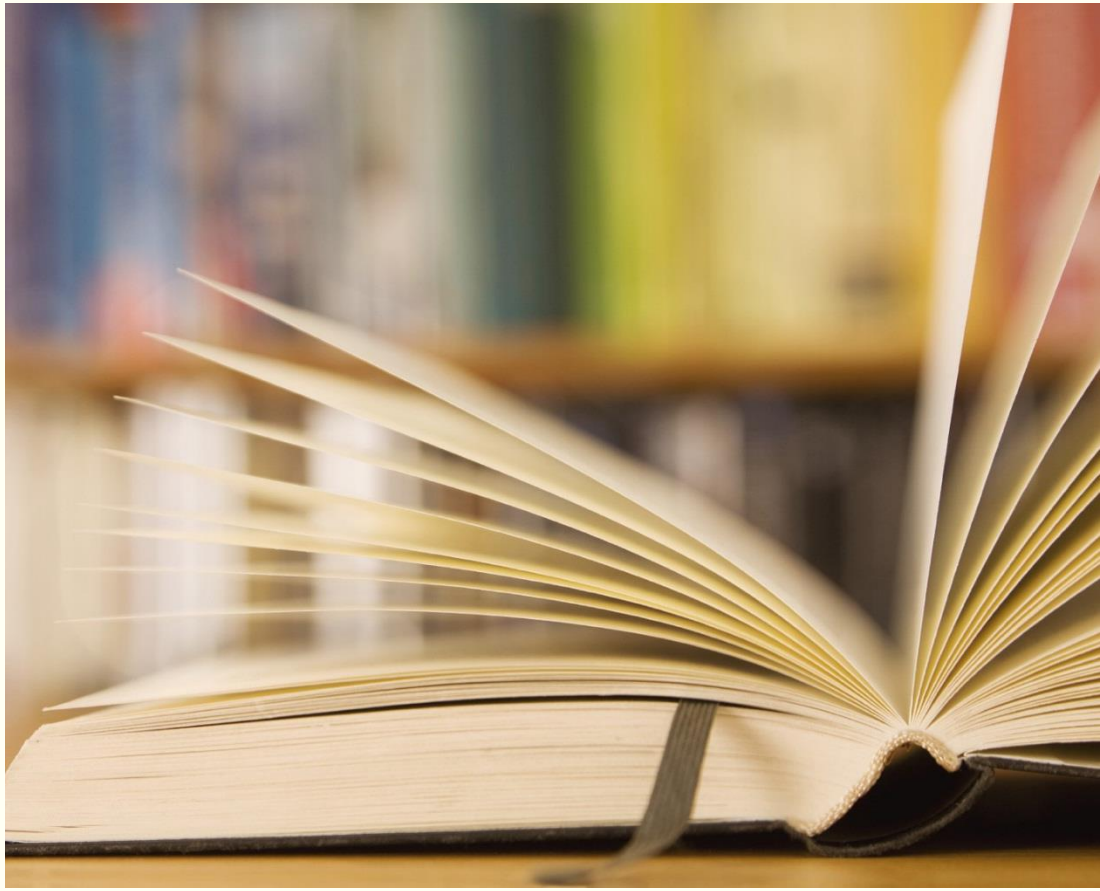
FORMATION JENKINS

2 jours



Disposition de titre et de contenu avec liste

- **Module 1: Introduction à l'intégration continue**
- **Module 2: Mise en place du serveur Jenkins**
- **Module 3: Intégration continue avec Jenkins**
- **Module 4: Inspection continue avec Jenkins**
- **Module 5: Déploiement continue avec Jenkins**
- **Module 6: Le plugin Pipeline as a code**
- **Module 7: Architecture Maître Esclave**
- **Module 8: Administration d'un serveur Jenkins**

MODULE 1

Introduction à l'intégration continue

Présentation

- Définition
- Intégration traditionnelle vs Intégration Continue
- Les pratiques de l'intégration continue
- Les principaux avantages

Définition

- ***"L'intégration continue est un ensemble de pratiques utilisées en génie logiciel. Elles consistent à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression de l'application en cours de développement."***

Wikipedia

- ***L'intégration continue est une pratique de développement logiciel où les membres d'une équipe intègrent leur travail fréquemment. En général, chacun intègre au moins quotidiennement. Chaque intégration est vérifiée par un système automatisé.***

Martin Fowler

Explications

- L'intégration continue est une méthode de développement de logiciel DevOps avec laquelle les développeurs intègrent régulièrement leurs modifications de code à un référentiel centralisé, suite à quoi des opérations de création et de test sont automatiquement menées.

- L'intégration continue désigne souvent l'étape de création ou d'intégration du processus de publication de logiciel, et implique un aspect automatisé (un service d'IC ou de création) et un aspect culturel (apprendre à intégrer fréquemment).

- Les principaux objectifs de l'intégration continue sont de trouver et de corriger plus rapidement les bogues, d'améliorer la qualité des logiciels et de réduire le temps nécessaire pour valider et publier de nouvelles mises à jour de logiciels.

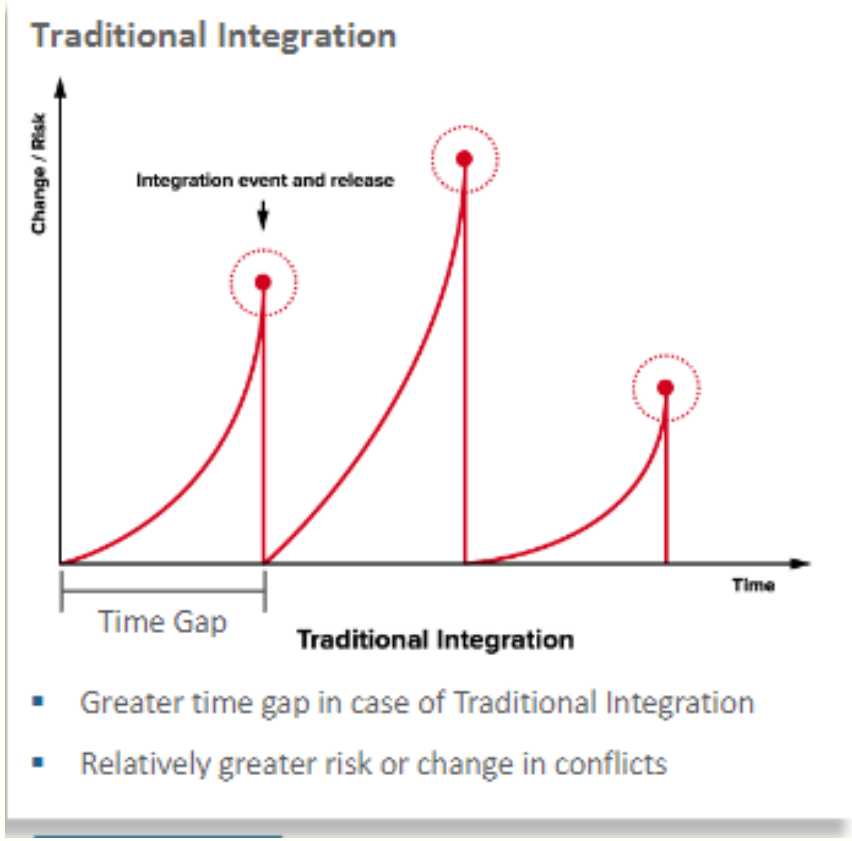
Pourquoi l'intégration continue est-elle nécessaire ?

- Autrefois, les développeurs au sein d'une même équipe avaient tendance à travailler séparément pendant de longues périodes et à n'intégrer leurs modifications au référentiel centralisé qu'après avoir fini de travailler.

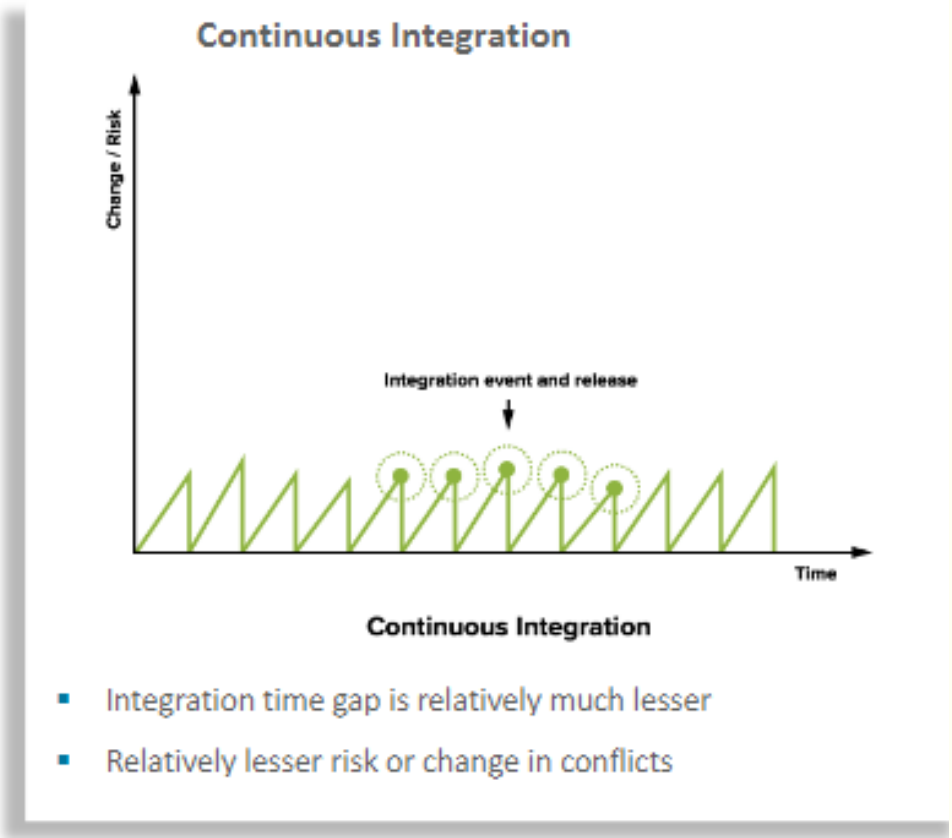
- Cela a rendu la fusion de changement de codes difficile et chronophage, et a également entraîné des bogues pendant une longue période, sans correction.

- La combinaison de ces différents facteurs empêchait de livrer rapidement des mises à jour aux clients.

Intégration traditionnelle vs Intégration Continue



- Greater time gap in case of Traditional Integration
- Relatively greater risk or change in conflicts

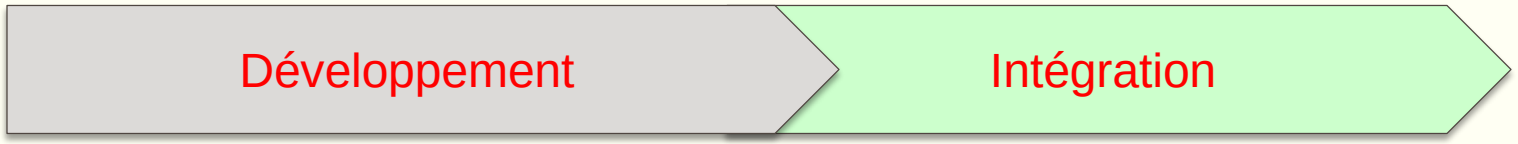


- Integration time gap is relatively much lesser
- Relatively lesser risk or change in conflicts

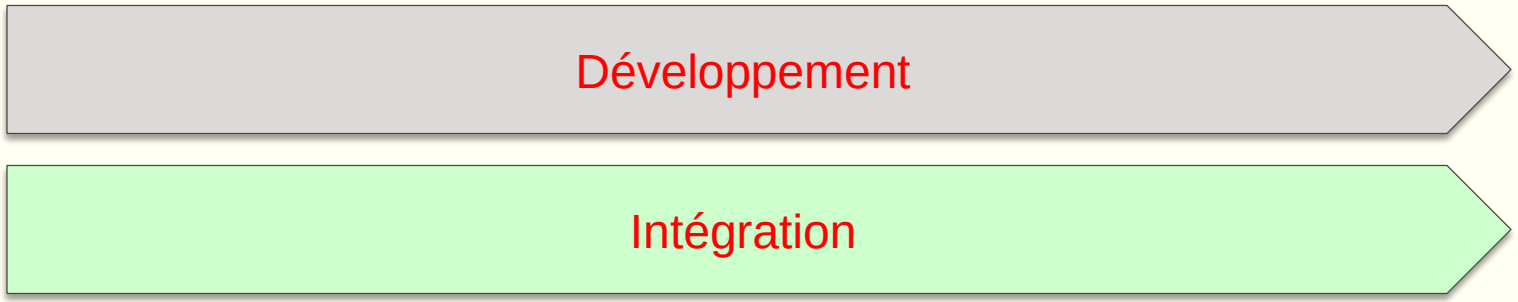
Intégration/Intégration Continue

- Intégration : on code tout, puis on teste tout.
- Intégration continue : on code, on commit, on intègre…

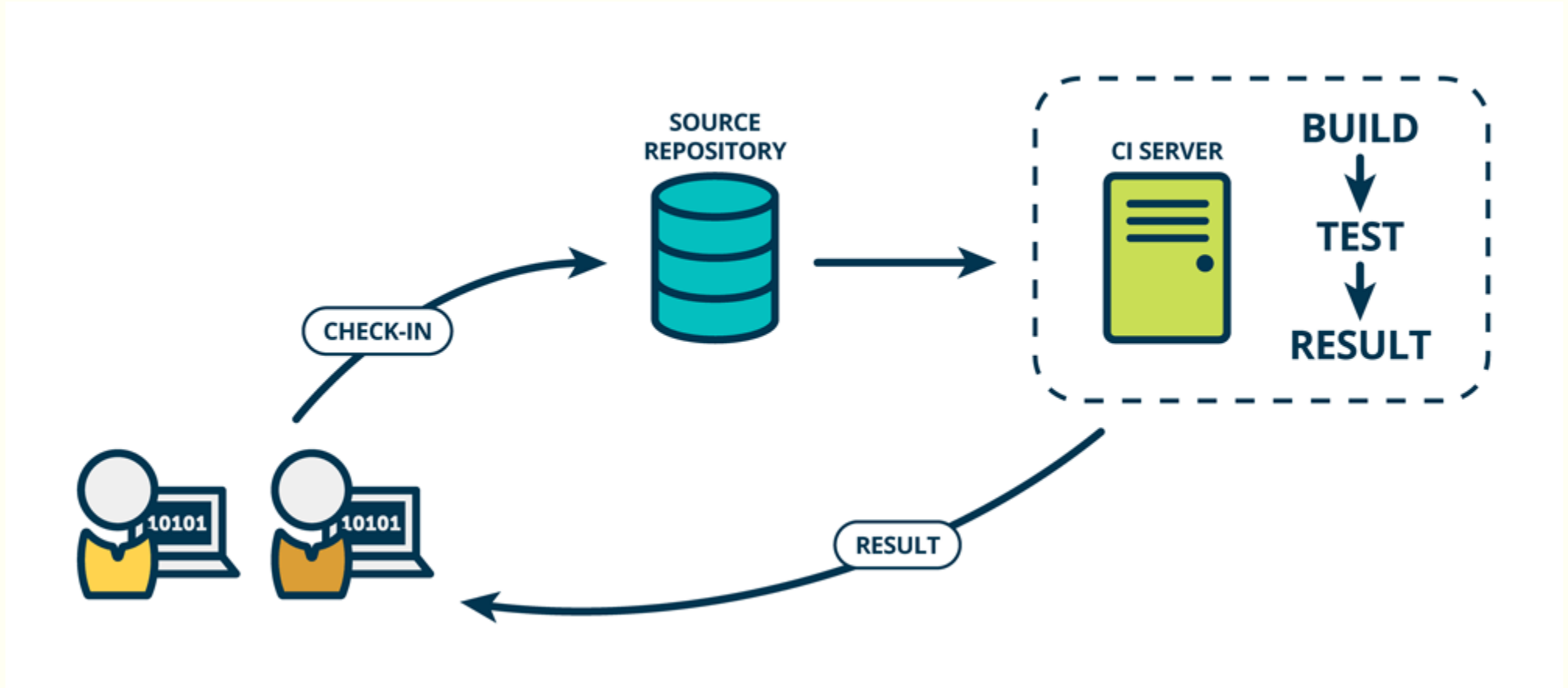
Intégration classique :



Intégration continue :



Les pratiques de l'intégration continue



Comment fonctionne l'intégration continue ?

