

# PRÉSENTATION ÉTUDE SCIENTIFIQUE ET TECHNIQUE



Romain GIRARD • INFO • E4

# Introduction et contexte

## Contexte

---

*Apprenti depuis septembre 2024 à MBDA France sur des missions de développement logiciel et sur l'intégration continue de ces développements.*

### Enjeux rencontrés :

Fiabilité

Délais

Livraisons de code

## Pourquoi étudier l'intégration continue ?

*Comment l'intégration continue influence-t-elle la qualité logicielle et les cycles de développement ?*



### Comprendre

les mécanismes de la CI



### Analyser

les impacts sur la qualité logicielle



### Évaluer

les effets sur le cycle de développement

### ✗ Exclu du périmètre de l'étude

Infrastructures spécifiques (embarqué, etc.)

Cadre spécifique de développement (Sécurité de l'information)

## Avant l'intégration continue : la gestion de configuration

*Processus d'ingénierie des systèmes qui aide les entreprises à maintenir la qualité des performances et le fonctionnement d'un produit, d'un système ou d'un autre actif informatique donné tout au long de son cycle de vie. — IBM*

### Méthodes classiques de gestion

#### Gestion de versions

*Git*

#### Gestion des branches

*GitFlow, Trunk-based*

#### Gestion des environnements

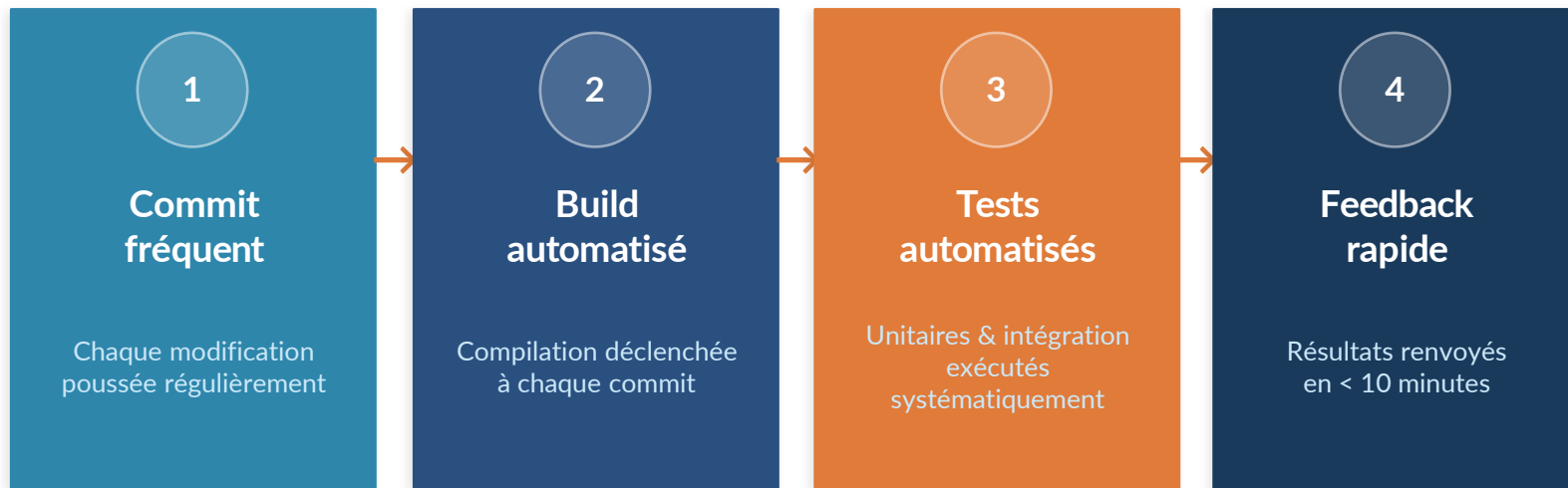
*Docker, configs partagées*

*« CI is a widely established development practice [...] members of a team integrate and merge development work frequently, for example multiple times per day. »*

*— Shahin, Babar & Zhu, Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices (2017) — Section II.A*

