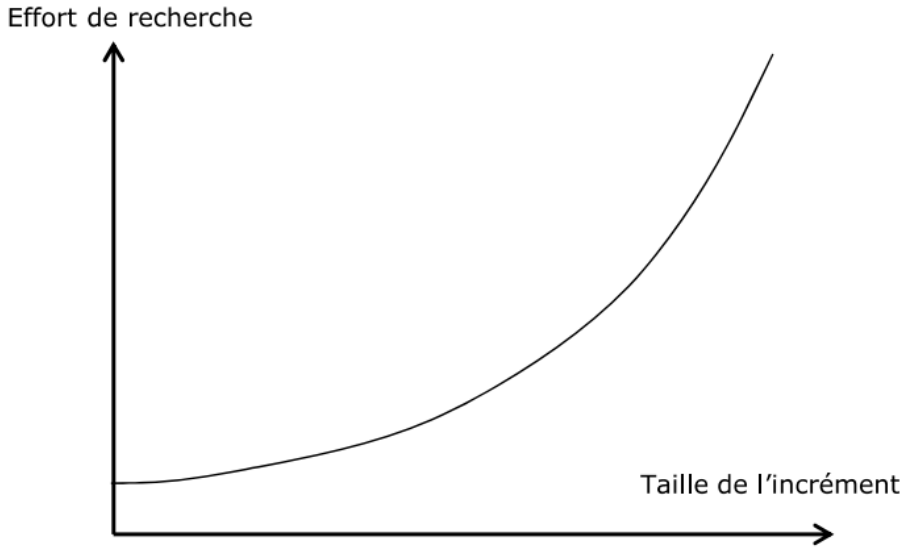
# Gestion des anomalies

- Utilisation de gestionnaires d'anomalies
  - Jira (Atlassian)
  - Mantis
  - Outils intégrés dans les suites ALM commerciales

- Génération automatique des anomalies
  - Récupération du contexte par programmation
    - Copie d'écran, log, résultats attendus
  - Utilisation des API/Web Services des gestionnaires d'anomalies, des référentiels de campagnes de tests
  - Dangereux sur les bases officielles d'anomalies
    - Passer par une base tampon avec validation humaine puis procéder par export/import


# L’intégration continue

- Le constat : plus la taille de l’incrément est grande plus l’effort de recherche et de réparation des défauts est grand