- Avec l'intégration continue, les développeurs appliquent régulièrement leurs modifications sur un référentiel partagé, avec un système de contrôle des versions comme Git.

- Avant d'envoyer leur code, les développeurs peuvent choisir d'exécuter des tests sur des unités locales pour le vérifier davantage avant son l'intégration.

- Un service d'intégration continue crée et exécute automatiquement des tests unitaires sur les nouveaux changements de codes pour détecter immédiatement n'importe quelle erreur.

Intégration continue

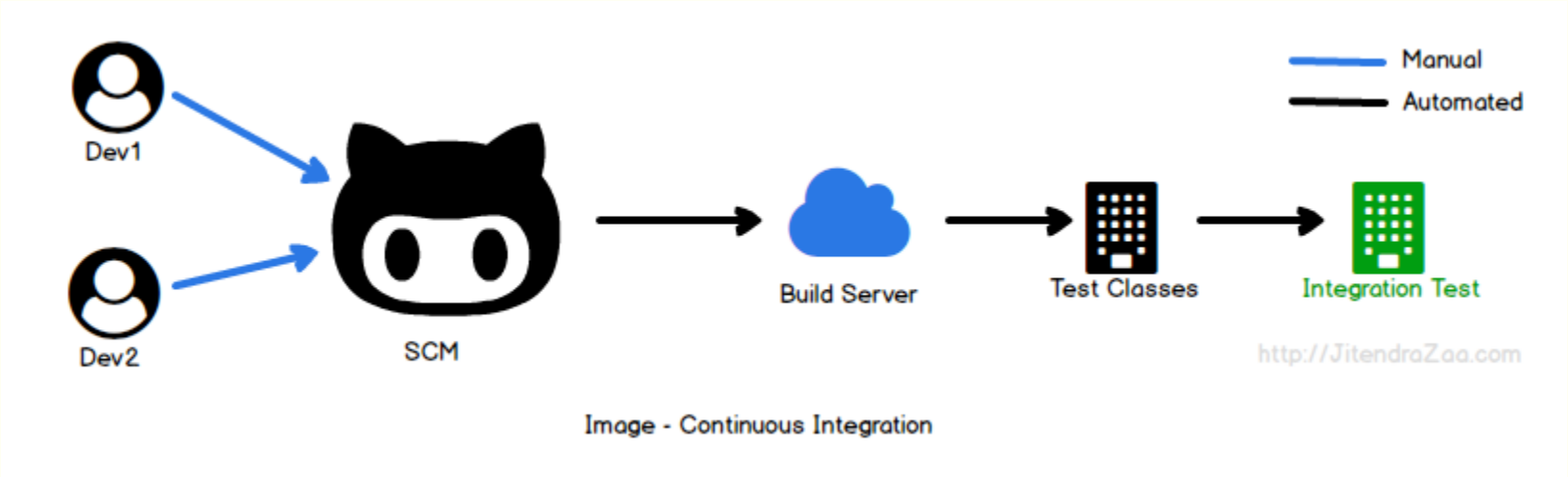


Image - Continuous Integration

- *L'intégration continue désigne les étapes de création et de test d'unité du processus de publication de logiciel.*
- *Chaque révision appliquée déclenche un processus de création et de test automatisé.*

Livraison continue

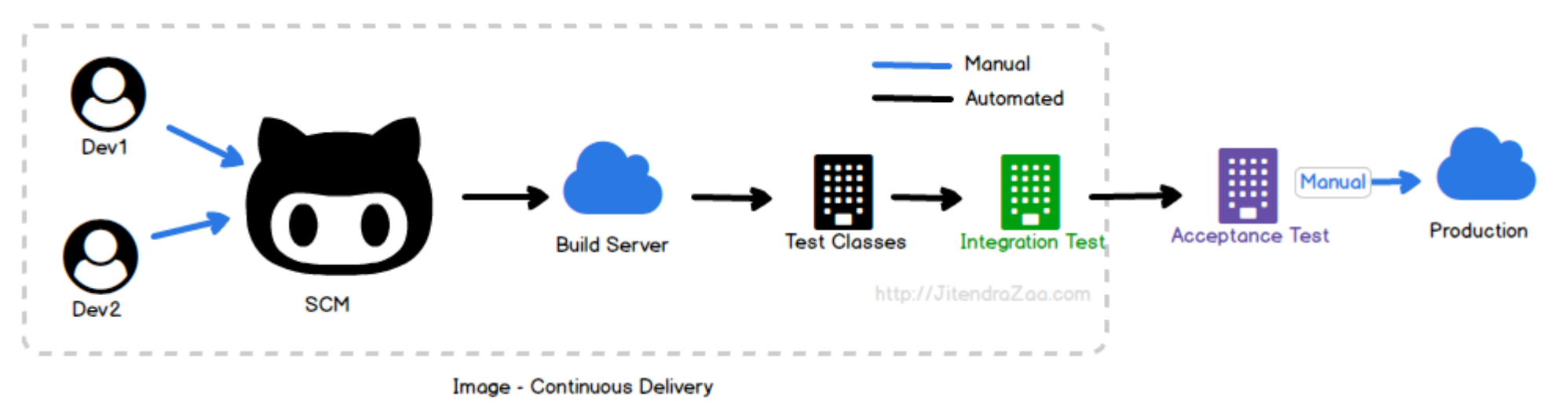


Image - Continuous Delivery

- Avec la livraison continue, les modifications de code sont automatiquement appliquées, testées et préparées à la production.
- La livraison continue étend le principe de l'intégration continue en déployant tous les changements de code dans un environnement de test et/ou un environnement de production après l'étape de création.

Déploiement continue

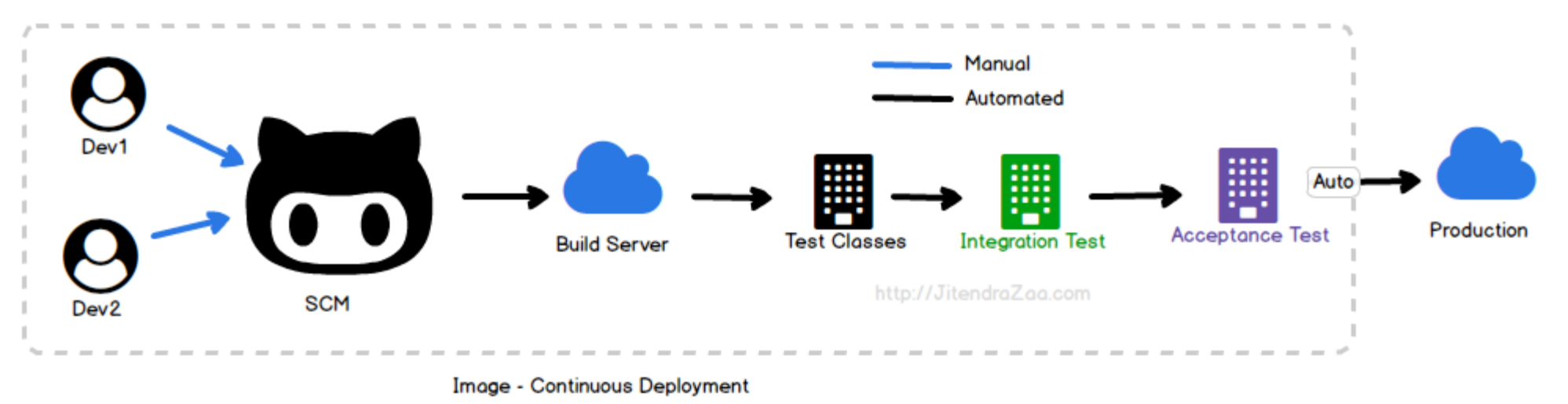
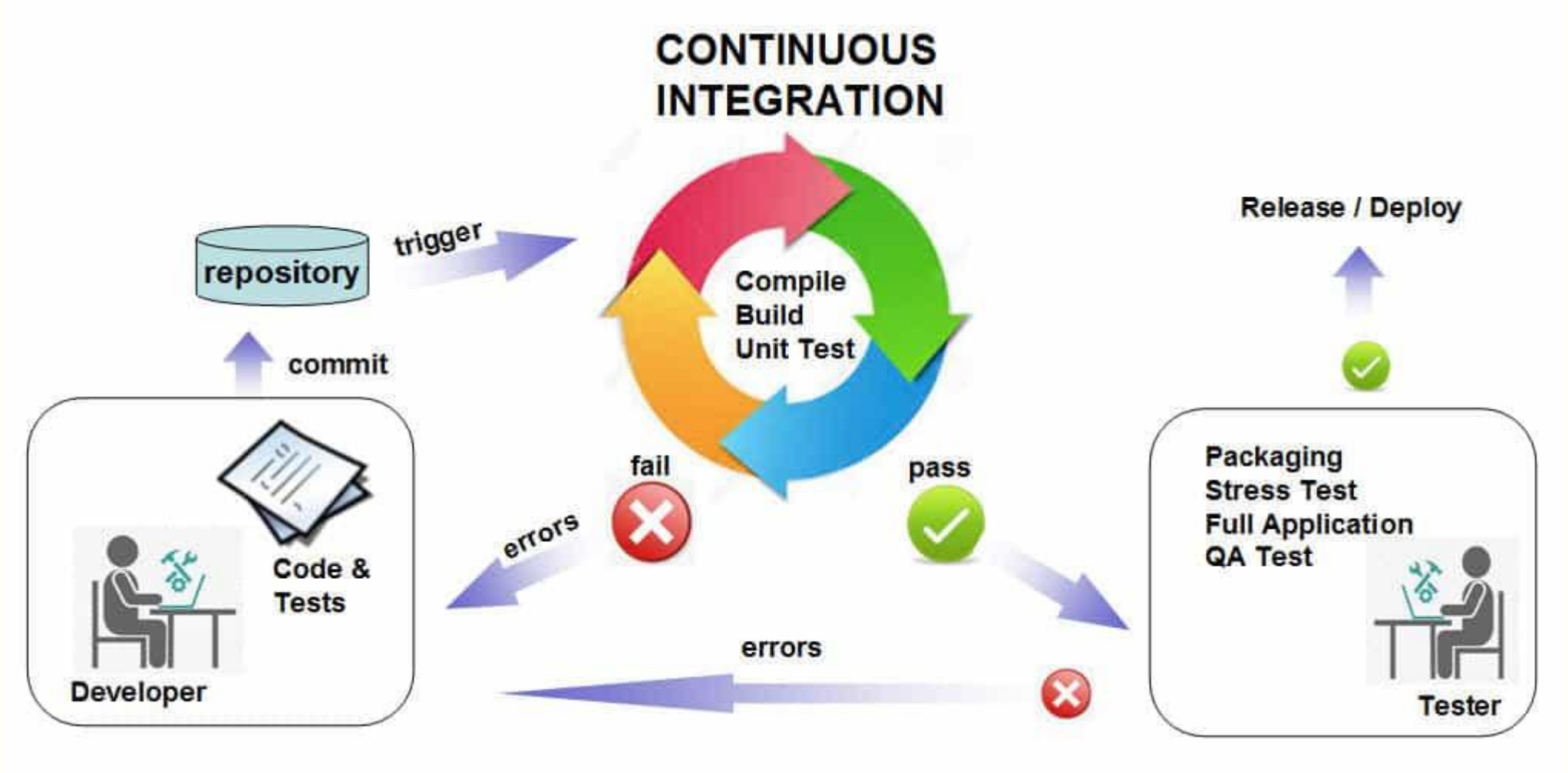


Image - Continuous Deployment

Principes


CONTINUOUS
INTEGRATION
repository
trigger
Compile
Build
Unit Test
Release / Deploy
commit
fail
pass
errors
Code &
Tests
Packaging
Stress Test
Full Application
QA Test
errors
Developer
Tester

Les principaux avantages

Trouver et corriger plus rapidement les bogues

- Avec des tests plus fréquents, votre équipe peut découvrir et corriger plus rapidement les bogues avant qu'ils ne prennent de l'ampleur.

- Les tests automatisés mis en place sur l'application, et joués à chaque intégration, permettent d'**identifier rapidement les changements problématiques**

Les principaux avantages

- **Améliorer la productivité des développeurs**

- L'intégration continue aide votre équipe à gagner en productivité, en limitant de nombre de tâches manuelles devant être accomplies par les développeurs et en encourageant les comportements qui contribuent à réduire le nombre d'erreurs et de bogues dans les versions publiées auprès des clients.
- Les problèmes d'intégration sont détectés rapidement, et peuvent donc être corrigés au fil de l'eau, sans avoir à attendre une passe d'intégration manuelle qui n'a lieu que trop rarement

Les principaux avantages

- **Livrer plus rapidement des mises à jour**

- L'intégration continue aide votre équipe à livrer plus rapidement et plus fréquemment des mises à jour après des clients.
- La dernière version stable de l'application est connue, et peut rapidement être obtenue *(pour tests, démonstration, ...)*.

