Industrialisation et Continuous Delivery

GÉRER SES SOURCES ET LES LIER À SON INTÉGRATION CONTINUE
AVEC L'UN DES OUTILS LES PLUS POPULAIRES

The world is how we shape it*

* Le monde est tel que nous le façonnons.

sopra steria

Vos intervenants

Benoit POIRIER
Architecte Solution
Sopra Steria
benoit.poirier@soprasteria.com

Fabrice ROULAND
Expert Technique
Sopra Steria
fabrice.rouland@soprasteria.com

Les métiers de l'IT

- Développeur (Back, Front, …)
- Expert Technique / Responsable Technique
- Chef de projet
- Directeur de projet
- Architecte (Solution, Infrastructure, Urbaniste)
- Business Analyste
- Data Scientist
- Web Designer
- Ergonome
- …

Ce que nous allons aborder

Pourquoi l'industrialisation ?

Le DevOps et le CI/CD

En pratique

GitLab en tant que Git server

GitFlow: savoir gérer ses sources efficacement

S'intégrer au CI MANCED

01

Pourquoi l'industrialisation ?

SE RECENTRER SUR L'ESSENTIEL : LA VALEUR MÉTIER

Certains projets IT…

Les autres…

Volumétrie métier

- > 15 000 utilisateurs
- > 15 millions de clients
- > 200 000 demandes clients traitées par jour
- >300 millions de communications facturées par jour
- > 1300 cas d'usages

Volumétrie projet

- Volume d'activités > 100 000 jh
- > 200 personnes en pic réparties sur 4 sites
- 350 documents de conception 3700 diagrammes > 100 000 pages
- 1,5 millions de lignes de codes
- *Plus qu'un projet, une véritable aventure !*

Volumétrie SI

- 85 applications interfacées
- > 250 To de données
- 450 flux
- 3 millions d'appels de Web Services par jour
- 40 serveurs (VM) 600 cœurs de CPU

DES CHIFFRES…

Gestion des exigences
Spécifications fonctionnelles
Architecture logicielle
Architecture technique
Spécification techniques
Code
Intégration continue
Installation/suivi des environnements de développement
Tests automatisés
Tests de performance
Sécurité, test d'intrusion
Mise en production
Gestion de la production

Business Analyst
Architecte (solution, urba, infra)
Expert technique
Développeurs
Opérationnels

Gestion des exigences
Spécifications fonctionnelles
Architecture logicielle
Architecture technique
Spécification techniques
Code
Intégration continue
Installation/suivi des environnements de développement
Tests automatisés
Tests de performance
Sécurité, test d'intrusion
Mise en production
Gestion de la production

Equipe France

Business Analyst
Architecte (solution, urba, infra)
Expert technique
Développeurs
Opérationnels

Equipe Inde

Business Analyst
Architecte (solut
Expert technique
Développeurs
Opérationnels

Gestion des exigences
Spécifications fonctionnelles
Architecture logicielle
Architecture technique
Spécification techniques
Code
Intégration continue
Installation/suivi des environnements de développement
Tests automatisés
Tests de performance
Sécurité, test d'intrusion
Mise en production
Gestion de la production

Equipe Canada

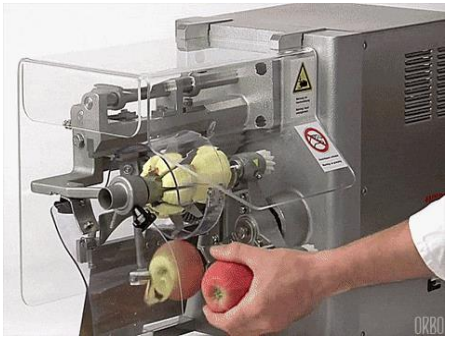
Business Analyst
Architecte (solution, urba, infra)
Expert technique
Développeurs
Opérationnels

SOLUTION REFLEXE : CRÉER DES PROCÉDURES SÉQUENTIELLES

L'ÉQUIPE EST SOUS L'EAU

Industrialisation

- Pour gagner en productivité
- Pour rester à la pointe des techniques de production



La solution : l'industrialisation

Expérience industrielle

Expérience utilisateur

sopra steria

Intégration continue

- **Garantir la qualité des livrables en intégrant et testant les changements dans le code *en continu***
  - *« … détecter les problèmes d'intégration au plus tôt lors du développement »*
  - *« … afin d'améliorer la qualité du code et du produit final »*
- **Gagner du temps en :**
  - Automatisant les taches répétitives,
  - Automatisant les taches complexes,

02

Le DevOps et le CI/CD

Le DevOps et le CI/CD

Dev et Ops : Une séparation marquée depuis des années

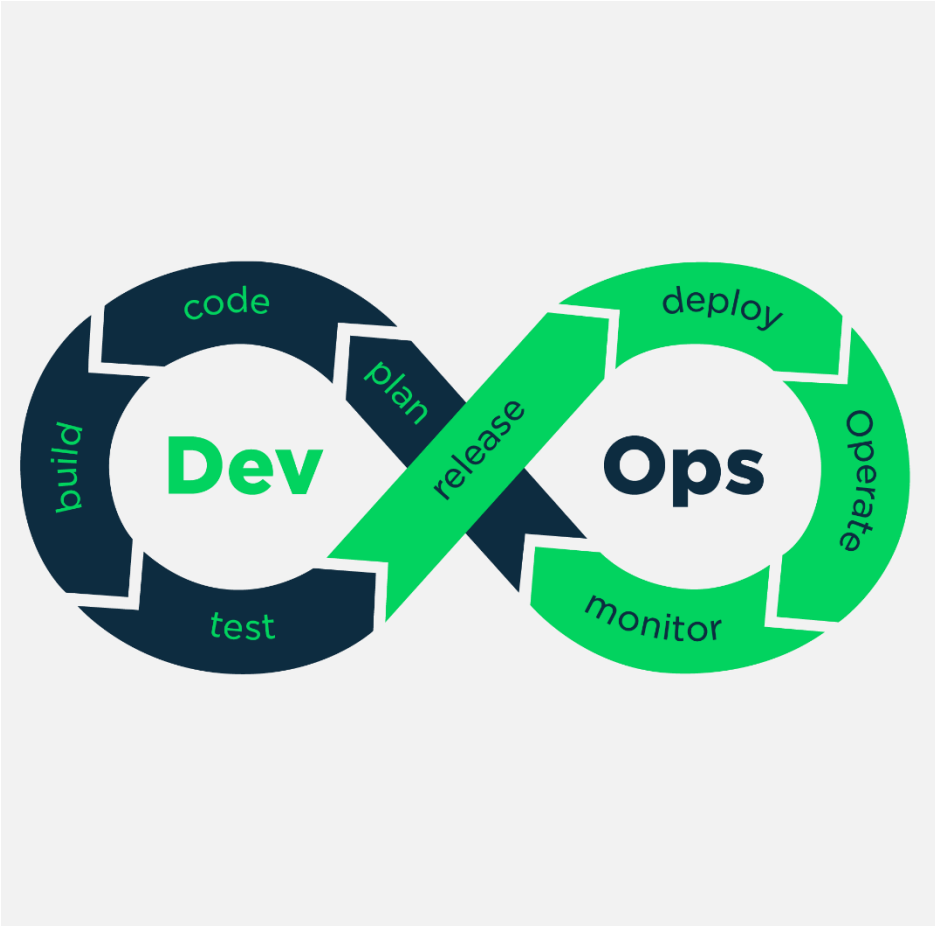
- **Dev**
  - Développement des applications
  - Objectifs : produire rapidement et à faibles coûts
  - Moyens : utilisation de Frameworks, de systèmes à paramétrer, …
  - Reproche aux équipes Ops leurs retards
- **Ops (Exploitation)**
  - Administration des infrastructures / systèmes
  - Objectif : Garantir la qualité et la stabilité du système
  - Moyens : contrôler sévèrement la qualité des changements
  - Reproche aux équipes Devs les incidents