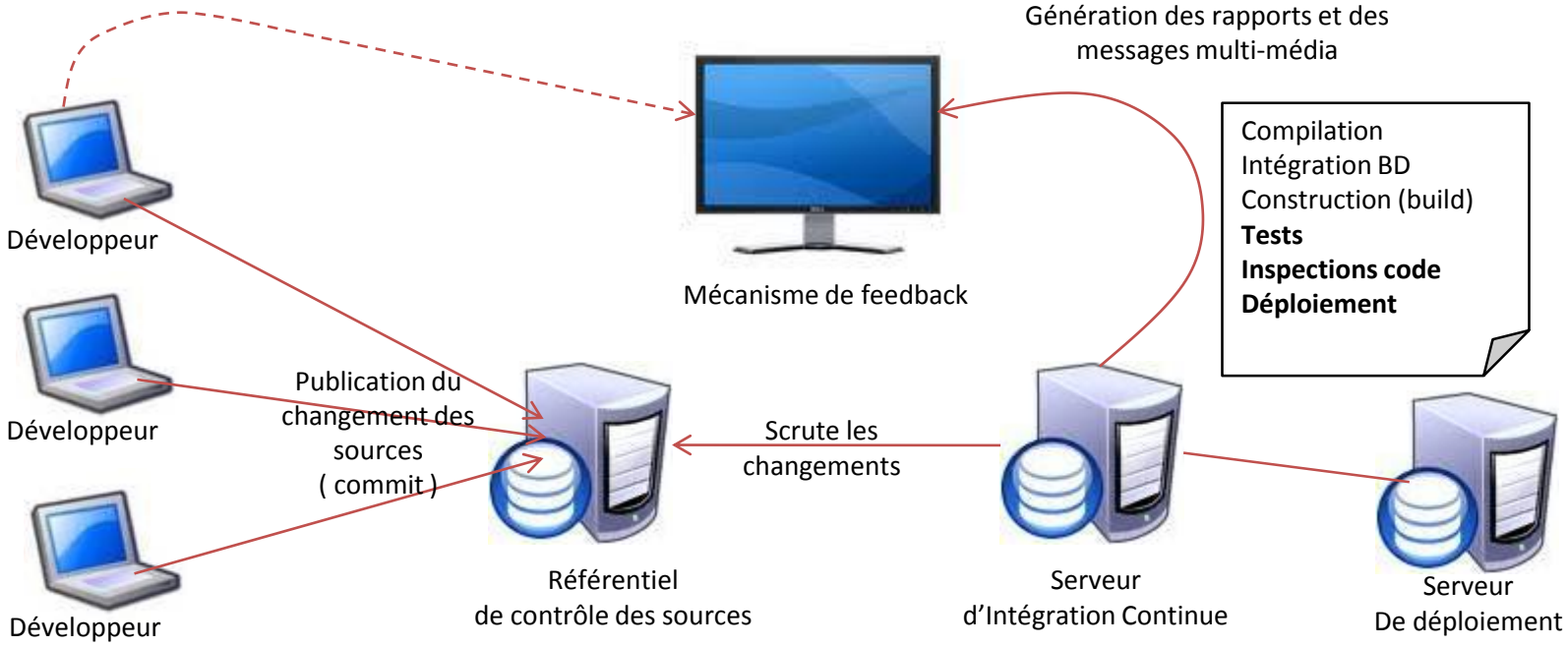



Il faut donc tester très souvent voir CONTINUEMENT !

C’est le rôle du serveur d’intégration continue que de lancer les chaînes de tests automatisés de manière régulière sans intervention humaine.

# Intégration Continue de base

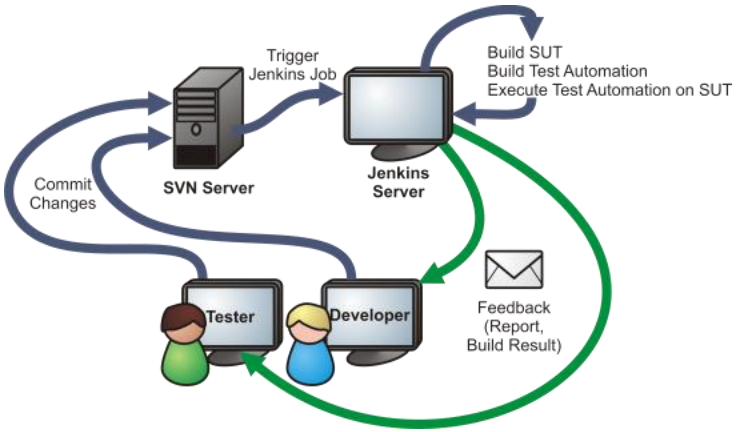
- Le changement constitue un risque.
- Le changement est inévitable
- Le changement est très fréquent
- L'IC propose de construire, de contrôler et de déployer automatiquement le logiciel dès qu'il y a un changement significatif ( qu'est ce qu'un changement significatif ? )



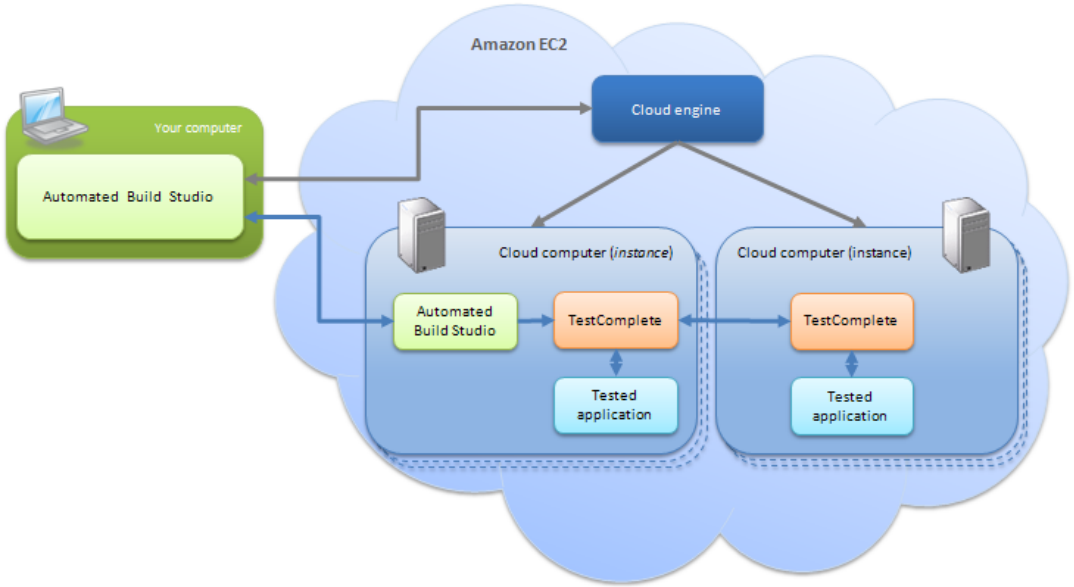
# IC et Tests de validation

- Lancement des robots de tests
  - Ligne de commande pour le lancement des robots
  - Modes silencieux
- Lancement des tests codés avec des bibliothèques
  - Code de pilotage des navigateurs (Selenium Web Driver)
  - xUnits utilisés pour la validation