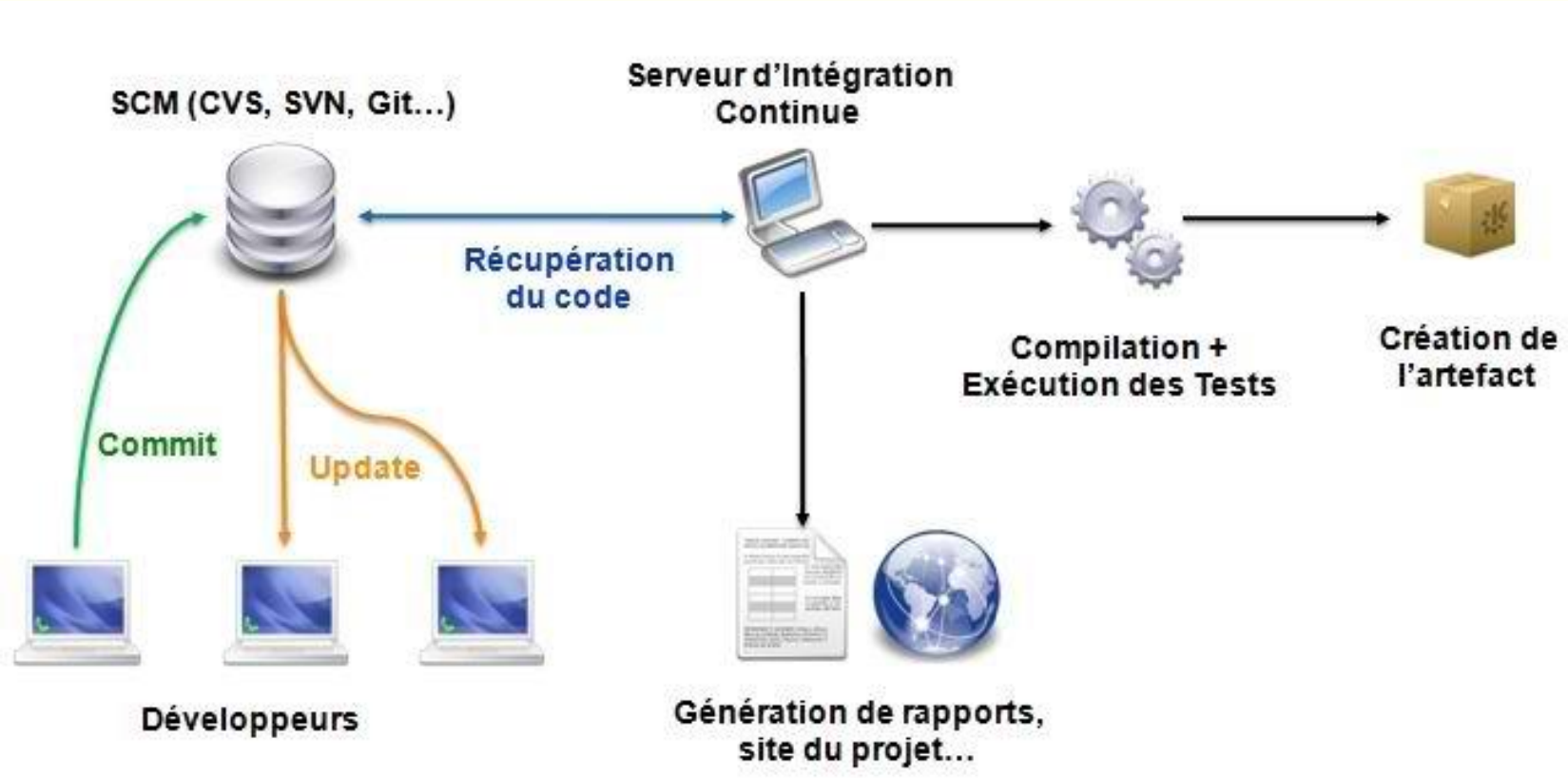
Architecture

- L'architecture comporte différents éléments n'intervenant pas uniquement dans l'intégration continue mais plutôt dans des équipes de développement en général.

Elle se compose donc :

- Un gestionnaire de code source
- Un logiciel d'intégration continue
- Une équipe de développeurs
- Un outil de reporting ou/et d'un serveur de suivi de bug

Architecture



Fonctionnement

Explication du fonctionnement de l'intégration continue en 4 étapes

Etape 1

- Le développeur code sa fonctionnalité ou le module qui lui a été défini.
- Après avoir conçu son module, il réalise des tests unitaires sur sa machine afin de s'assurer que tout fonctionne correctement dans son environnement.
- Il récupère le code sur le gestionnaire de source pour mettre à jour le code qu'il a.
- Il fusionne son code avec le code qu'il a récupéré puis résoud les conflits, teste de nouveau son code sur sa machine et apporte des corrections éventuelles.
- Si tout est ok, il publie alors son code via son gestionnaire de code source.

Fonctionnement

- **Etape 2**
- Le serveur d'intégration possède un service de détection de modification de code, suite à la dernière publication du développeur il prépare une tâche qui consiste à récupérer le dernier fragment de code développé et à l'intégrer dans sa plateforme.
- Il exécute cette tâche appelée communément **Job**.

Etape 3

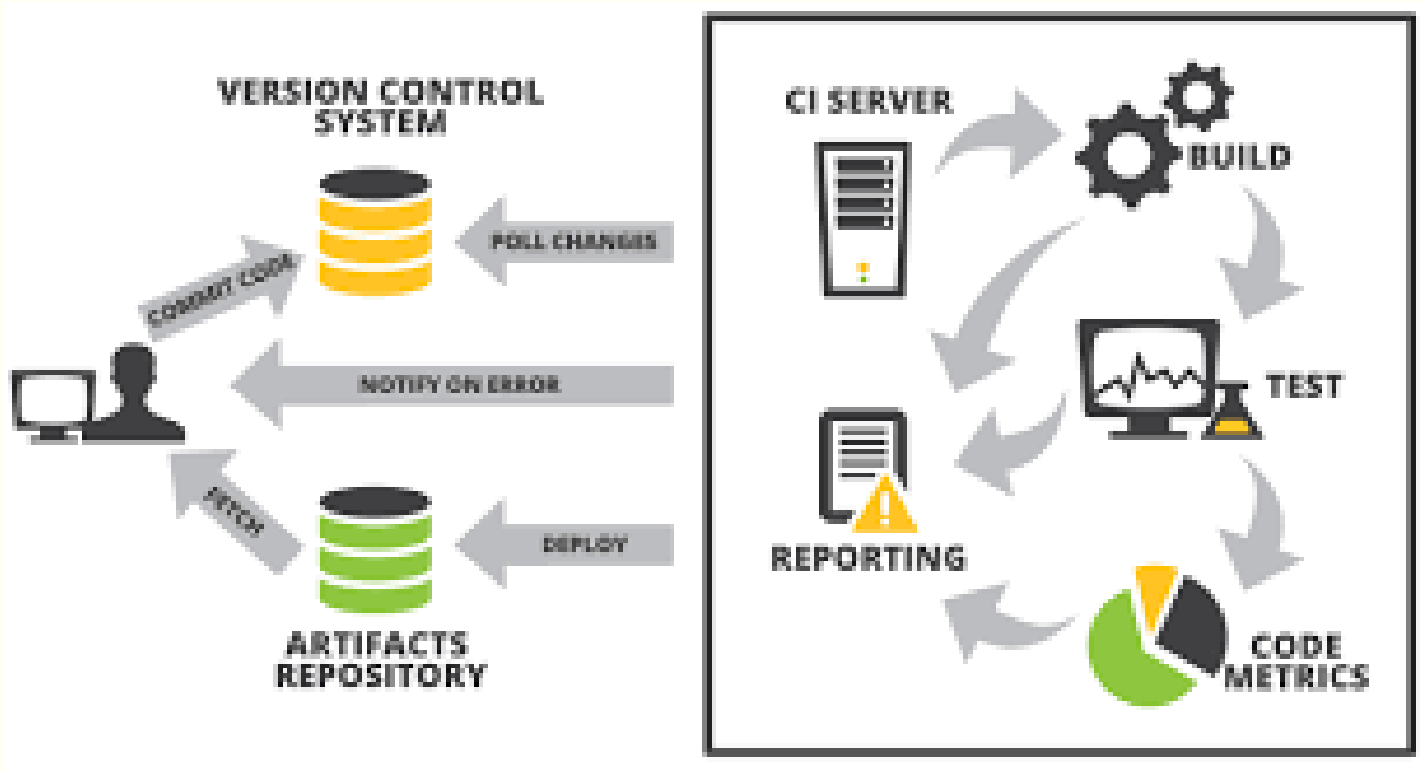
- Une fois le job terminé, des rapports portant sur la qualité, la stabilité ou encore les divers bugs pouvant être rencontrés, sont générés et transmis à l'ensemble de l'equipe ou juste au développeur.

Etape 4

- L'équipe peut alors consulter ces rapports, les analyser, traiter des bugs s'il y en a, puis continuer à développer les autres phases du projet.

Les outils

- Un gestionnaire de code source
- Un gestionnaire de builds
- Un gestionnaire de tests
- Un gestionnaire de logs



Un gestionnaire de code source

- Un gestionnaire de code source dit "Source Code Manager" est un outil qui permet de centraliser du code sur un dépot commun dit "Repository".
- Il en existe plusieurs types tels que
  - ✓ CVS,
  - ✓ SVN,
  - ✓ Git,
  - ✓ Mercurial
  - ✓ ...
- C'est à partir de cet outil que le développeur va pouvoir récupérer le code de l'équipe mais aussi de publier le sien.
- Par la suite, l'outil d'intégration continue va récupérer l'ensemble du code publié pour l'exécuter sur sa plateforme grâce à un gestionnaire de builds.

Un gestionnaire de builds

- Un gestionnaire de build est un outil qui permet d'éxecuter un script qui contient une suite d'objectifs souvent appelés cibles ou "target".
- Chaque target a un rôle bien particulier par rapport au code qu'il accompagne.
- Certain target prépare l'environnement de compilation, d'autre le compile ou encore le nettoie.
- On peut aussi générer la documentation à partir des commentaires du code.
- Les outils les plus répandus sont ceux de la fondation Apache tels que :
  - ✓ ANT,
  - ✓ MAVEN
  - ✓ ou IVY.

Un gestionnaire de tests

- Un outil de gestion de tests est généralement un framework qui s'appuie sur le type de langage avec lequel le code a été conçu.

- Le principe étant de jouer des scénarios de tests avec l'utilisation de jeux de données. Ces données peuvent varier selon le test voulu.

- Il peux avoir des cas de tests passant ou non passant. On programme à l'avance le résultat attendu pour chaque test en fonction du jeu de données.

- Si le résultat obtenu diffère de celui attendu, le build signalera des erreurs à la fin de son exécution.

Un gestionnaire de logs

- La gestion de logs est un élément tout aussi important que les autres.
- Il permet de stocker les informations et les traces produites par l'exécution des jobs.
- Toutes ces informations peuvent servir ensuite pour les chefs de projet, les équipes mais aussi pour comprendre certain bugs découverts.
- On les couple souvent à un bug tracker pour le suivi de bugs.
- Le gestionnaire de logs peut aussi se charger de transmettre par mail le résultat de chaque build à des utilisateurs prédéfinis dans le job.

DevOps et l’intégration continue

Les serveurs d’intégration continue

- Travis
- CircleCI
- Bamboo
- Codeship
- Jenkins

Disposition de titre et de contenu avec liste

Question ?

MODULE 2

Mise en place du serveur Jenkins

Présentation

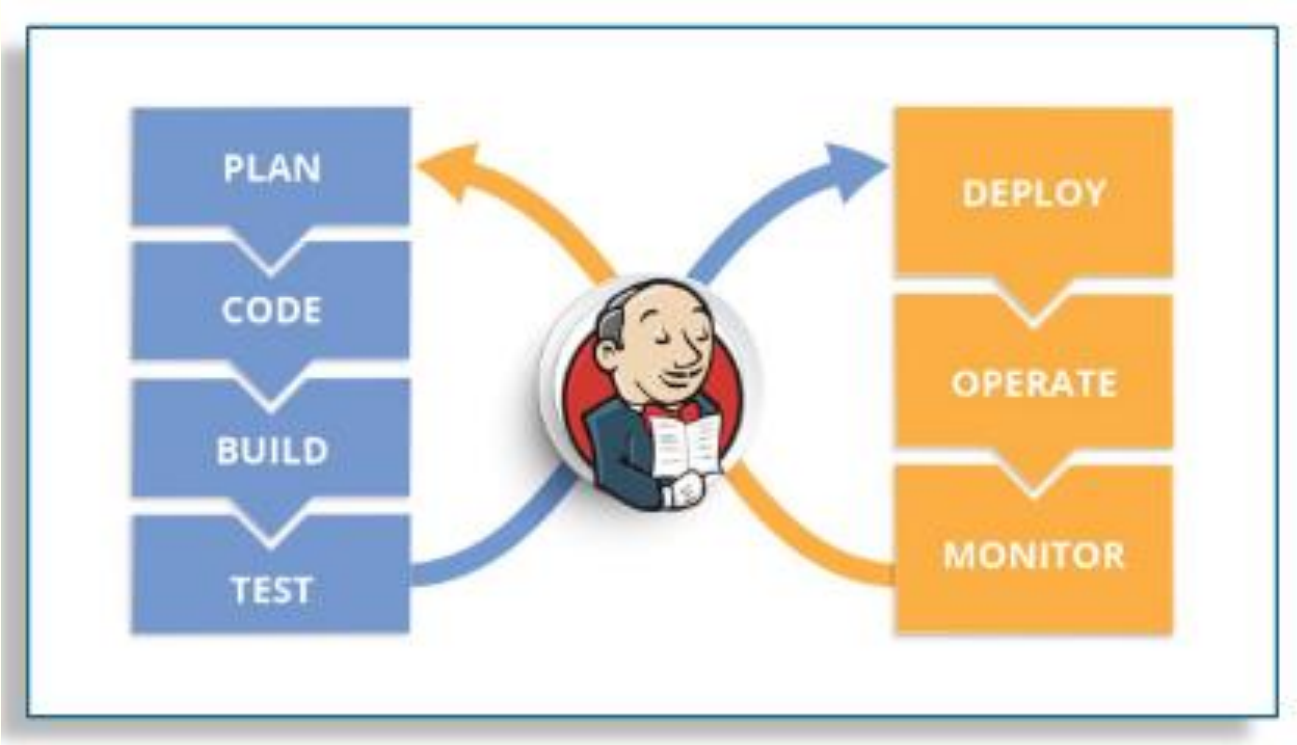
Introduction

Jenkins *est un outil open source d'intégration continue, fork de l'outil* ***Hudson*** *après les différends entre son auteur, Kohsuke Kawaguchi, et Oracle. Écrit en Java, Jenkins fonctionne dans un conteneur de servlets tel qu'Apache Tomcat, ou en mode autonome avec son propre serveur Web embarqué.*

Source : Wikipédia.

Jenkins : qu’est-ce que c’est ?