sopra steria

Le DevOps et le CI/CD

Réunir 2 mondes grâce aux pratiques DevOps

- **Culture**
  - Privilégier les échanges
  - Transparence totale
  - Itérer d'avantage en réduisant les cycles
- **Processus**
  - Agilité
  - La valeur métier comme principal objectif
  - Création de petits éléments (micro-services, …)
- **Plateformes et outils**
  - Utilisation d'outils de sécurité, de travail en collaboration et d'analyse de données
  - Automatisation des builds, des tests, des contrôles de sécurité et des déploiements



Le DevOps et le CI/CD

Une culture à repenser

- **Documenter les pratiques dans un wiki**
- **Privilégier la proximité des équipes et les échanges en direct**
- **Fonctionner en itérations courtes de 2 à 3 semaines**
- **Ne plus voir la livraison comme une échéance critique mais comme une tâche parmi les autres**
- **Inciter à l'expérimentation pour ne plus avoir peur de l'échec**

- **Partage**
- **Transparence**
- **Contact**
- **Cycles courts**
- **Expérimentations**

Le DevOps et le CI/CD

Le DevOps est né des pratiques agiles

Conférence « Agile » de Toronto en 2008

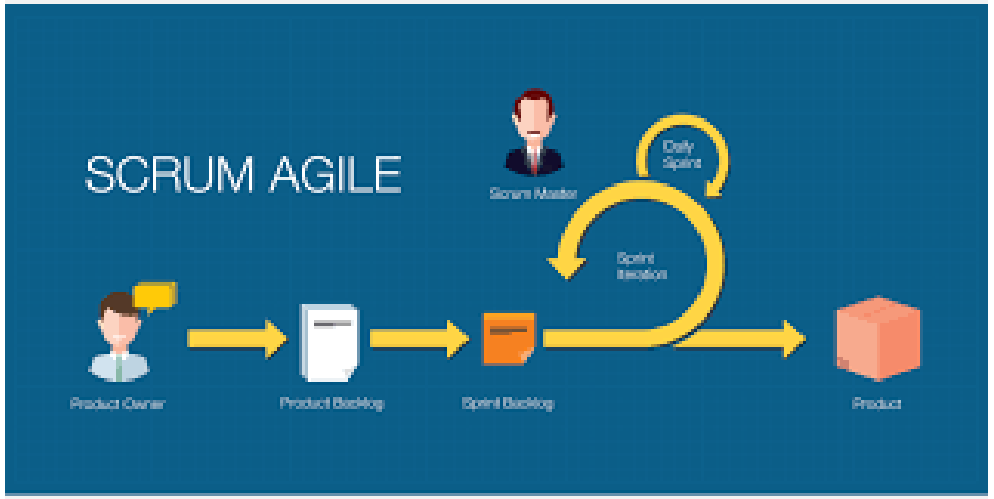
La valeur métier au cœur de la préoccupation de l'équipe

Fonctionnement par itérations

Sprints courts

Développer l'autonomie de l'équipe

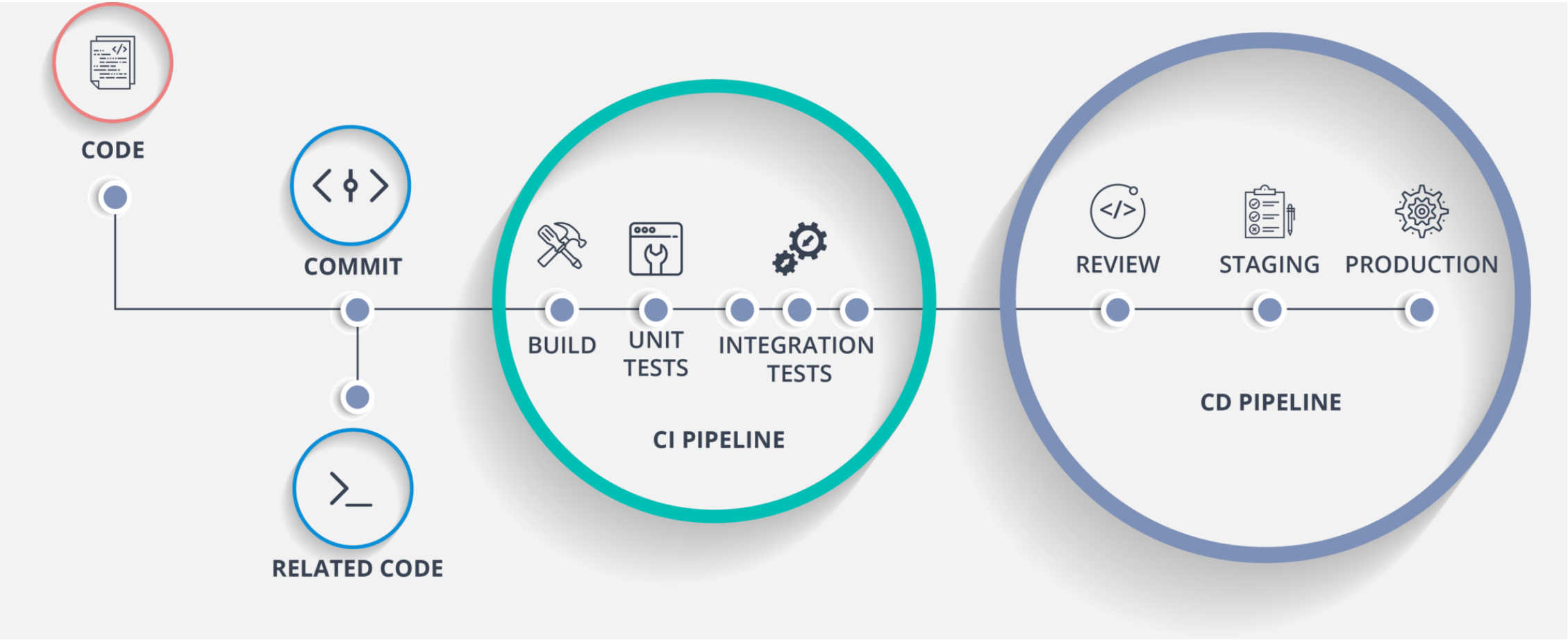
Des cérémonies avec un timing maitrisé et un but concret



sopra steria

Le DevOps et le CI/CD

Focus sur l'intégration et le déploiement continu



sopra steria

Le DevOps et le CI/CD

Focus sur l'intégration et le déploiement continu

- **Le but du CI/CD est d'automatiser toutes les tâches de build, test, check de sécurité, analyse statique, déploiement, … à chaque modification du code**
- **Depuis une simple modification sur une branche d'un gestionnaire de sources, nous pouvons déclencher cette suite d'actions : le pipeline**
- **Il suffit d'avoir les outils, de dire à l'utilitaire quelle(s) branche(s) scruter dans notre gestionnaire de sources et ce dernier s'occuper de lancer les pipelines**