Jenkins/Hudson
Continuum
Cruise Control



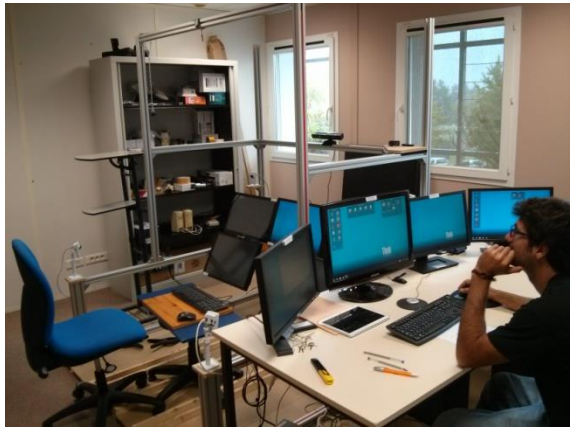

Exemple avec Jenkins




Exemple avec Automated Build Studio
de SmartBear sur le cloud.

# Plateforme de tests

- **L'infrastructure technique**
  - Représentativité (hardware, software) ?
  - Machines Virtuelles ( Virtual Box, VMWare, … ) ?
  - Plateforme de développement, d'intégration, de validation de pré-production ?



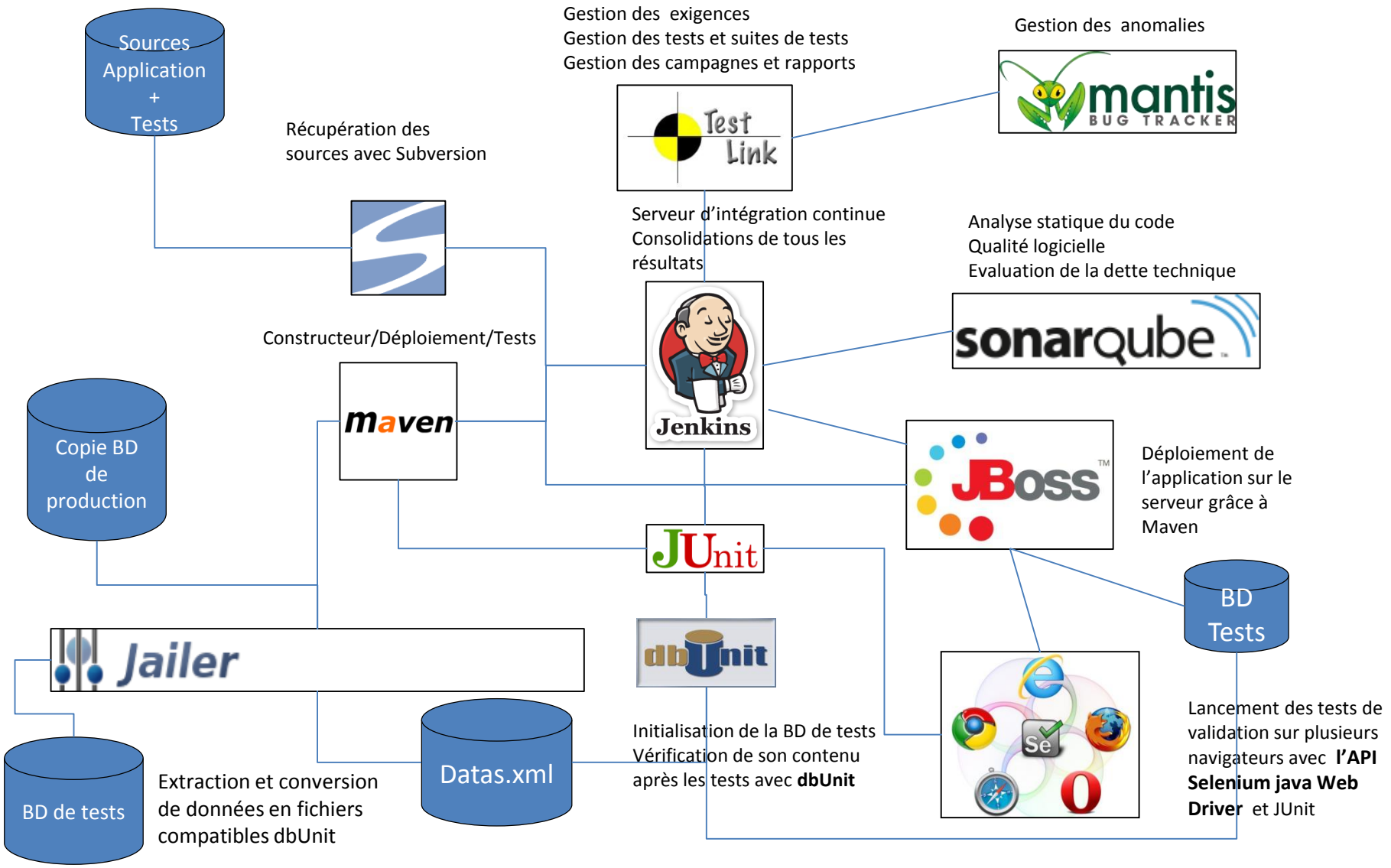
- **Les logiciels et outils**
  - Pour gérer la partie software
  - Pour gérer la partie testware

