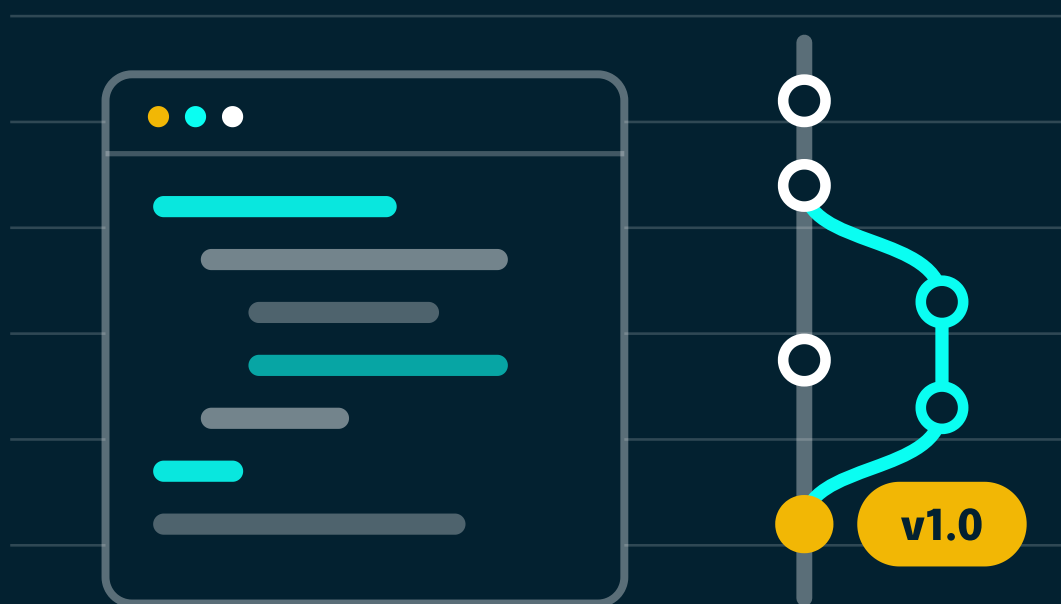


LIVRE BLANC · IT ET NUMÉRIQUE

Claude Code en équipe : **coder, relire et documenter** avec l'IA.

Pour les développeurs, les tech leads et les CTO des ESN, des éditeurs et des DSI. Explorer un dépôt, poser les règles de l'équipe, implémenter, tester, relire, documenter, moderniser, accueillir un nouveau développeur : dix cas où Claude Code propose et où l'équipe garde la main, avec des prompts prêts à copier.



SOMMAIRE

Dix cas d'usage, de la carte du dépôt à l'accueil d'un nouveau développeur.

Chaque cas tient sur une page : pour qui, avec quel outil, quelles données fournir, le prompt à copier, ce que vous obtenez et le point de vigilance. Commencez par les cinq règles de la page suivante.

–	Avant de commencer : cinq règles pour vos données	p. 3
01	Explorer un dépôt inconnu et en dresser la carte	p. 4
02	Poser les règles de l'équipe pour Claude Code	p. 5
03	Implémenter une user story sur une branche	p. 6
04	Écrire les tests qui manquent	p. 7
05	Corriger un bug à partir d'un ticket	p. 8
06	Relire une pull request avec Claude Code	p. 9
07	Documenter un module et tenir le README à jour	p. 10
08	Préparer les notes de version	p. 11
09	Résorber la dette technique par petites étapes	p. 12
10	Accueillir un nouveau développeur avec Claude Code	p. 13
–	Aller plus loin avec Wavia	p. 14

Cinq règles pour coder avec Claude Code sans perdre la main sur la qualité et la sécurité.

1

Des droits limités, réglés une fois pour toute l'équipe

Dossiers autorisés, commandes à confirmer, aucun accès aux environnements de production : les permissions de Claude Code se règlent dans le projet et se relisent comme du code.

2

Aucun secret à portée de l'IA

Les clés et mots de passe vivent dans un coffre ou dans des variables d'environnement exclues du dépôt et des fichiers lus par Claude Code. Un contrôle automatique des secrets tourne avant chaque commit.

3

Une branche, un plan, une revue

Claude Code travaille sur une branche, propose un plan avant d'écrire, et chaque pull request est relue et fusionnée par un humain. Le développeur reste l'auteur de ce qu'il pousse.

4

Les tests d'abord

Un test qui reproduit le bug ou qui fige le comportement actuel avant toute modification : c'est ce qui permet de faire confiance au code proposé par l'IA.