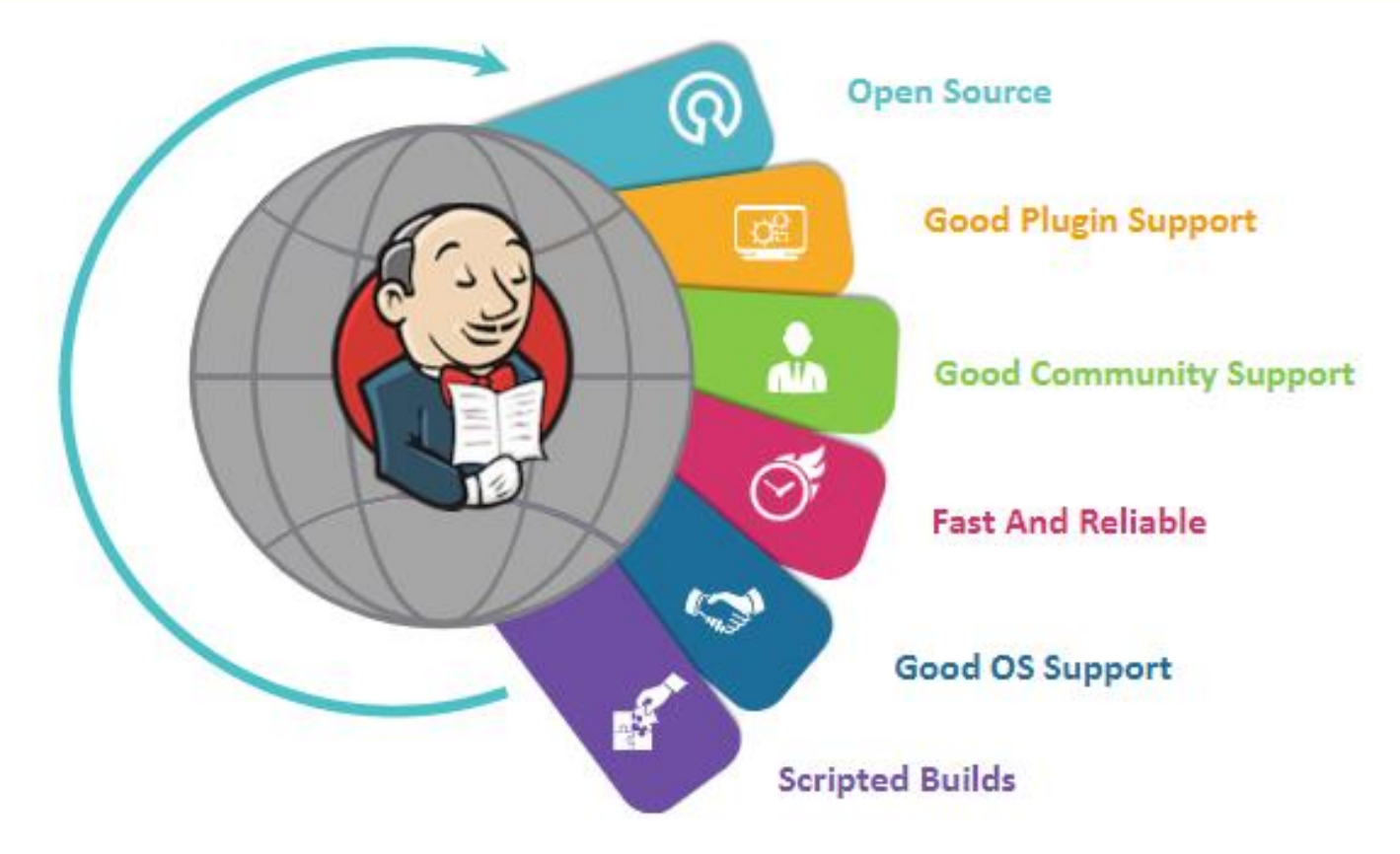
- Jenkins est un outil logiciel d’intégration continu.
- Jenkins permet l’intégration de toutes les étapes du cycle de développement.
  - Plan
  - Code
  - Build
  - Test
  - Deploy
  - Oprate
  - Monitor



Jenkins : qu’est-ce que c’est ?

- Il s’agit d’un **logiciel open source, développé à l’aide du langage de programmation Java**.
- Il permet de tester et de rapporter les changements effectués sur une large base de code en temps réel.
- En utilisant ce logiciel, les développeurs peuvent détecter et résoudre les problèmes dans une base de code et rapidement.
- Ainsi les tests de nouveaux builds peuvent être automatisés, ce qui permet d’intégrer plus facilement des changements à un projet, de façon continue.
- L’objectif de Jenkin est en effet d’accélérer le développement de logiciels par le biais de l’automatisation.

Quels sont les avantages de Jenkins ?



Fonctionnalités Jenkins

Il comporte les fonctionnalités suivantes :

- support d'outils de build: Suite Apache et script shell
- support de systèmes de gestion de versions :
  - CVS,
  - Subversion,
  - ClearCase,
  - Perforce,
  - Starteam,
  - Visual Source Safe,
  - CM Synergy,
  - Bazaar, Mercurial
- type de construction : manuel, programmé
- modèle de job : l'utilisateur peut définir ses propres constructions sur chaque projet

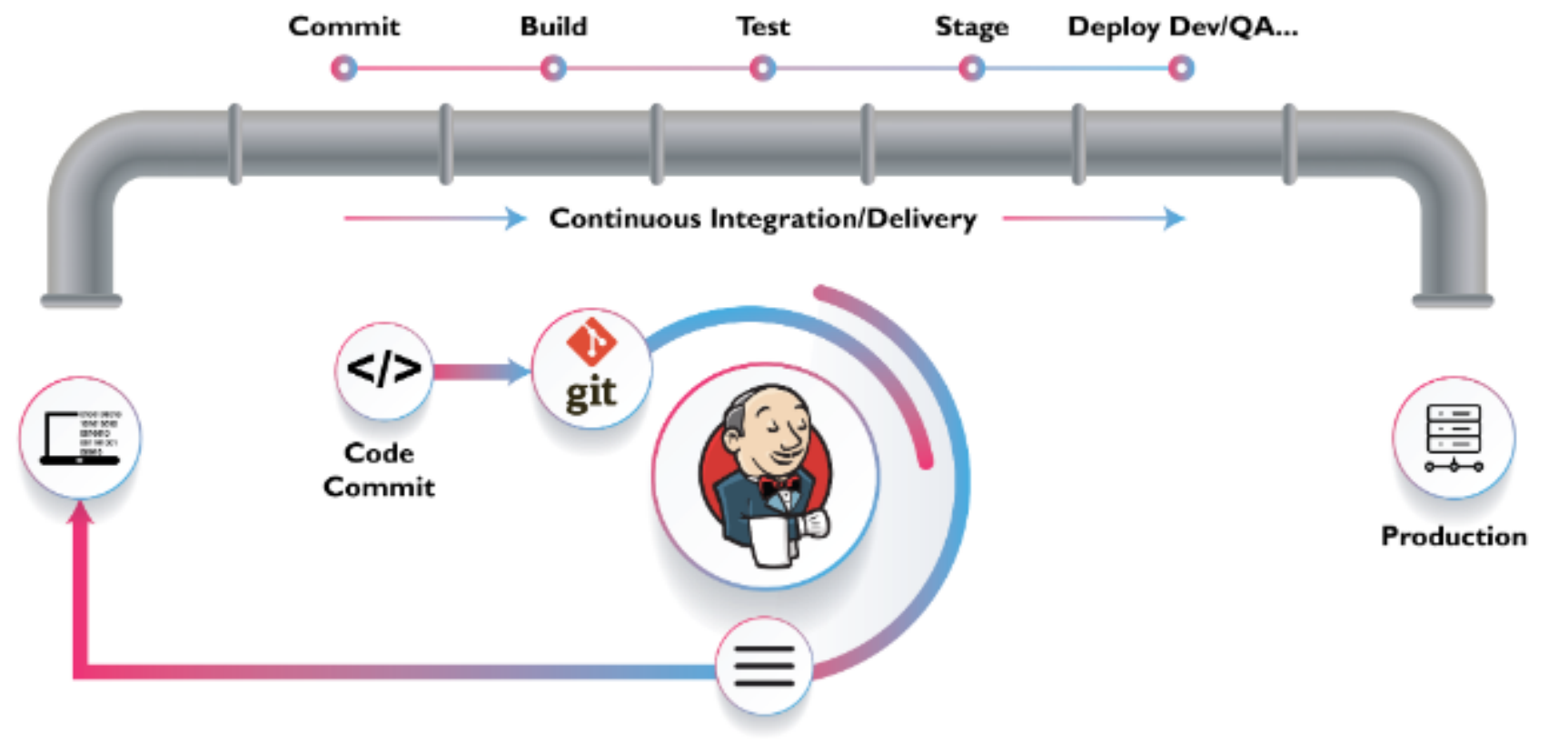
Fonctionnalités Jenkins

- modèle de job : l'utilisateur peut définir ses propres constructions sur chaque projet
- notification de résultat de la construction
- construction en parallèle grâce à la gestion de plusieurs files de construction
- construction en mode distribué grâce à des agents de construction
- Intégration avec différents outils de suivi d'incidents ou de bogues
- Exécution de scripts à distance par SSH
- Transfert de fichiers par SCP ou FTP
- Transfert d'artefacts vers un repository
- Exécution d'outils de contrôles de qualité de code

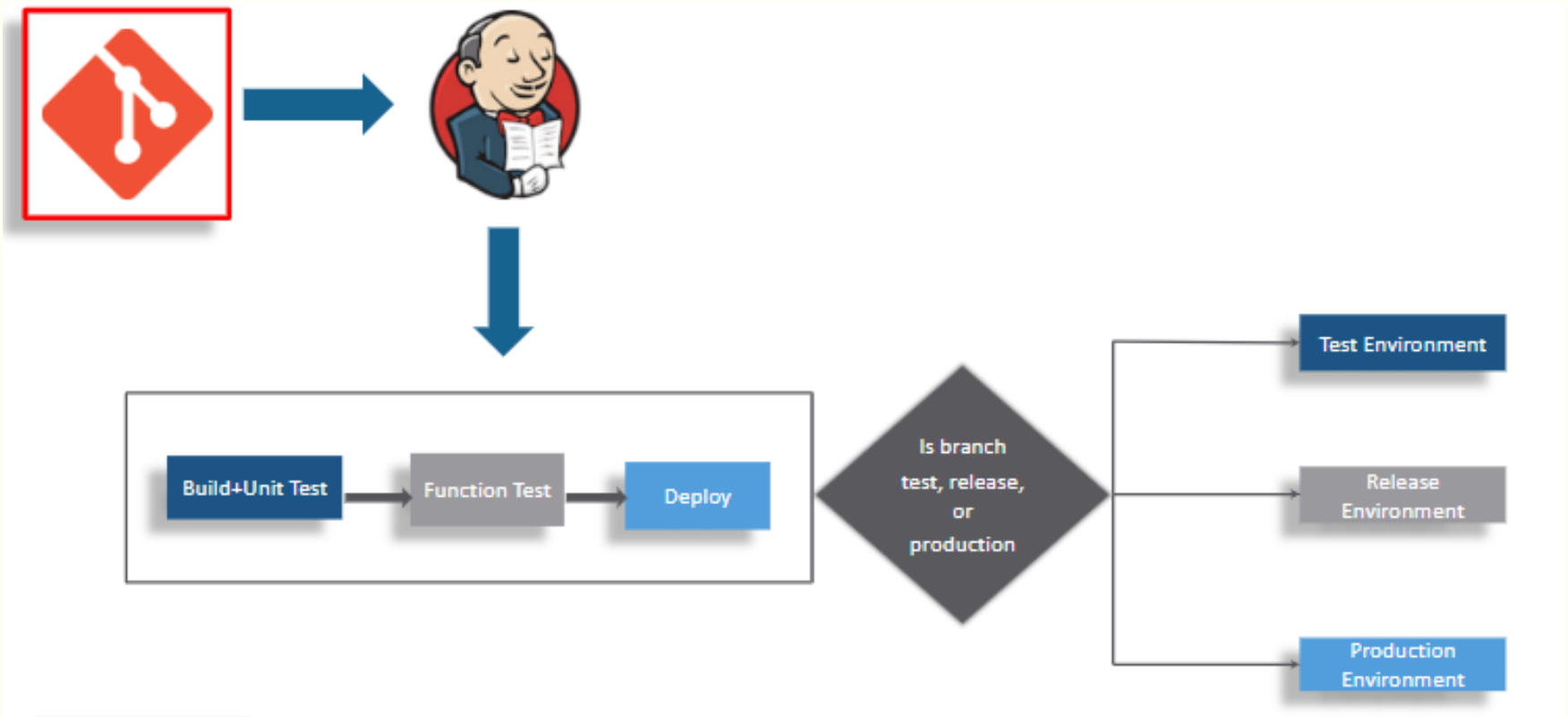
Features Of Jenkins

- Easy Installation Process
- Provides advance security
- Optimized Performance
- Upgrades are easily available
- Lightweight container support
- Distributed Team Management