- **Plusieurs options**
  - Ensemble de logiciels libres (Problématique de l'interconnexions des outils)
    - Attention à la maintenance
  - Ensemble de suites de logiciels commerciales (HP, Microsoft, SmartBears, Micro Focus, …)
    - Attention au coût des licences et au coût de support
  - Solutions mixtes
  - Locations de plateformes prêtes à l'emploi « On the cloud »

- **De plus en plus d'environnement basés sur des logiciels libres sur des projets professionnels dans des milieux réputés « difficiles » ( Banque Assurance, Téléphonie, …)**

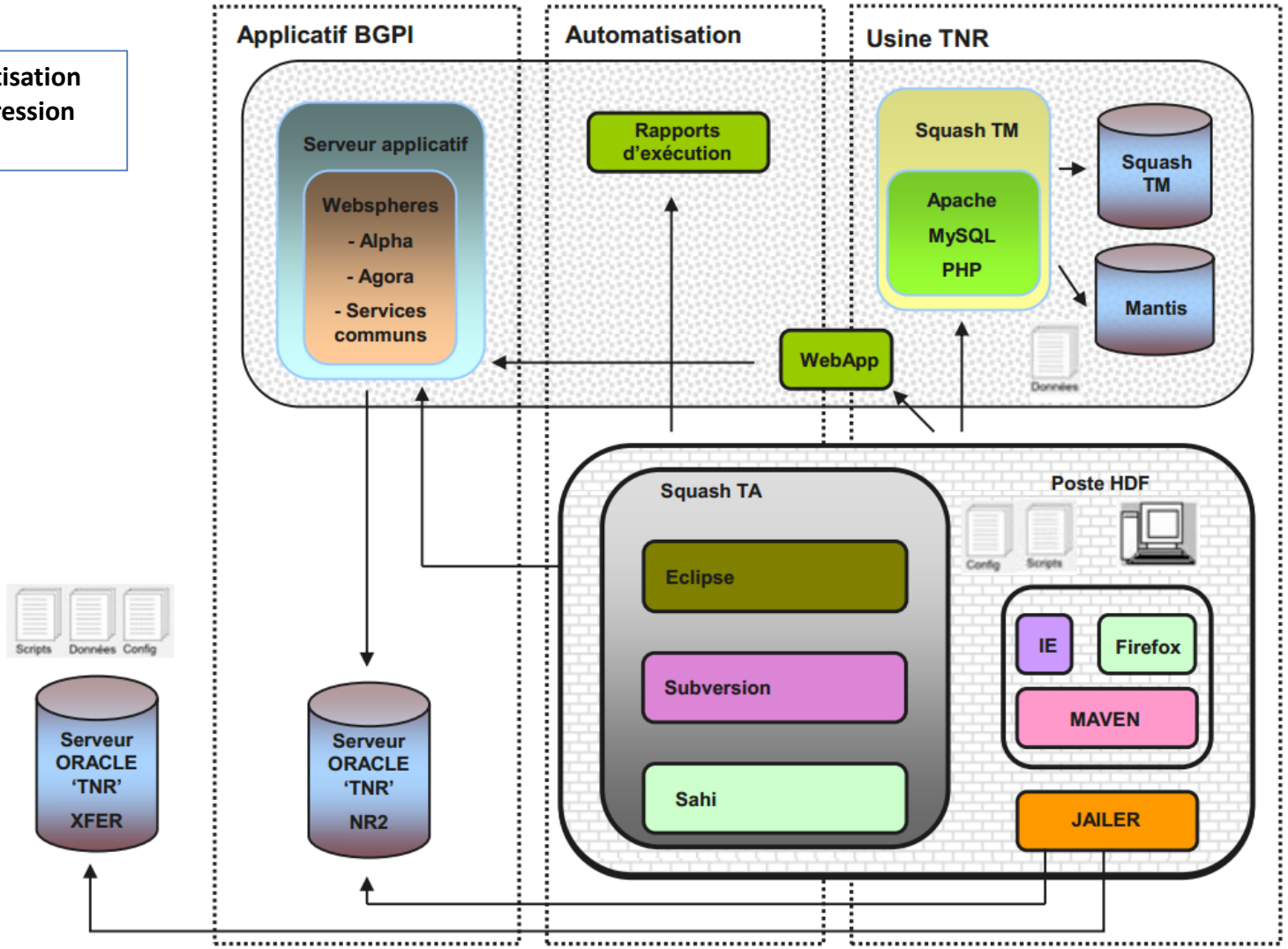
- **Une offre "on the cloud" pour tester les applications mobiles**