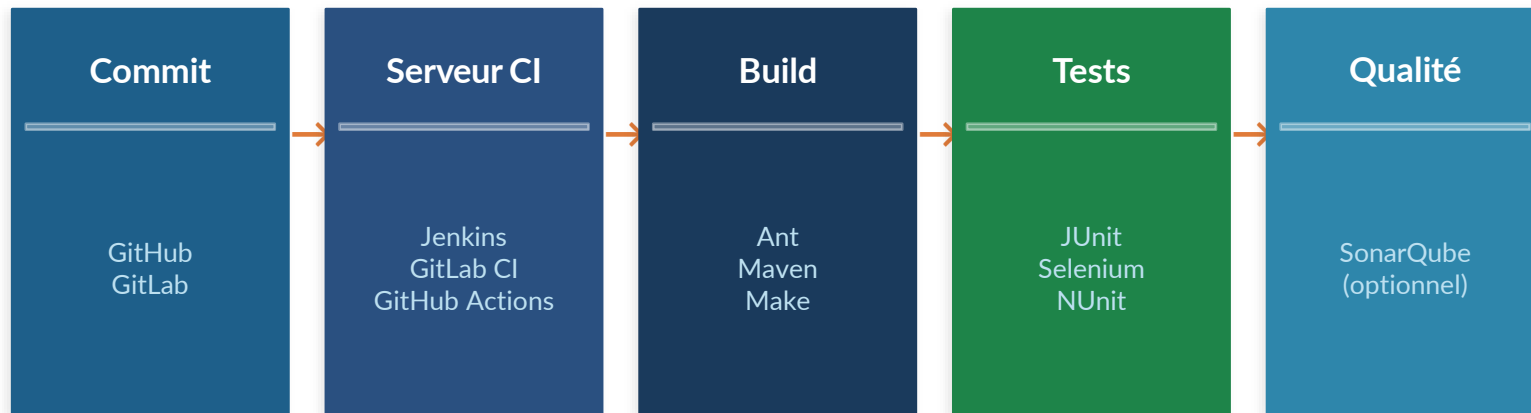
# L'intégration continue : définition et principes fondamentaux

« Continuous Integration requires that every time somebody commits any change, the entire application is built and a comprehensive set of automated tests is run against it... The goal of continuous integration is that the software is in a working state all the time.»  
— Humble & Farley, Continuous Delivery (2010) — Ch. 3



*Continuous integration is a practice, not a tool. It requires a degree of commitment and discipline from your development team.*  
(Humble & Farley, Continuous Delivery, 2010) — Ch. 3

## Un pipeline d'intégration continue typique



Git push → Déclenchement automatique → Pipeline complet en < 10 min

« Jenkins has gained the most attention among existing CI servers [...]. A deployment pipeline should include explicit stages [...]. Automation is a critical practice. »

— Shahin, Babar & Zhu, Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices (2017) — Section IV.C

« We mined the build history of projects in the Gitlab CI service. Additionally, we collected code coverage metrics from the Sonarqube tool. »

— Santos, da Costa & Kulesza (2025), Monitoring Continuous Integration Practices in Industry: A Case Study — Section III

# Intégration continue et qualité logicielle - Qualité du code

## Métriques clés suivies en CI

Couverture de tests  
(%)

Complexité  
cognitive

Dette technique  
(ex: SonarQube)

## Effet de la CI sur la qualité du code

Sans CI	Avec CI
Bugs détectés tard (coût élevé)	Détection immédiate au commit
Standards de code non vérifiés	Linting et analyse automatiques
Dette technique invisible	Métriques suivies en continu

« Median build success rate: 79.5% for commits [...] CI as a quality control mechanism is common to both commercial and open source software development. »

— Vasilescu et al., Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub (2014) — Section III.C

« Monitoring CI practices proved to be easy to use and low-cost [...] it brings value [...]. It has notifications so that developers always know what the values are like. »

— Santos, da Costa & Kulesza, Monitoring Continuous Integration Practices in Industry: A Case Study (2025) — Section IV.B

# Intégration continue et qualité logicielle - Fiabilité & maintenabilité

## Bénéfices mesurés sur la durée



### Régressions réduites

Tests rejoués à chaque commit  
→ bugs réintroduits détectés immédiatement



### Stabilité accrue

Versions livrées plus fiables,  
moins de bugs en production



### Maintenabilité facilitée

Intégrations fréquentes = dette technique qui s'accumule moins vite

« The "Time to Fix a Broken Build" experienced a significant reduction [...]. As a consequence, the "Build Health" has increased significantly. »

– Santos, da Costa & Kulesza, *Monitoring Continuous Integration Practices in Industry: A Case Study (2025)* – Section IV.E

« The aim of the deployment pipeline is [...] to improve feedback so that problems are identified, and so resolved, as early in the process as possible. »

– Humble & Farley, *Continuous Delivery (2010)* – Ch. 1



## Intégration continue et cycles de développement : plus vite, mieux



### Accélération du cycle

- 1 Feedback en < 10 min après chaque commit
- 2 Réduction du temps perdu à déboguer tardivement
- 3 Intégrations multiples par jour (vs hebdomadaire)



### Compatibilité avec l'Agile

- 1 S'intègre naturellement aux sprints courts
- 2 Livraisons plus fréquentes et fiables
- 3 Soutient l'amélioration continue

« Long feedback loops means a higher cost for the project »

— Mårtensson et al., *Continuous Integration is Not About Build Systems* (2017) — Section III.B