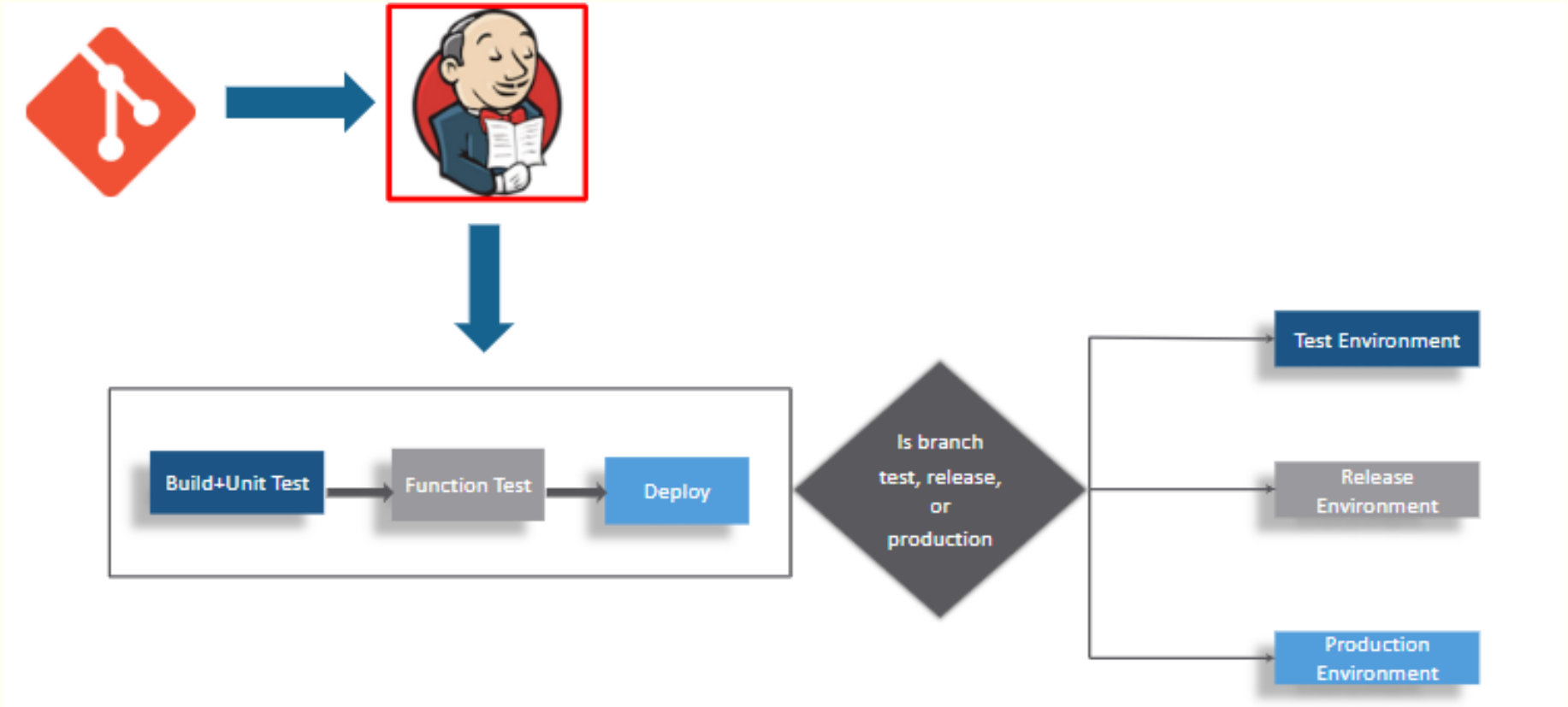
Rôle de Jenkins dans devops



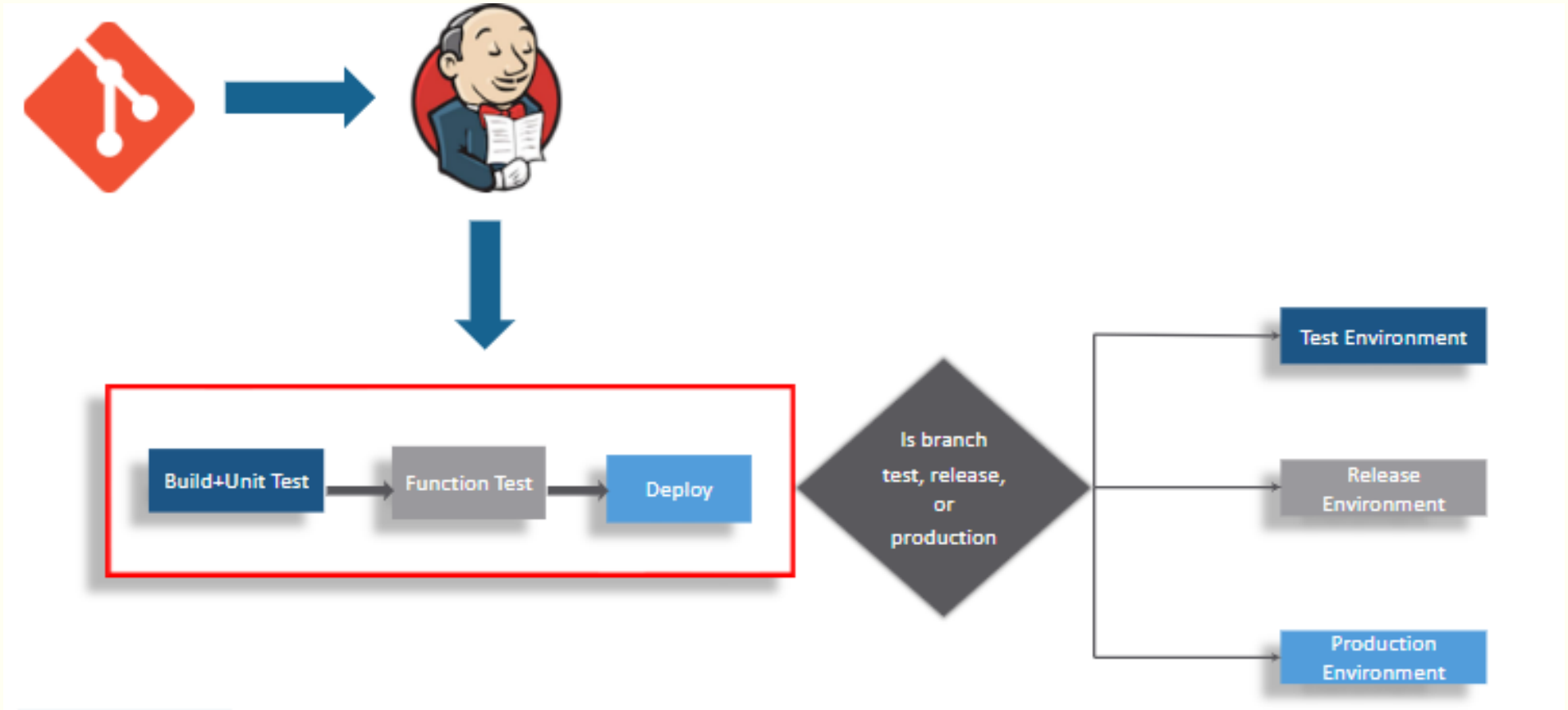
Architecture Jenkins: Source Control Management



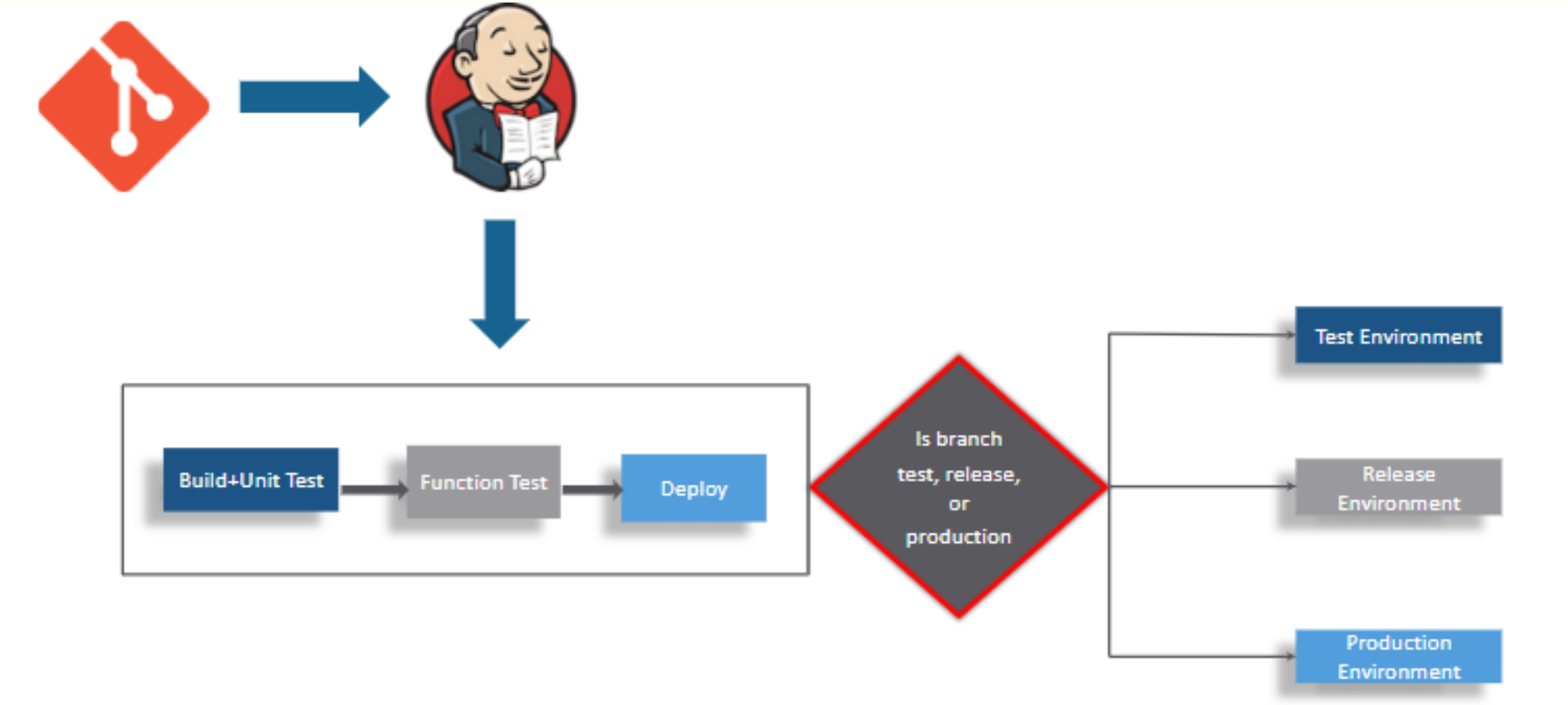
Architecture Jenkins: Jenkins Opreation



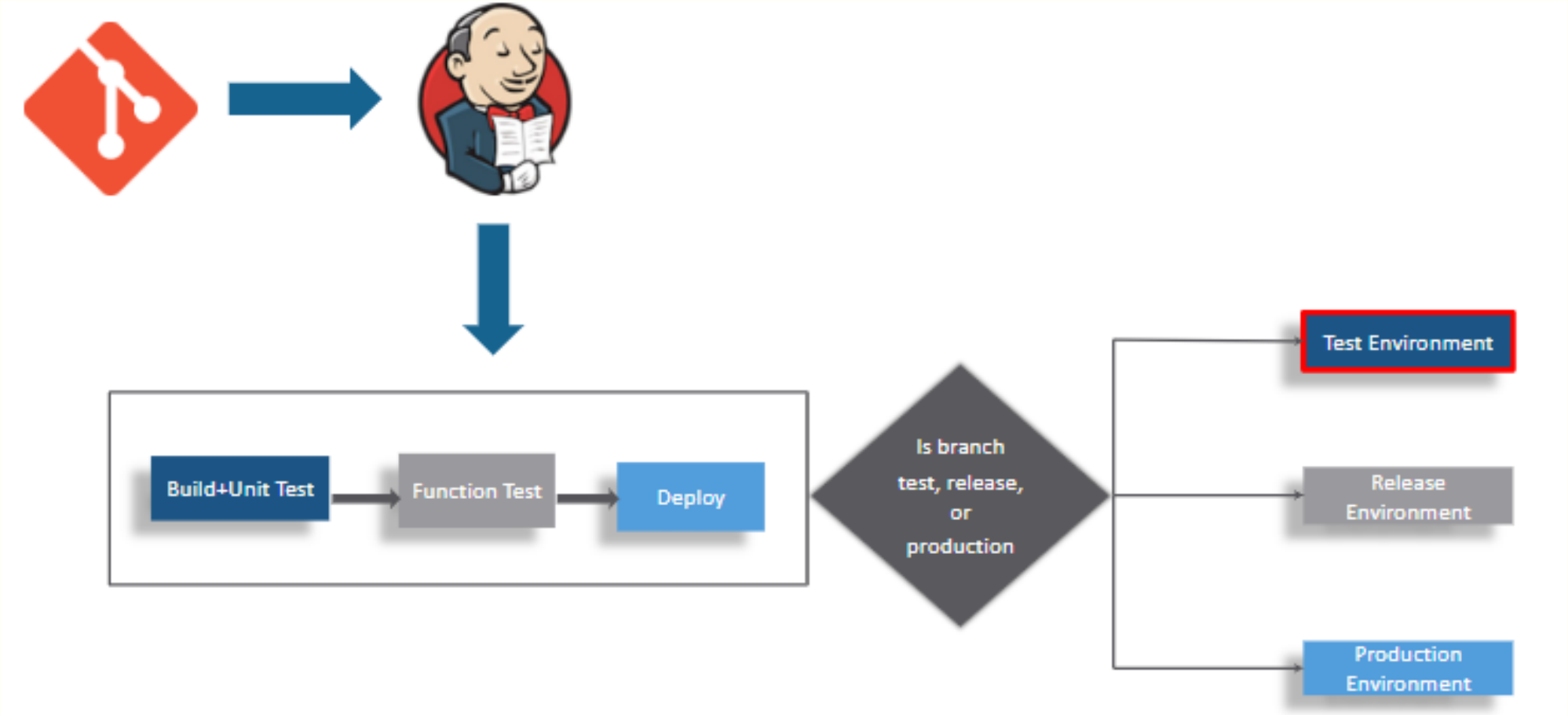
Architecture Jenkins: Build, Function & Test