01

Gestionnaire de sources

02

Outils de build, de test, de sécurité, de déploiement, …

03

Mise en place de pipelines

Le DevOps et le CI/CD

Des outils de CI/CD

sopra steria

03

La pratique du build dans le continuous delivery

VOUS AVEZ DIT JENKINS ?

Quel contexte ?

- **Pour tous les projets…**
  - Avec au moins 1 développeur
  - Au moins 1 version en cours de développement

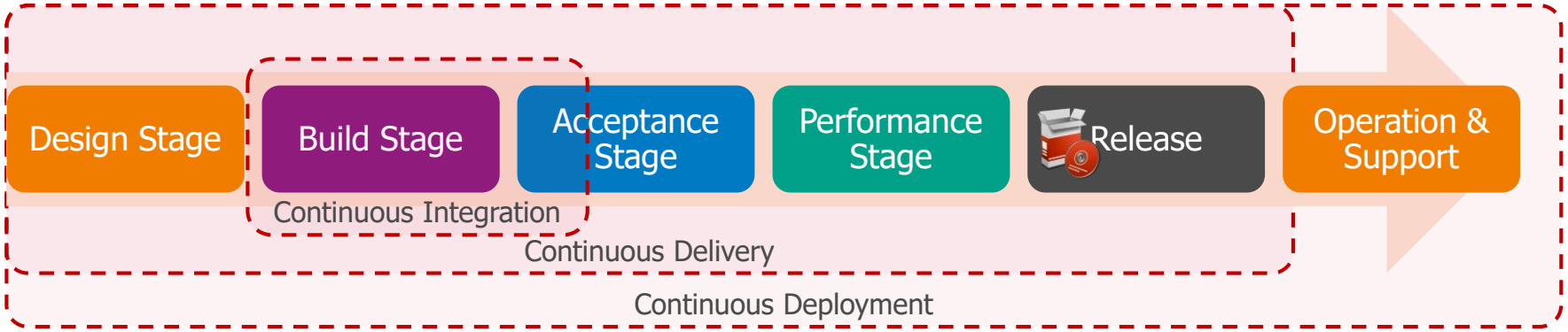
- **Une bonne Intégration Continue s'appuie sur :**
  - Une gestion de configuration (exigences, composants logiciel, …)
  - Un gestionnaire de build
  - Un gestionnaire de dépendances
  - Une **plateforme** d'intégration continue

  - **Et…**

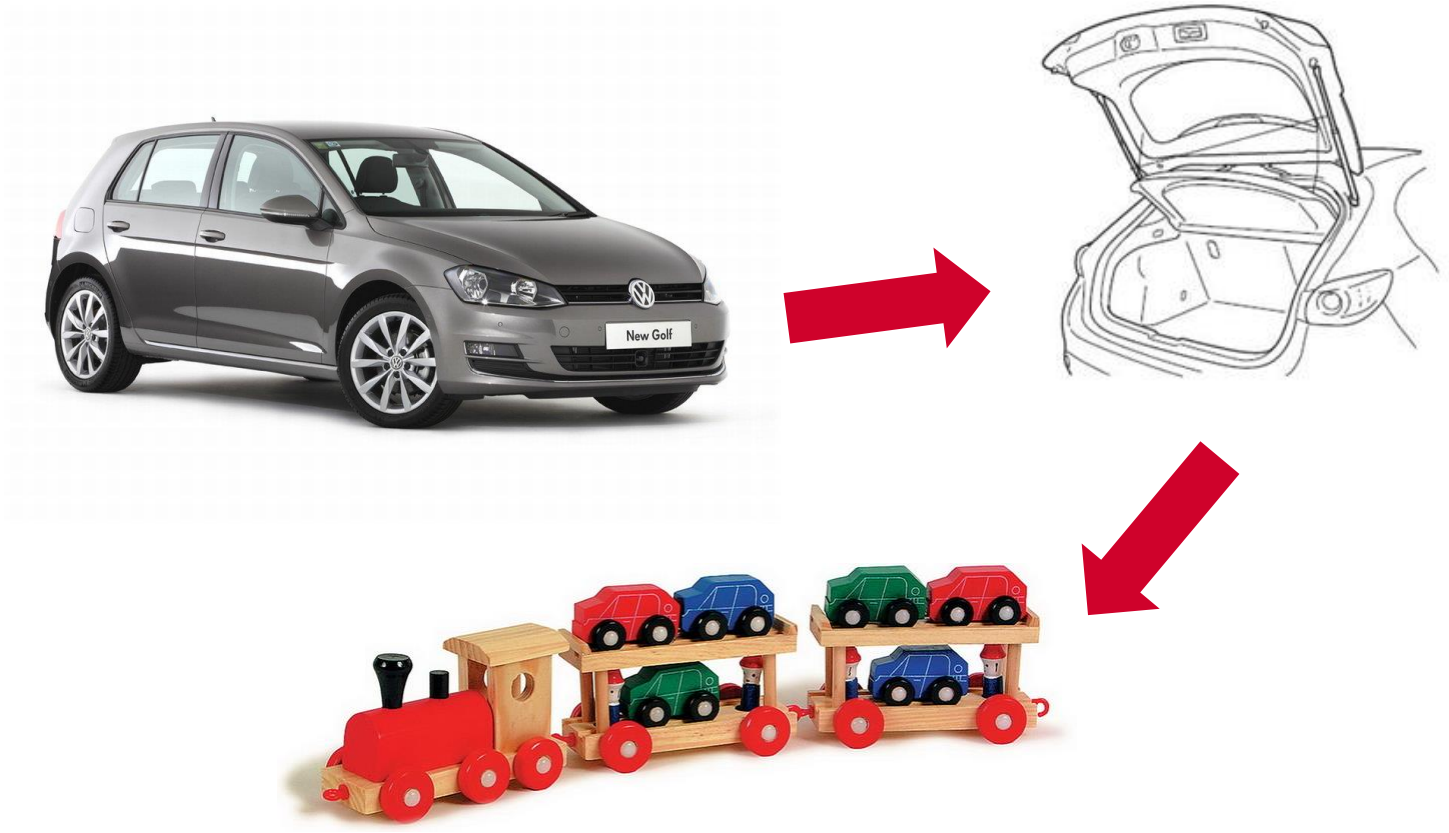
maven

Continuous Delivery

- **Extension naturelle de l'intégration continue**
- **Développer et livrer pour la production rapidement, avec un haut de niveau de qualité**