# Exemple de chaîne d'outils 100% open source dans le monde java

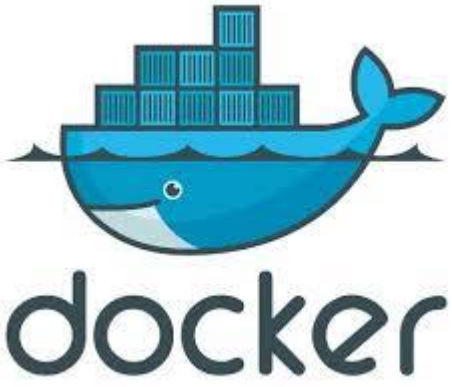


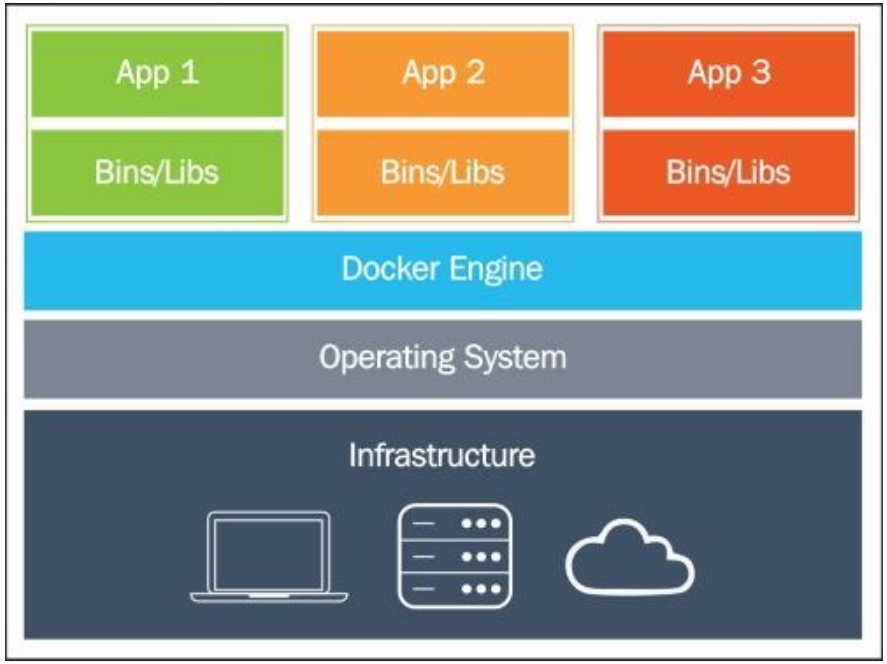
# Une solution avec Squash TM/TA

**Une usine d'automatisation des tests de non régression avec Squash**



# Utilisation de Docker pour les configurations multiples




Permet d'avoir des environnements **indépendants avec leurs propres variables d'environnements**.

Ces environnements peuvent néanmoins **communiquer les uns avec les autres et peuvent avoir des adresses IP différentes**

**Exemple d'utilisation pour selenium GRID :**
3 instances de Selenium server, 1 **hub** et 2 **nœuds** paramétrés avec des configurations systèmes différentes (Chrome et Firefox)

# Architecture Microsoft TFS