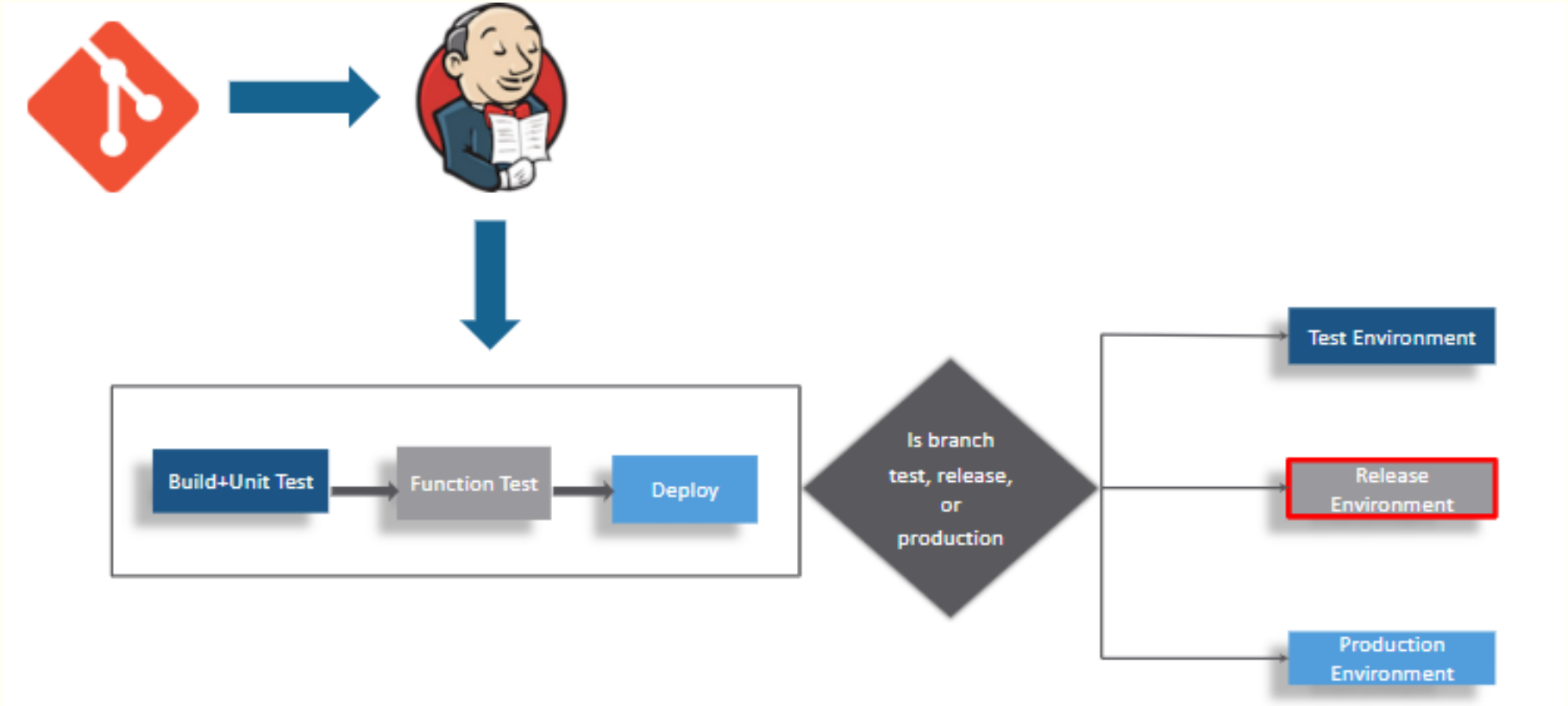
Architecture Jenkins: Conditon Check



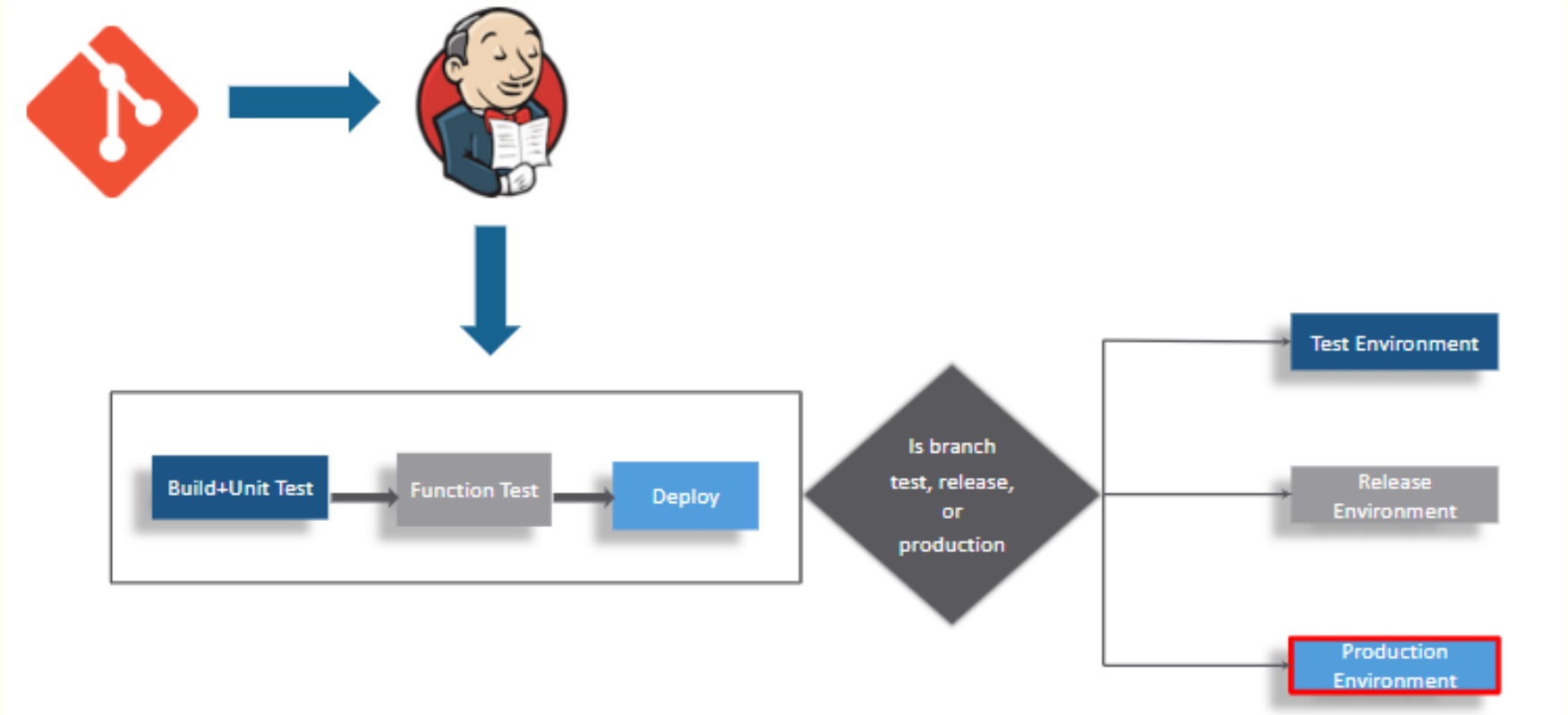
Architecture Jenkins: Deployed For Testing



Architecture Jenkins: Deployed For Release



Architecture Jenkins: Deployed For Testing



Installation de Jenkins : Ubuntu

- Vous avez besoin de quatre lignes de commandes seulement pour pouvoir lancer l'installation de Jenkins.
- Les lignes sont les suivantes :

```
wget -q -O -http://pkg.jenckins-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
```

Une fois l'installation effectuée, il ne vous manquera plus qu'à accéder à son interface par le biais d'un navigateur web. (A titre d'information, Jenkins écoute sur le port : 8080).

Prérequis => Avoir JDK Oracle d’installé

Installation de Jenkins : Docker

Installation de Jenkins : Windows

- Pour télécharger votre copie de Jenkins, allez directement sur la page **https://jenkins.io/download/** et sélectionner l'archive qui convient pour votre système d'exploitation.

- 2 façons de procéder pour installer Jenkins:
  - ✓ Vi**a l'archive war**
  - ✓ ou **via l'installeur windows**.

- Si vous avez opté de télécharger la version avec le zip contenant l'installeur windows. Pour effectuer l'installation, il suffit d'extraire et lancer le fichier exécutable, en suite suivre les indications pour terminer l'installation.

- L'archive war téléchargé est un conteneur tomcat qui peut être installé dans un serveur d'application comme tomcat. Il peut être aussi exécuté directement à partir de la commande suivante depuis une console windows.

Prérequis => Avoir JDK Oracle d'installé

Accès en client web

- Une fois Jenkins installé, il ne vous reste plus qu'à accéder, via votre navigateur web, à son interface.

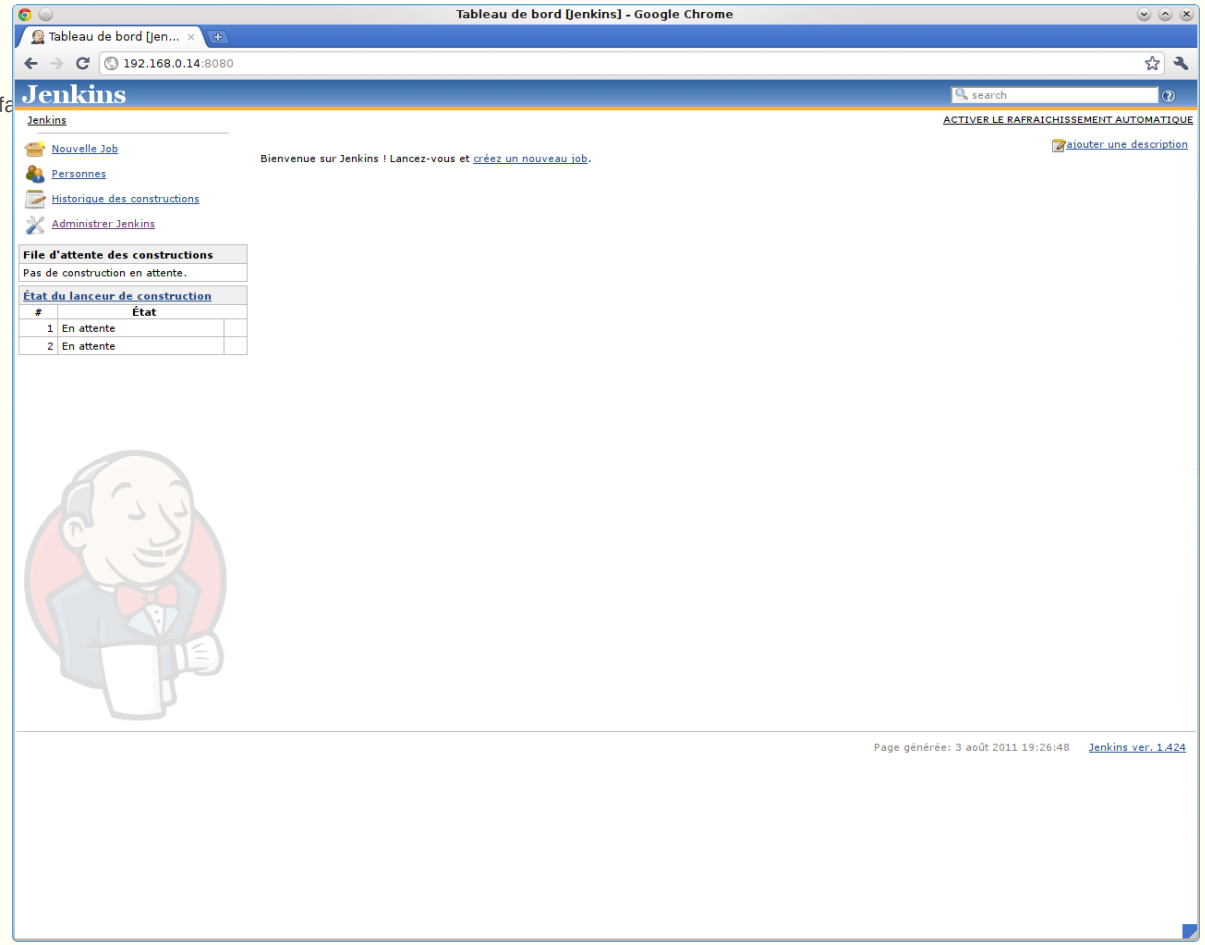
- Par défaut Jenkins, écoute sur le port 8080

- En fonction de l'adresse de votre serveur d'intégration, vous utiliserez une URL de ce type pour y accéder