Design | Build | Acceptance | Performance | Release | OPS



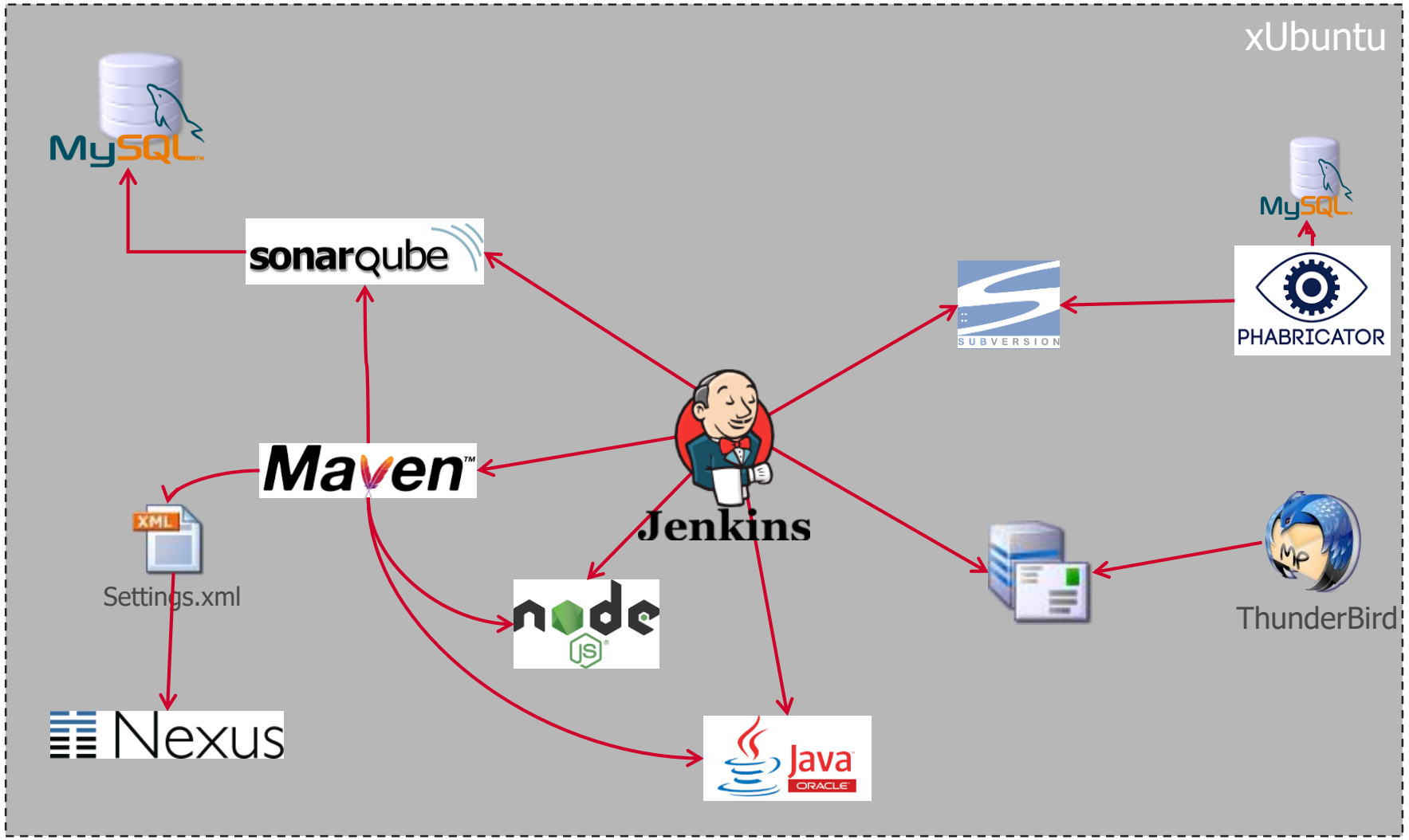
GESTION DES EXIGENCES : TRANSPARENCE ET INSPECTION

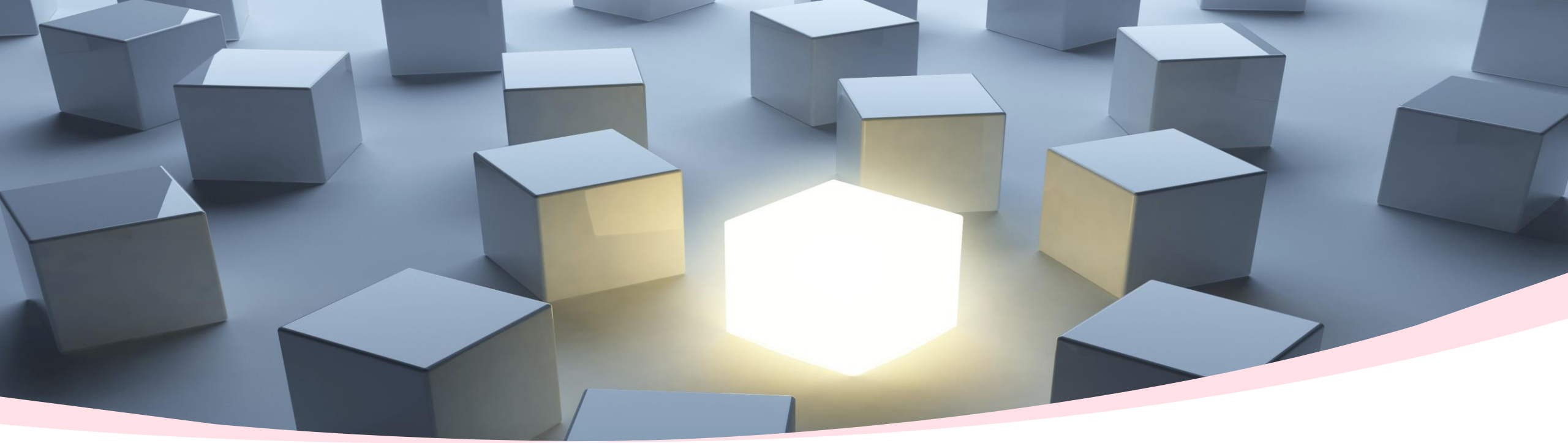
Build dans le continuous delivery
LA PRISE EN COMPTE D'UN COMMIT

- **L'aventure commence avec le commit.**

- **Les 5 grandes pratiques dans le package « Build » du Continuous Delivery**
  - Intégration continue
  - Exécution des tests de développement
  - Mesure de la qualité logicielle
  - Revue de code
  - Packaging et Promotion

Build dans le continuous delivery
UN EXEMPLE D'ENVIRONNEMENT





INTÉGRATION CONTINUE

Intégration continue
LE QUIZZ

L'intégration continue

- ✗ N'est utile que pour les grands projets
- ✗ Ne doit être exécutée qu'une fois par jour
- ✓ Se lance à intervalles réguliers pour intégrer les nouvelles contributions
- ✗ Sa mise en place est compliquée

Intégration continue
LE QUIZZ

L'intégration continue:

- ✔ Est utilisée par les développeurs
- ✘ Est définie et administré par l'intégrateur
- ✘ Son usage est interdit aux fonctionnels et aux PM
- ✔ Est utilisable par tous de façon simple

Intégration continue
LE QUIZZ

Les jobs de l'intégration continue:

- ✗ Ne sont lancés que manuellement
- ✗ Sont indépendants les uns des autres
- ✗ Ne font que du build
- ✓ Sont chainés en suivant les processus définis sur le projet.