## L'intégration continue : ni magique, ni gratuite

Coût initial de mise en place  
(infrastructure, formation)

Maturité & discipline de l'équipe  
nécessaires

Maintenance des pipelines  
(tests cassés, faux positifs)

Résistance au changement culturel

 **La CI seule ne garantit pas la qualité — elle nécessite des tests pertinents et maintenus. Un pipeline avec de mauvais tests donne une fausse assurance.**

**Build vert = priorité absolue**

*Ne jamais laisser un build cassé*

**Build court (< 10 min)**

*Pour ne pas freiner les commits*

**Chaque échec traité immédiatement**

*Responsabilité collective*

**Pipeline versionné comme le code**

*Reproductibilité garantie*

« "Testing" is the most frequently mentioned factor for CI success (39.1%). CI demands new technical and soft skills. »  
— Shahin et al., *Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices* (2017) —Section IV.B and D

## Synthèse et ouverture

### Sur la qualité logicielle

- Détection précoce des défauts
- Réduction des régressions
- Métriques de qualité suivies en continu

### Sur les cycles de développement

- Feedback rapide aux développeurs
- Accélération des livraisons
- Alignement naturel avec l'Agile

La CI améliore mesurément la qualité logicielle en détectant les défauts au plus tôt, et transforme les cycles de développement en rendant les livraisons plus fréquentes, plus fiables et naturellement alignées avec les pratiques agiles — à condition que l'équipe adopte la discipline qu'elle exige.

*Cette discipline exigée et la taille des projets grandissant peut nous amener à explorer l'utilisation de l'IA pour faciliter les processus d'intégrations par exemple, en générant automatiquement les tests et/ou en faisant des analyses prédictive de qualité.*

## Références

M. Fowler, "Continuous Integration," martinowler.com, 2024.  
<https://martinowler.com/articles/continuousIntegration.html>.

N. Cassee, B. Vasilescu and A. Serebrenik, "The Silent Helper: The Impact of Continuous Integration on Code Reviews," 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, ON, Canada, 2020, pp. 423-434, doi: 10.1109/SANER48275.2020.9054818. keywords: {Codes;Conferences;Focusing;Software quality;Data models;Open source software;Software engineering},

Shubham and L. M. Saini, "The Impact of Devops on Software Quality," 2024 Third International Conference on Smart Technologies and Systems for Next Generation Computing (ICSTSN), Villupuram, India, 2024, pp. 1-6, doi: 10.1109/ICSTSN61422.2024.10670849. keywords: {Surveys;DevOps;Automation;Correlation;Quality assurance;Bibliographies;Software quality;Software Quality;Devops;Software Development Life Cycle;Continuous Integration;Continuous Development},

S. A. I. B. S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," 2018 Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 2018, pp. 156-161, doi: 10.1109/MERCon.2018.8421965. keywords: {Production;Benchmark testing;Software;Tools;Automation;Pipelines;continuous integration;continuous delivery;agile project management;version management;configuration management},

T. Mårtensson, P. Hammarström and J. Bosch, "Continuous Integration is Not About Build Systems," 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, Austria, 2017, pp. 1-9, doi: 10.1109/SEAA.2017.30. keywords: {Interviews;Software;Measurement;Industries;Pipelines;Companies;Testing;software integration;continuous integration;developer behaviors;build system},

## Références

B. Vasilescu, S. van Schuylenburg, J. Wulms, A. Serebrenik and M. G. J. van den Brand, "Continuous Integration in a Social-Coding World: Empirical Evidence from GitHub," 2014 IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, 2014, pp. 401-405, doi: 10.1109/ICSME.2014.62. keywords: {Java;Open source software;Encoding;Blogs;Programming;continuous integration;automatic build;GitHub;collaborative software development},

J. Santos, D. A. da Costa and U. Kulesza, "Monitoring Continuous Integration Practices in Industry: A Case Study," 2025 IEEE International Conference on Software Maintenance and Evolution (ICSME), Auckland, New Zealand, 2025, pp. 721-731, doi: 10.1109/ICSME64153.2025.00072. keywords: {Industries;Measurement;Surveys;Software maintenance;DevOps;Companies;Continuous integration;Interviews;Monitoring;Best practices;CI Practices;CI Metrics;Monitoring;Case Study;CI Maturity},

M. Shahin, M. Ali Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," in IEEE Access, vol. 5, pp. 3909-3943, 2017, doi: 10.1109/ACCESS.2017.2685629. keywords: {Software;Organizations;Software engineering;Systematics;Bibliographies;Testing;Production;Continuous integration;continuous delivery;continuous deployment;continuous software engineering;systematic literature review;empirical software engineering},

J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Upper Saddle River, NJ, USA: Addison-Wesley, 2010.