- http://<ip machine>:8080

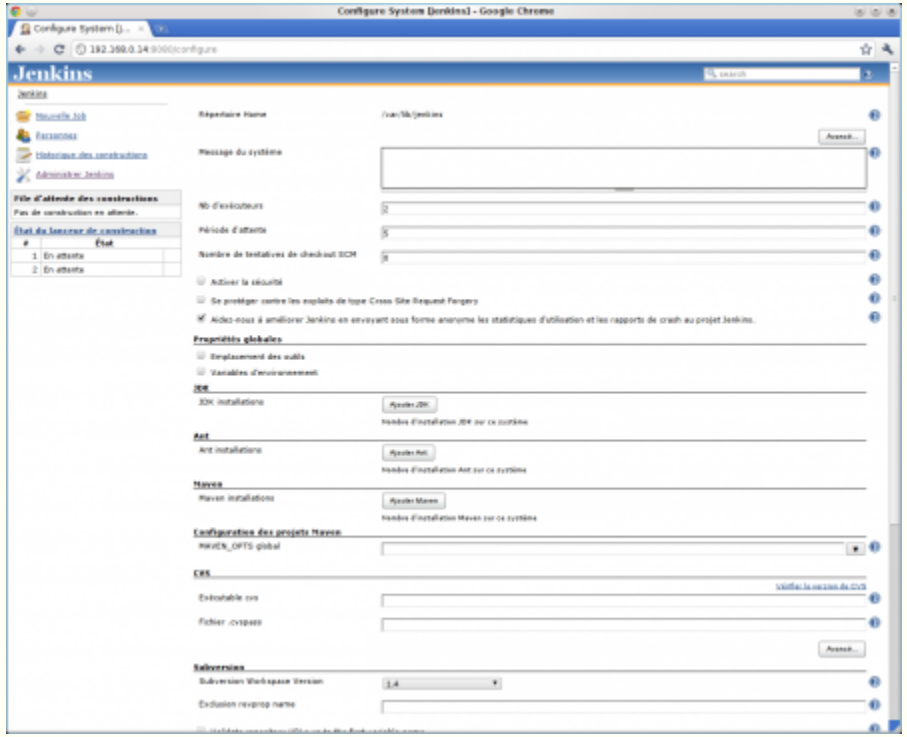
Interface d'administration

Interfa

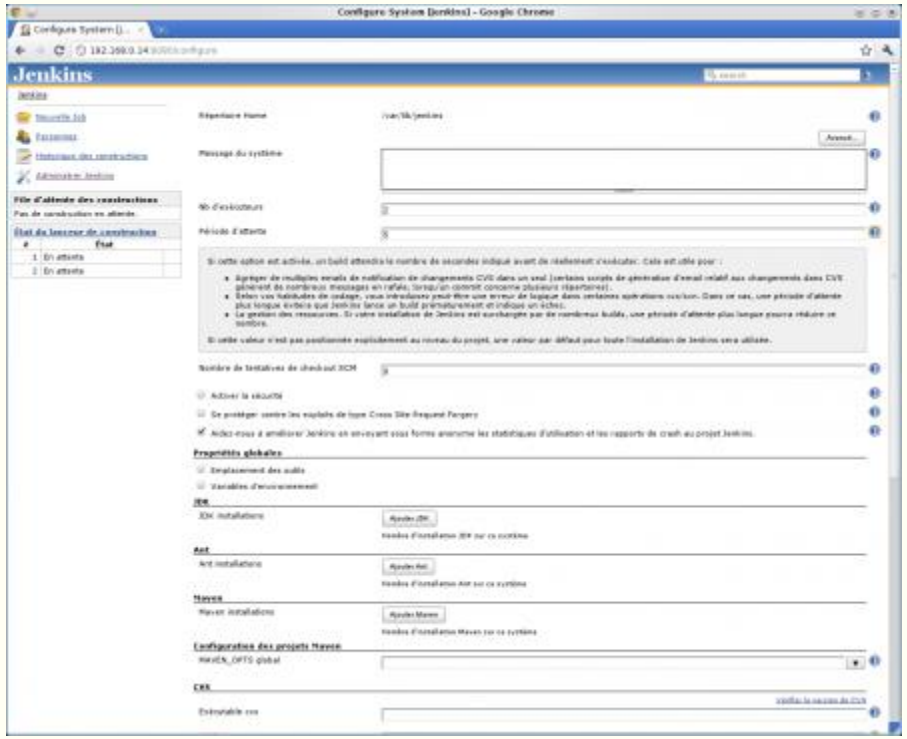


Administrer Jenkins

Configurer le système



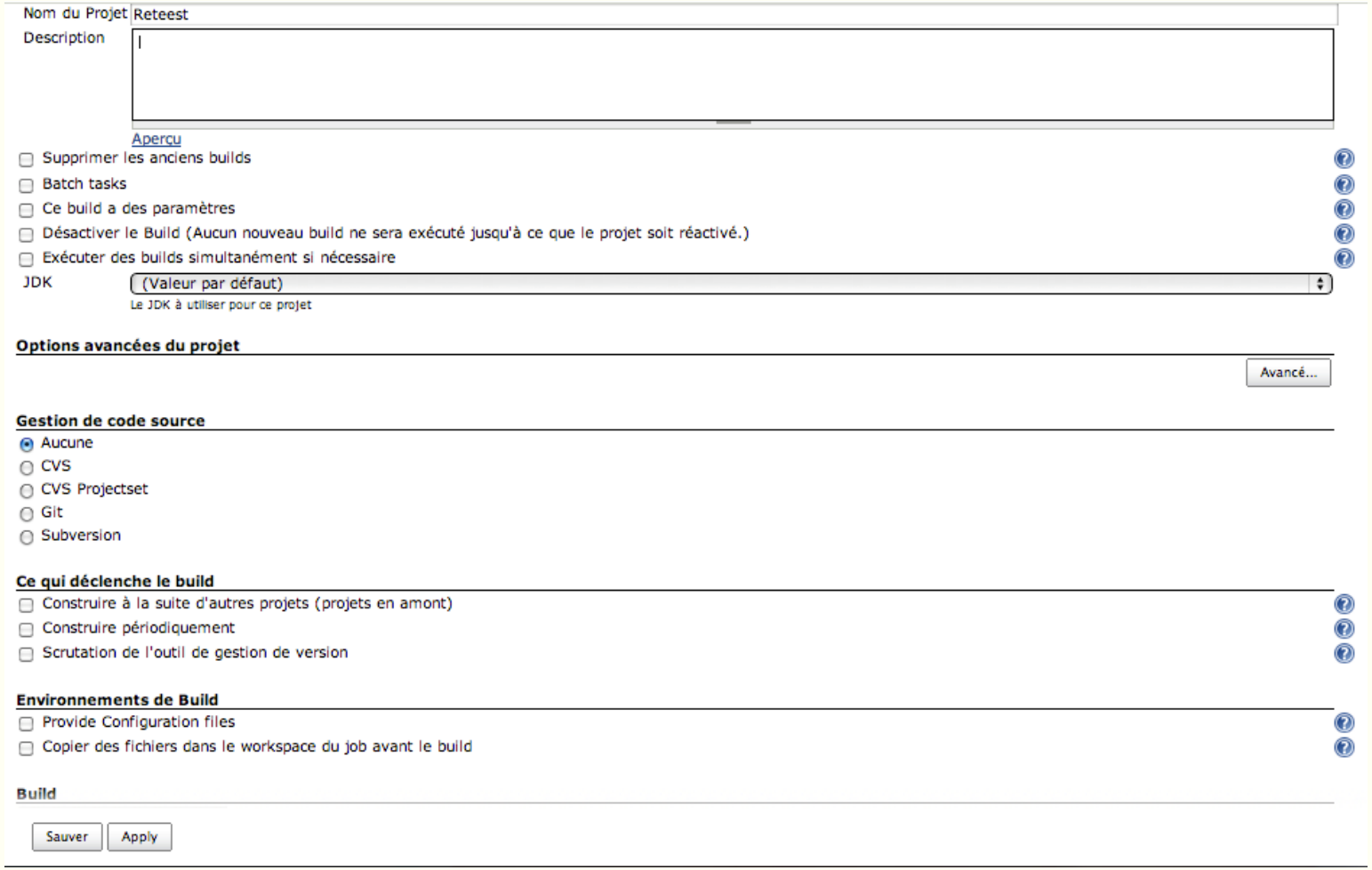
Aide



Configuration jobs

- Free-Style
- Maven
- Monitoring
- Multi-Project

Création d’un job



Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste

TP

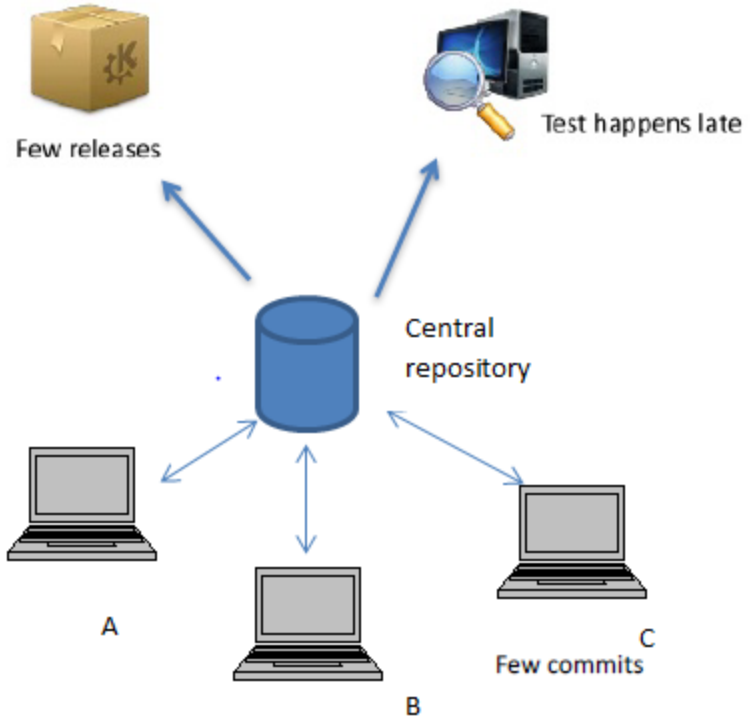
Installation de Jenkins

MODULE 3

Intégration continue avec Jenkins

Présentation

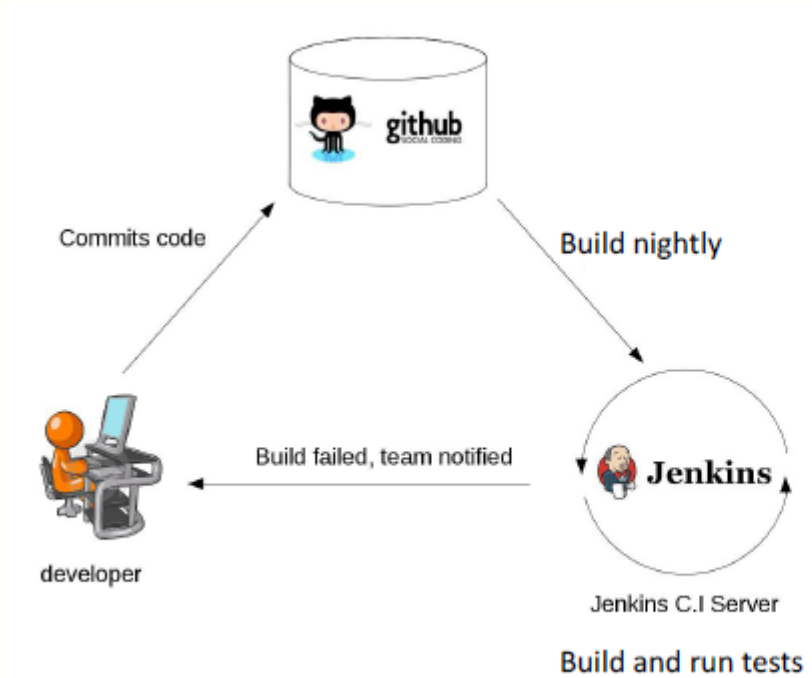
Adoption de l’intégration continue : Les étapes



Stage 1:

- No build servers.
- Developers do NOT commit on a regular basis.
- Changes are integrated and tested manually.
- Few releases.

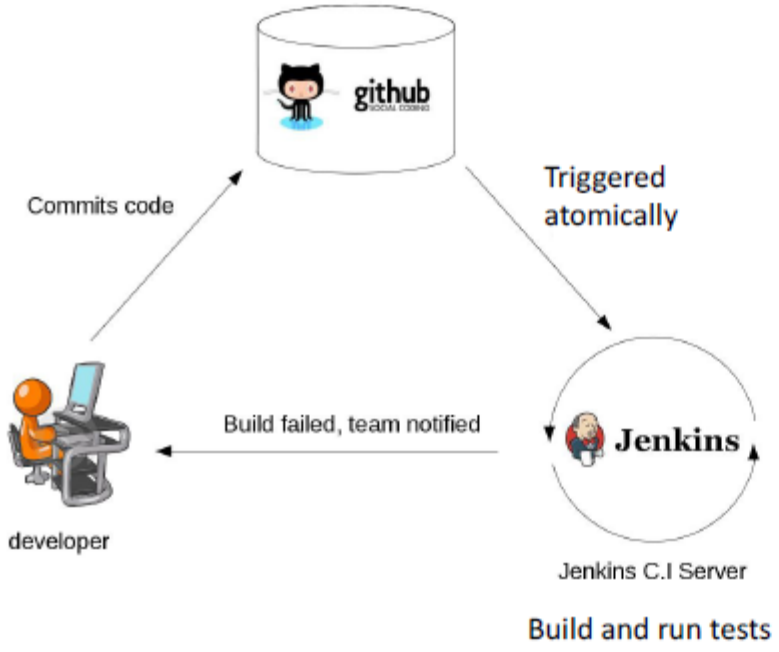
Adoption de l’intégration continue : Les étapes



Stage 2:

- Automated builds are scheduled on a regular basis.
- Build script compiles the application and runs a set of automated tests.
- Developers now commit their changes regularly.
- Build servers would alert the team members in case of build failure.

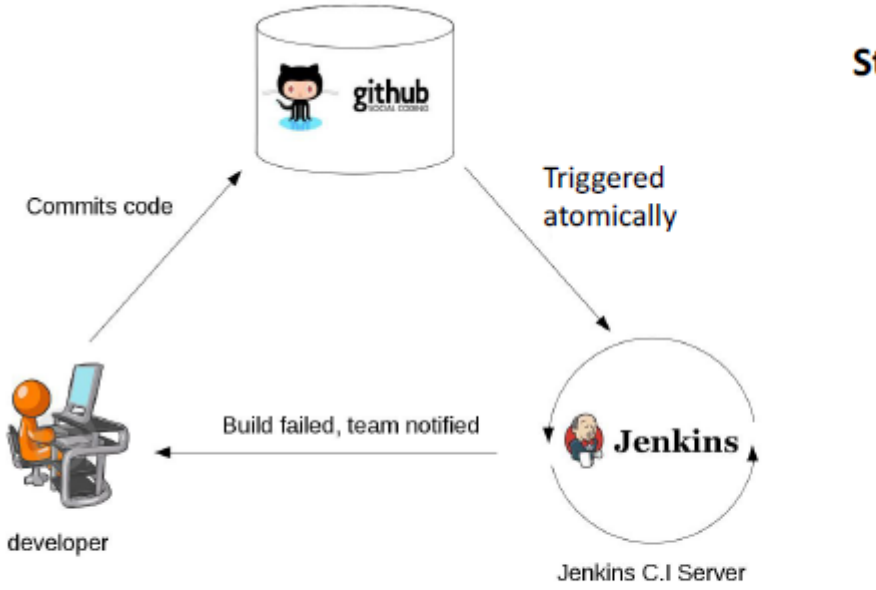
Adoption de l’intégration continue : Les étapes



Stage 3:

- A build is triggered whenever new code is committed to the central repository.
- Broken builds are usually treated as a high priority issue and are fixed quickly.

Adoption de l’intégration continue : Les étapes



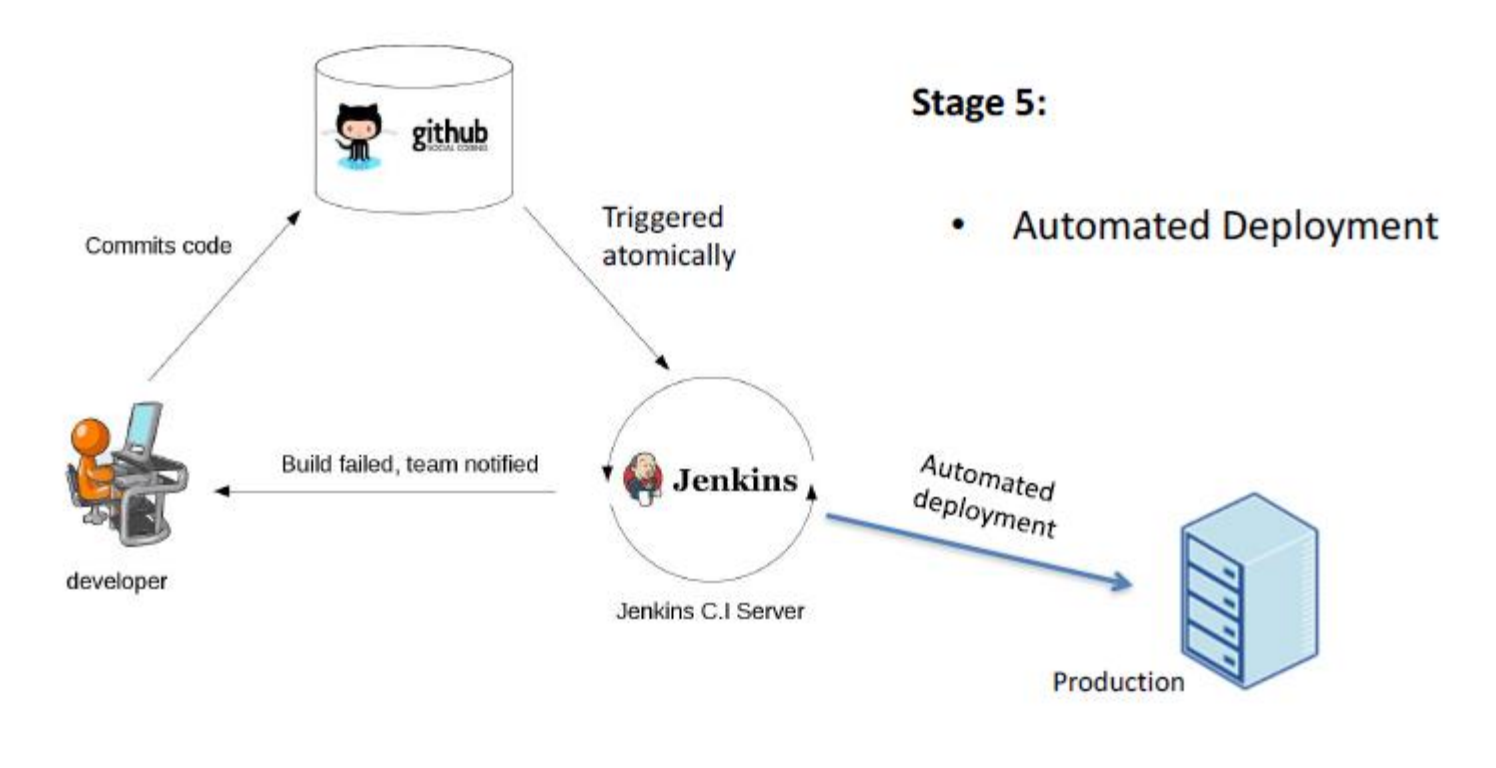
Stage 4:

- Automated code quality and code coverage metrics are now run along with unit tests to continuously evaluate the code quality.