Intégration continue
POURQUOI AVOIR UNE INTÉGRATION CONTINUE ?

- **L'intégration, c'est la mise en commun de toutes les contributions pour découvrir et résoudre au plus tôt les problèmes d'intégration.**

- **Les gains d'une intégration plus fréquente sont :**
  - La détection rapide de problème
  - La localisation de l'origine simplifiée
  - La correction rapide

- **L'intégration devient systématique et continue.**

Intégration continue
COMMENT FAIRE UNE INTÉGRATION CONTINUE ?

- **L'intégration continue est d'abord :**
  - Une **bonne organisation de projet**
  - Une discipline d'équipe
- **Pour une bonne intégration continue, il faut**
  - Une **gestion de configuration complète**
  - Une plateforme d'intégration continue
  - Un **environnement dédié**
  - Une alimentation uniquement par la gestion de configuration
- **Elle comprend au strict minimum**
  - une phase de construction de l'ensemble de la solution
  - Une exécution une fois par jour

Intégration continue
COMMENT FAIRE UNE INTÉGRATION CONTINUE ?

- **Le Continuous Delivery introduit le concept de pipeline**
  - Définition des grandes activités (Build, Analyse, Packaging, Intégration, Qualification)
  - Enchainement automatique des différentes activités
  - Support aux processus et à l'organisation du projet
  - Suivi des pipelines via InTools et les feux tricolores

- **Chaque commit déclenche la fabrication d'une version potentielle de l'application.**

Intégration continue
COMMENT FAIRE UNE INTÉGRATION CONTINUE ?

- **Automatisation poussée au maximum**
  - Actions automatiques (construction, analyse, déploiement, package, tests)
  - Elle prend en compte automatiquement à intervalles réguliers toutes les nouvelles contributions
  - Indépendant des acteurs (disponibilité et compétences)
  - Répétable pour permettre l'amélioration
- **Gage de qualité :**
  - Accessible à tous
  - Simplicité d'utilisation
  - Obligatoire dans son utilisation
  - Partagé et documenté : une étape remplit une mission définie
  - Normalisation des étapes pour toutes les applications de la solution.
- **Adaptable au projet :**
  - Cette représentation est adaptée à tout type de Delivery Process : Agile, itératif et incrémental ou cycle en V

INTEGRATION CONTINUE
BONNES PRATIQUES : GARANTIR LA TRAÇABILITÉ

- **C'est-à-dire qu'on doit savoir avec certitude**
  - Ce que l'on construit
  - Ce que l'on teste
  - Ce que l'on analyse
  - Ce que l'on package
  - Ce que l'on tague
  - Ce que l'on déploie
  - Ce que l'on qualifie
  - Ce que l'on livre

TESTS DE DÉVELOPPEMENT

Tests de développement
LE QUIZZ

Les tests de développement, à la différence des tests d'intégration :

- ✓ Permettent de valider l'algorithme
- ✗ Permettent de valider le déploiement
- ✗ N'ont pas besoin de jeux de données
- ✗ Doivent être nécessairement bouchonnés

Test de développement
LE QUIZZ

Les tests de développement sont :

- ✓ Des tests unitaires exécutables par le développeur
- ✓ Des tests unitaires exécutables par l'intégration continue
- ✗ Des tests obligatoirement développés en JUnit.
- ✓ Autoporteurs en terme de donnée
- ✗ Dépendant d'un déploiement de la solution

Tests de développement
LE QUIZZ

Pour le BDD

- ✔ On développe du code technique.
- ✔ On écrit le test en bon français.
- ✔ On peut le rejouer à l'infini.
- ✘ Il faut d'abord créer le jeu de données avant chaque lancement.

Tests de développement
POURQUOI AVOIR DES TESTS DE DÉVELOPPEMENT AUTOMATISÉS ?

- **L'objectif est de trouver les défauts d'implémentation**
  - Vérifier que le **comportement** des composants développés est celui attendu
  - Vérifier la **robustesse** des composants développés face à des conditions aux limites
- **Un test de développement est un test unitaire ou d'intégration qui nécessite ni de déploiement de l'application ni de donnée extérieure**
- **Les tests de développement automatisés :**
  - Aident au développement
  - Documentent les composants
  - Contrôlent des éventuelles régressions
  - Sécurisent le développement
  - Bonifient l'intégration continue

Tests de développement
COMMENT FAIRE DES TESTS DE DÉVELOPPEMENT ?

- **Il faut tester au plus juste**
  - Il faut **prendre le temps de tester,** mais sans y passer trop de temps
  - Il faut donc être **pertinent** dans ce qu'on teste et comment on le teste
- **La validation du composant et des tests de développement associés se fait en même temps.**
- **Le projet doit définir :**
  - Couverture de tests attendue
  - Analyser des risques
  - Identifier les incontournables
  - Framework de test unitaire
- **Il ne faut pas tout tester :**
  - Il faut faire une analyse de la valeur et du ROI

Tests de développement
COMMENT FAIRE DES TESTS DE DÉVELOPPEMENT ?

- **Principaux framework de tests**
  - JUnit, TestNG, nUnit
- **Mesure de la couverture**
  - Cobertura, Jacoco
  - Rapport intégrable dans le tableau de bord SonarQube.
- **Dans un contexte TMA,**
  - Ne pas chercher à créer des tests de développement sur tout le périmètre
  - Ne pas se focaliser sur les tests « unitaires »
  - Travailler de façon opportuniste
  - Ne pas hésiter supprimer des tests caduques
  - La présence de tests de développement est contrôlée lors de la revue de code.