  Is the code coverage increasing?

  Do we have fewer and fewer build failures?

Adoption de l’intégration continue : Les étapes



Que fait GIT ?

- Gestion de versions décentralisé

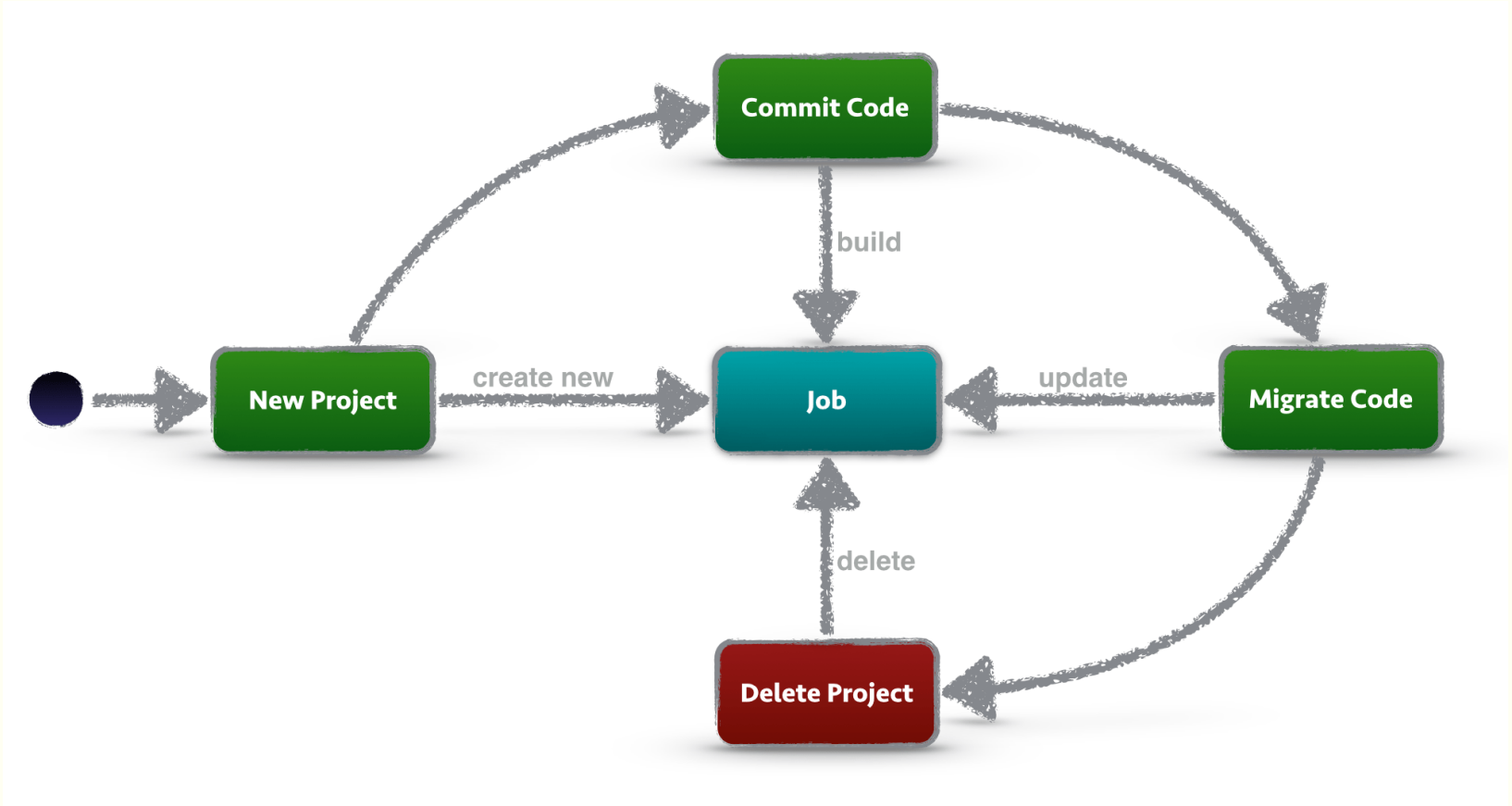
- Logiciel libre

OUTILS DE BUILDS JAVA

Job

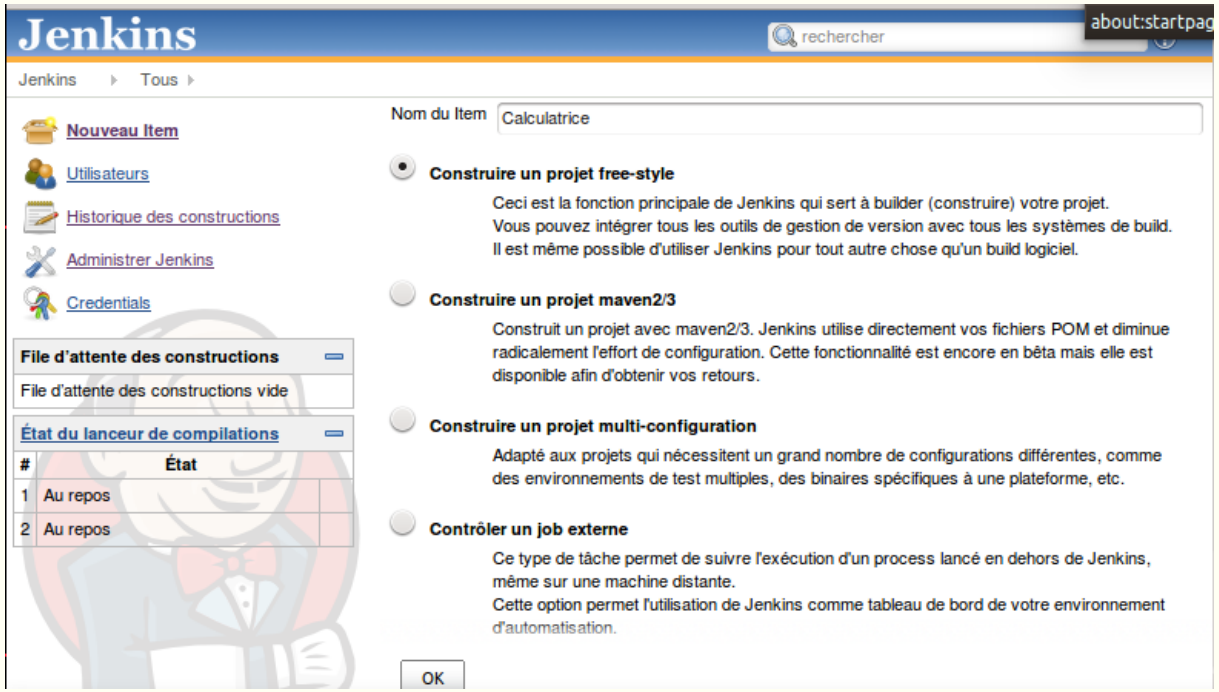
- Un **job** est l’équivalent d’un **projet**.
- Il va donc contenir un
  - **workspace** (le répertoire où seront stockés les fichiers sources),
  - **des tâches**
  - et **des builds**

Cycle de vie d’un Job

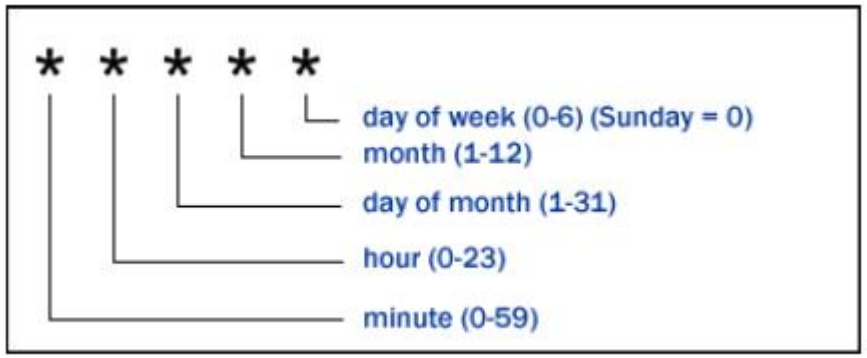


Création d’un Job

- Sur l’interface d’accueil, on peut ajouter un job en passant par le lien « **Nouveau Item** » disponible dans le menu de gauche.



Cron Syntax



To specify multiple values for one field:

*	all valid values
M-N	a range of values
A,B,Z	enumerates multiple values

Cron Syntax

- To schedule your build every 5 minutes, this will do the job : */5 * * * * OR H/5 * * * *
- To the job every 5min past every hour(5th Minute of every Hour) 5 * * * *
- To schedule your build every day at 8h00, this will do the job : 0 8 * * *
- To schedule your build for 4, 6, 8, and 10 o'clock PM every day – 0 16,18,20,22 * * *
- To schedule your build at 6:00PM and 1 AM every day – 0 1,18 * * *
- To schedule your build start daily at morning – 03 09 * * 1-5
- To schedule your build start daily at lunchtime – 00 12 * * 1-5

Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste

TP **Installation de Jenkins**

MODULE 4

Automatisation des tests

Présentation

Qualité du logiciel

- Le Manuel qualité transcrit, au niveau de l'entreprise, l'ensemble des méthodes, règles et procédures mises en œuvre pour développer un produit logiciel de **qualité**
  - **C'est une base documentaire**
  - **Qui s'enrichit au cours du temps**

- Le Plan qualité détermine le processus de développement **propre à un projet**, avec ses particularités et spécificités

- La Qualité est une **démarche continue** tout au long du développement du produit logiciel

- La Qualité se **définit**, se **mesure** et se **concrétise**

Qualité du logiciel

- **Validité** : aptitude d'un produit logiciel à réaliser exactement les tâches définies par sa spécification
- **Exploitabilité** : aptitude à être utilisé par les différentes parties prenantes
  - opérateurs,
  - utilisateurs,
  - maintenance...
- **Efficacité** : mesure de la bonne utilisation des ressources
- **Fiabilité** : aptitude d'un logiciel à fonctionner sans incident
- **Intégrité** : aptitude d'un composant à protéger ses différentes composantes contre des accès ou des modifications non autorisées

Qualité du logiciel

- **Extensibilité** (*Maintenabilité*) : facilité d'adaptation d'un logiciel aux changements de spécifications
- **Exactitude** : aptitude d'un logiciel à produire des résultats corrects
- **Robustesse** : aptitude d'un logiciel à fonctionner dans des conditions anormales
- **Portabilité** : facilité avec laquelle un constituant logiciel peut être adapté à différents environnements matériels et logiciels

Qualité du logiciel

- **Réutilisabilité** : aptitude d'un logiciel à être réutilisé pour de nouvelles applications
- **Lisibilité** : faculté avec laquelle un composant est lu et donc compris
- **Vérifiabilité** : facilité de préparation des procédures de test pour le constituant considéré
- **Compatibilité** : aptitude des logiciels à pouvoir être combinés les uns avec les autres

Qualité du logiciel

- La qualité est difficile à obtenir dans les applications informatiques Les objectifs de qualité sont peu souvent définis
- Le seul moyen de s'en assurer est souvent limité aux tests Les tests sont peu productifs
- Beaucoup d'erreurs sont découvertes en exploitation
- Les tests coûtent cher
- La non-qualité des applications coûte encore plus cher
- Pourra-t-on se le permettre encore longtemps ? (car de plus en plus d'activités humaines dépendent directement de la qualité des applications informatiques)

Qualité du logiciel

- Le 22 juillet 1962, une fusée Atlas-Agena lançant un satellite Mariner I fut détruite par erreur

- **Cause :**
  - Erreur minime dans le logiciel de guidage, amenant l'ordinateur à croire que la fusée se comportait mal, ce qui n'était pas le cas

- **Coût :**
  - 18,5 millions de dollars *Source : Les avatars du logiciel, Addison-Wesley*

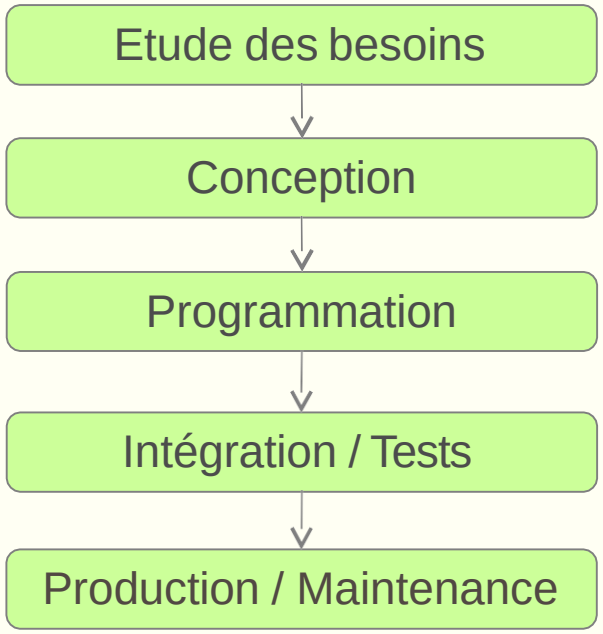
Qualité du logiciel

- En juillet 1991, des séries de coupures affectèrent les abonnés du téléphone de Los-Angeles, Washington, etc.

- **Cause :**
  - Le programme de commutation avait été modifié. Ce programme comporte plusieurs millions de lignes de code, et un cycle de tests représente 13 semaines.
  - La modification ne touchant que **trois lignes de code**, on avait estimé inutile de refaire des tests complets !

Qualité du logiciel

> Exemple de cycle de vie



- Les tests constituent une phase isolée, indépendante, semblant située **après** les autres phases.
- Est-ce la meilleure solution ?
- N'est-ce pas une des causes de la mauvaise qualité du logiciel ?

Les tests devraient être « horizontaux »

Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste

MODULE 5

Automatisation du déploiement

Présentation

Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste

MODULE 6

Le plugin Pipeline

Présentation

Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste

MODULE 7

Architecture Maître Esclave

Présentation

Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste

MODULE 8

Administration d'un serveur Jenkins

Présentation

Disposition de titre et de contenu avec liste

Question ?

Disposition de titre et de contenu avec liste