Tests automatisés

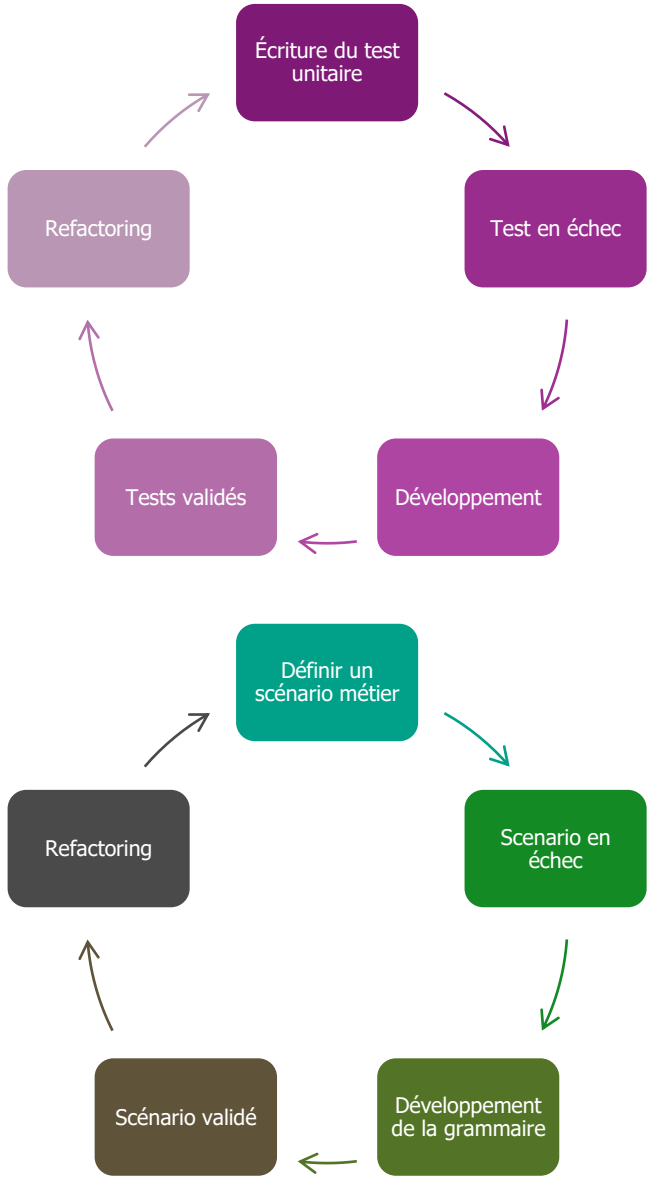
DIFFÉRENTS NIVEAUX DE TESTS AUTOMATISÉS

- **Les tests de développement**
  - Tests unitaires ou d'intégration sans déploiement de l'application
  - Non régression et aide au développement.
- **Tests d'intégration technique**
  - Validation d'un déploiement et d'un packaging
- **Tests fonctionnels**
  - Complémentaires aux tests manuels
  - Création automatisée de jeux de données
- **En synthèse :**
  - Un investissement qui sera vite rentabilisé, capitalisable
  - Répétables, rejouables à l'infini n'importe quand
  - Une mesure objective et rapide de la solution
  - Une stratégie globale de tests

Tests automatisés

POUR ALLER PLUS LOIN : DU TDD AU BDD

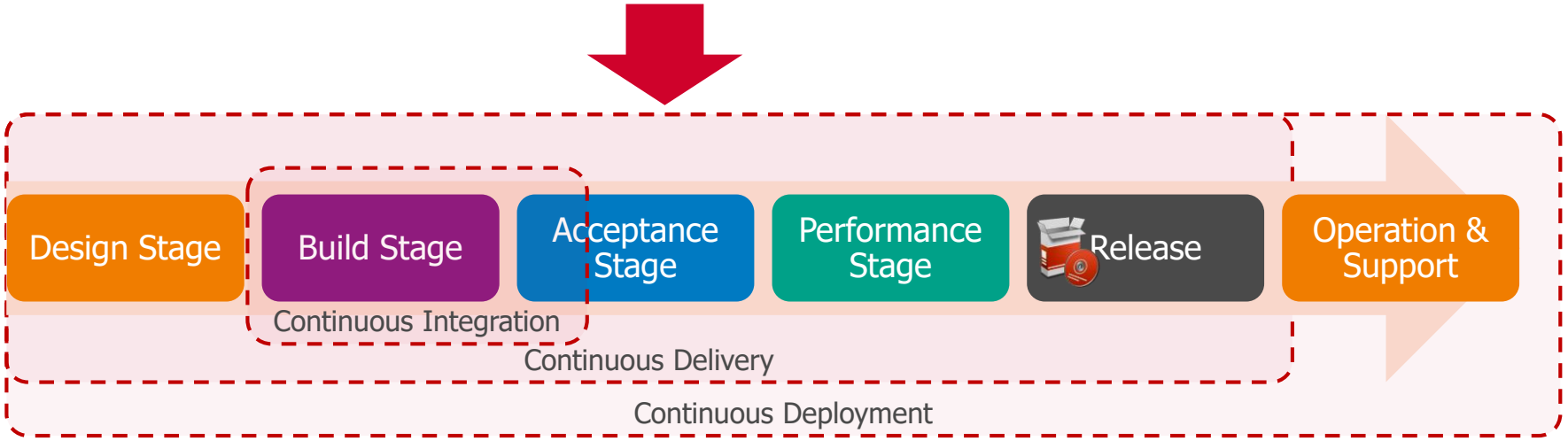
- **Du Test Driven Development**
  - Compréhension du besoin
  - Écriture d'un test unitaire
  - Lancement du test qui est en échec.
  - Implémentation de la fonctionnalité pour que le test passe
  - Réalisé par le développeur
- **Au Behavior Driven Development**
  - Concevoir les tests à partir d'un comportement ou d'une fonctionnalité.
  - Écrire en français le test par rapport à un besoin (et non la solution) à partir d'une grammaire définie
  - Implémentation de la grammaire
  - **Implique la collaboration entre les fonctionnels et les techniciens**



QUALIMÉTRIE
Analyse statique du code

Continuous Delivery

- **Extension naturelle de l'intégration continue**
- **Développer et livrer pour la production rapidement, avec un haut de niveau de qualité**



Qualimétrie - Analyse de code
LE QUIZZ

L'analyse de code révèle:

- ✔ Le manque des Tests Unitaires
- ✘ Le manque de Tests d'Intégration
- ✔ Les exceptions mal catchées
- ✔ Les failles de sécurité
- ✔ Le non respect des standards

Qualimétrie - Analyse de code
LE QUIZZ

Une bonne analyse de code :

- ✗ Ne doit pas relever de violation
- ✗ Doit être faite à chaque commit
- ✗ Doit être faite à chaque livraison
- ✓ Est une mesure objective de la dette technique

Qualimétrie - Analyse de code
NOTION DE DETTE TECHNIQUE

- **Définition de la dette technique**

  Le terme de dette technique provient initialement de la logique d'intérêts que l'on retrouve dans le calcul d'une dette dite financière : il s'agit de son application dans la vie d'un projet de développement logiciel.

  L'analogie illustre la notion d'intérêts : s'ils ne sont pas remboursés rapidement, le coût de la dette augmente jusqu'à un point où il n'est plus possible de la rembourser intégralement, et où tout paiement ne sert qu'à rembourser ces intérêts

- **La dette technique est nécessairement relative.**
  - Elle est liée à un choix d'architecture et de règles de conception.
  - Il n'y a pas de vérité absolue.

Qualimétrie - Analyse de code
POURQUOI AVOIR UNE ANALYSE DE CODE ?

- **Pourquoi une analyse de code ?**
  - Pour garantir la qualité par rapport à un niveau attendu
  - Pour estimer la dette technique de la solution
  - Pour respecter nos engagements contractuels (exigences spécifiques)
  - Pour faire grandir nos collaborateurs
- **La dette technique peut être**
  - Consciente :
    - On a pris un raccourci pour respecter un délai.
    - On a ajouté des fonctionnalités sans faire le refactoring nécessaire
  - Inconsciente :
    - Non respect de règles
- **La qualité logicielle aura un lien direct sur la maintenabilité :**
  - Stabilité de la solution
  - Coûts des évolutions ou des corrections

Qualimétrie - Analyse de code
COMMENT FAIRE DES ANALYSES DE CODE ?

- **L'analyse de code peut être très large et variée.**
- **L'outil central reste SonarQube qui permet de définir ses propres règles et d'agréger des analyses faites en amont.**
- **D'autres outils d'analyse peuvent être utilisés :**
  - Analyse statique de code avec des plugin Maven (pmd, checkstyle, findbugs)
  - Analyse de couverture de test (jacoco, cobertura)
  - Analyse de sécurité (Owasp dependencies check)
  - Analyse plus complète avec l'outil CAST
- **Dans tous les cas :**
  - Il faut une vision claire des règles contrôlées et des normes associées.
  - Il faut une vision claire des résultats
  - Une notification directe du développeur qui a intégré la violation est très utile
  - Il est important de gérer la dette technique ainsi mise en évidence.

Qualimétrie - Analyse de code
BONNES PRATIQUES

- **Elle adresse :**
  - Le respect des normes de développement
  - La mesure de la qualité logicielle par rapport à des critères de contrôle
- **Attention faux positifs et la configuration de vos plugins**
  - Certaines violations peuvent être admissibles sur le projet.
  - La configuration des postes doit être commune et en phase avec la configuration de l'analyse de l'intégration continue.
- **La mesure de la dette technique ne sert à rien si elle n'est pas gérée avec des plans d'actions adaptés.**
- **Le niveau de qualité ne doit pas être un frein à la mise en place d'une analyse de code :**
  - « J'ai plus de 3 000 violations ; cela ne sert à rien »

REVUE DE CODE
Phabricator

PHABRICATOR

Qualimétrie - Analyse de code
LE QUIZZ

La revue de code permet de :

- ✗ Identifier les mauvais élèves
- ✓ Partager le savoir faire technique
- ✗ Coder en binôme
- ✗ Revoir l'architecture

Qualimétrie - Analyse de code
LE QUIZZ

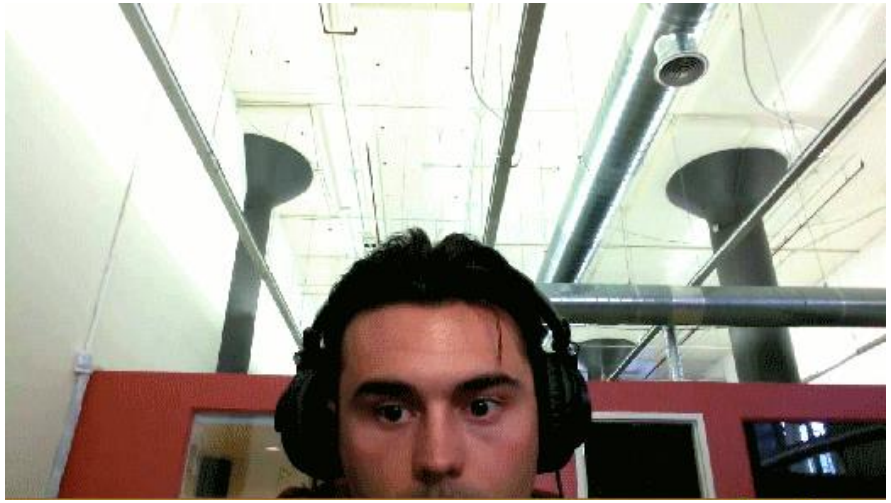
La revue de code est réalisée :

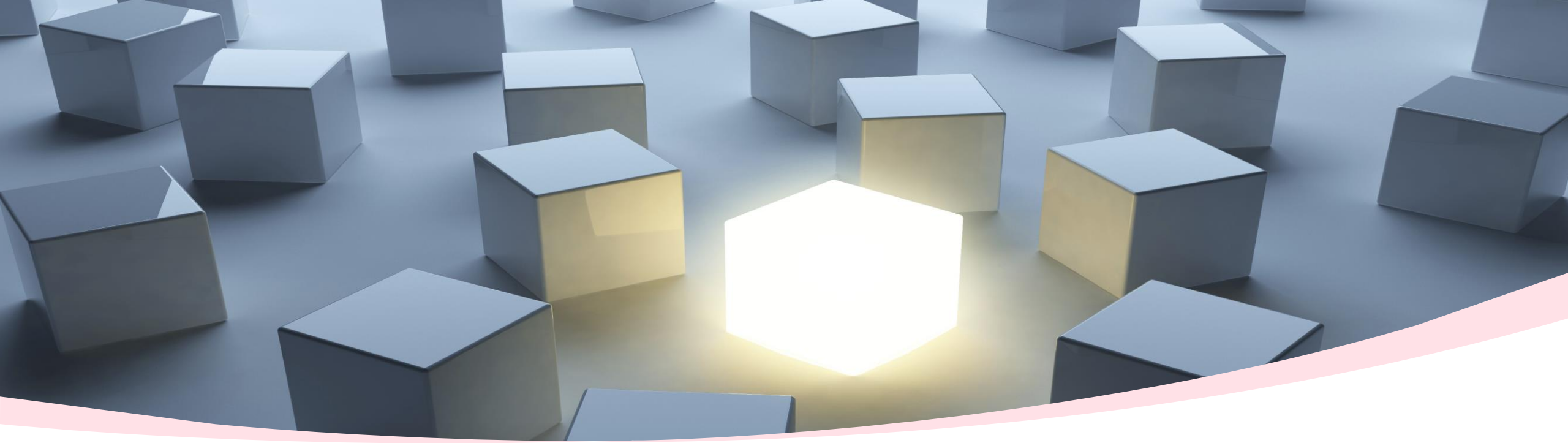
- ✗ Que par le référent technique du projet
- ✓ Sur les parties critiques du projet
- ✗ En fin de phase de développement
- ✗ Que sur le code produit par le stagiaire

Revue de code
POURQUOI AVOIR DES REVUES DE CODE ?

- **Pourquoi une revue de code ?**
  - Pour garantir la qualité (contrôle)
  - Pour faire grandir nos collaborateurs (audités et auditeurs)
  - Pour partager la connaissance sur le patrimoine applicatif.

- **La revue de code est le complément indispensable de l'analyse statique de code.**

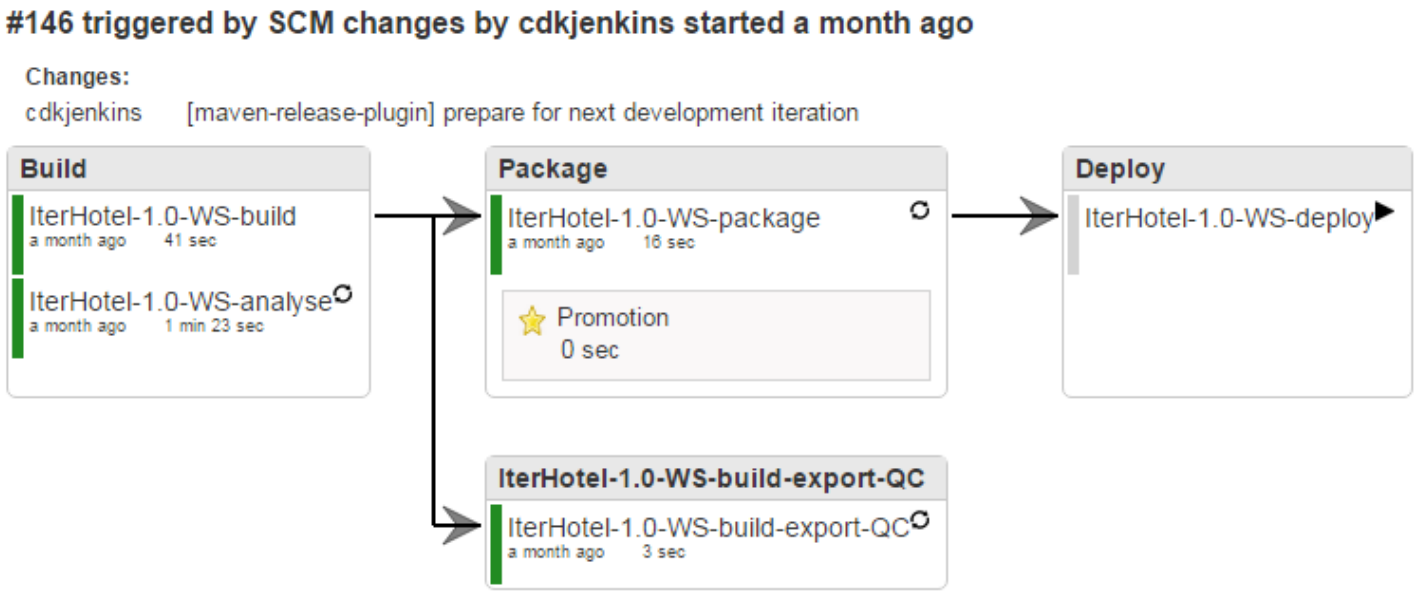




PACKAGING ET PROMOTION

Packaging
POURQUOI AVOIR UN PACKAGING AUTOMATISÉ ?

- **Qu'est ce que le packaging ?**
  - C'est l'activité de construction de la solution pour permettre son déploiement et sa livraison au client.
- **Pourquoi un Packaging automatisé ?**
  - Pour construire le package de la solution en iso client
  - Pour construire de façon répétitive et donc fiable
  - Pour ne pas dépendre d'une personne
  - Pour permettre un déploiement automatisé et donc simple et rapide



Packaging
COMMENT FAIRE UN PACKAGING AUTOMATISÉ ?

- **Maitriser sa solution et l'environnement de production**
  - Organisation système
  - Répartition de la solution
- **Connaitre les processus de mise en production Client**
  - Utilisation d'un outil spécifique
  - Installation manuelle
  - Demande client d'un script automatique type « Presse-Bouton »
- **Utiliser des outils de construction**
  - Maven package
  - Maven Assembly
  - MSBuild
  - Scripts
  - La configuration applicative doit être accessible sans impacter le package
- **Le packaging doit être documenté.**

Packaging
COMMENT FAIRE UN PACKAGING AUTOMATISÉ ?

- **Le package produit doit être testé :**
  - Utilisé lors de déploiement d'intégration et de qualification
  - Cela sera le package livré
- **Le périmètre du package doit être connu :**
  - Anomalies et évolutions prises en compte
  - Association à un tag de gestion de configuration
- **Traçabilité de la solution :**
  - Intégration des métadonnées permettant de connaitre l'origine du package
  - Rendre ces données accessibles de façon directe

Packaging
L'ASSISTANCE À LA LIVRAISON

- **Quel est le contexte ?**
  - Quelles sont les informations nécessaires pour la livraison ?
  - Quelles sont les informations disponibles ?
  - Peut-on les fournir en automatique ?
- **Qu'est ce qui est faisable simplement ?**
  - Lister les fichiers modifiés (svn diff)
  - Lister le contenu du colis (à partir du tar.gz, du zip, …)
  - Lister les fiches Jira de la version (via un plugin Jira)
  - Lister les modifications via le message de commit (svn log)
- **Comment ?**
  - Au sein du Delivery Pipeline
  - Via l'outil de gestion de configuration
  - Via un mail riche avec des pièces contenant les informations extraites
  - Des étapes de scripts shell sont souvent très utiles.

Est-ce que c’est possible ?

Exemple : Le KIT d'industrialisation Sopra Steria (CDK)

Exemple : Le KIT d'industrialisation Sopra Steria (CDK)

- des logiciels open-source customisés
- de la documentation, des guides
- des best-practices documentées

Plateforme SaaS

- Mise à disposition de tous les outils

Communauté

- Partage de l'information, échange de pratiques
- Amélioration continue

- **Capture des exigences**
- **Gestion du patrimoine des exigences**
- **Traçabilité de bout en bout**
- **Compilation**
- **Intégration continue avancée**
- **Automatisation des tests unitaires**
- **Gestion des revues de code**
- **Qualimétrie**
- **Automatisation du packaging**
- **Gestion du patrimoine de tests**
- **Modèles de stratégie de tests**
- **Automatisation des tests de non régression**
- **Test de Sécurité Applicative**
- **Anonymisation des données sensibles**
- **Contrôle des licences**
- **Modèles de stratégie de tests de perf. et bilan de tests**
- **Automatisation des tests de performances**
- **Déploiement et installation automatique**
- **Analyse automatique des logs**
- **Supervision**

sopra steria

CONTINUOUS DELIVERY KIT

UNE VEILLE PERMANENTE POUR ENRICHIR LA SOLUTION OU INTÉGRER DE MEILLEURS COMPOSANTS

- **HP ALM ou HP QC pour:**
  - Le référentiel des exigences
  - Le référentiel des tests
- **Jenkins pour:**
  - L'intégration continue
  - L'automatisation des tests (fonctionnels et performance)
- **Nuget, Nexus et Maven pour la gestion des dépendances et des artéfacts**
- **Phabricator pour les revues de code**
- **Junit pour les tests unitaires**
- **Sonarqube et FxCop pour l'analyse statique de code**
- **OWASP Zap pour la détection des vulnérabilités Web, OWASP Dependency Check pour la vérification des dépendances, LIcenseFinder pour détecter les licences**
- **Gherkin/Cucumber pour l'automatisation des tests fonctionnels**
- **Loadrunner et Jmeter pour les tests de performance**
- **Kibana pour l'Application Performance Monitoring**
- **Elastic Search, Logstash et Kibana pour la gestion des logs**
- **Rundeck et Docker pour la gestion du déploiement continu**

sopra steria