5

Une charte d'équipe et une formation

Outils et offres autorisés, code client concerné ou non, licences des dépendances, mention de l'IA dans les pull requests : une page signée par l'équipe, et une formation pratique sur vos dépôts pour que chacun applique les mêmes règles.

01

Explorer un dépôt inconnu et en dresser la carte

POUR QUI

Développeurs qui arrivent sur un projet, consultants en régie, tech leads

OUTIL CONSEILLÉ

Claude Code dans le terminal, en lecture seule

DONNÉES À FOURNIR

Le dépôt du projet, sa documentation existante

LE PROMPT, À COPIER ET ADAPTER

Explore ce dépôt sans rien modifier.

1. Explique en dix lignes à quoi sert l'application et comment elle est organisée (dossiers, couches, points d'entrée).
2. Liste les technologies et les versions utilisées, et la façon de lancer le projet et les tests en local.
3. Décris le parcours d'une requête type, de [point d'entrée] jusqu'à la base de données.
4. Signale ce qui t'étonne : code dupliqué, dossiers inutilisés, conventions contradictoires.

Écris le résultat dans un fichier [docs/carte-du-projet.md], sans toucher au reste.

Ce que vous obtenez

Une carte du projet écrite en une heure, au lieu de plusieurs jours à lire le code seul.

Point de vigilance

Vérifier la carte avec un développeur qui connaît le projet : l'IA peut mal interpréter un choix ancien.

02

Poser les règles de l'équipe pour Claude Code

POUR QUI

Tech leads, CTO

OUTIL CONSEILLÉ

Claude Code (fichier CLAUDE.md et réglages de permissions du projet)

DONNÉES À FOURNIR

Vos conventions de code, vos commandes de test, vos règles de sécurité

LE PROMPT, À COPIER ET ADAPTER

Rédige le fichier CLAUDE.md de ce dépôt, lu par Claude Code à chaque session.

1. Résume l'architecture et les conventions de nommage en vigueur, en t'appuyant sur le code existant.
2. Liste les commandes à utiliser : installation, tests, linter, lancement en local.
3. Écris les interdits : aucun secret dans le code, aucune modification de [dossiers sensibles], aucune dépendance ajoutée sans le signaler.
4. Propose les réglages de permissions : commandes autorisées, commandes à confirmer, dossiers exclus.

Une page au plus, des phrases courtes, à faire valider par l'équipe.

Ce que vous obtenez

Des règles communes que Claude Code applique pour chaque développeur, versionnées avec le code.

Point de vigilance

Le fichier se relit et se fusionne comme du code : une règle mal écrite s'applique à toute l'équipe.

03

Implémenter une user story sur une branche

POUR QUI

Développeurs

OUTIL CONSEILLÉ

Claude Code dans le terminal

DONNÉES À FOURNIR

La user story et ses critères d'acceptation, le dépôt

LE PROMPT, À COPIER ET ADAPTER

Crée une branche [nom de branche] et implémente la story suivante : « [user story] », critères d'acceptation : [critères].

1. Commence par me proposer un plan : fichiers touchés, approche, points de doute. Attends mon accord.
2. Implémente par petites étapes, en respectant les conventions du CLAUDE.md.
3. Écris les tests qui vérifient chaque critère d'acceptation et lance-les.
4. Résume ce que tu as fait, ce que tu n'as pas fait et ce qu'il faut vérifier à la main.

Ne pousse rien et n'ouvre aucune pull request sans mon accord.

Ce que vous obtenez

Une story implémentée et testée sur sa branche, avec un plan validé avant la première ligne.

Point de vigilance

Relire chaque modification comme celle d'un collègue : le développeur reste l'auteur de ce qu'il pousse.

04

Écrire les tests qui manquent

POUR QUI

Développeurs, QA

OUTIL CONSEILLÉ

Claude Code dans le terminal

DONNÉES À FOURNIR

Le module visé, ses tests existants, le rapport de couverture

LE PROMPT, À COPIER ET ADAPTER

Voici le rapport de couverture de tests (fichier joint) pour le module [chemin].

1. Liste les fonctions et les branches non couvertes, classées par risque métier.
 2. Pour les cinq plus risquées, écris des tests unitaires dans le style des tests existants : cas nominal, cas limites, erreurs attendues.
 3. Lance les tests et corrige uniquement les tests, jamais le code testé.
 4. Si un test révèle un comportement douteux du code, signale-le au lieu de l'adapter.
- Résume à la fin la couverture avant et après.

Ce que vous obtenez

Les zones à risque couvertes par des tests lisibles, et les comportements suspects mis en évidence.

Point de vigilance

Un test écrit pour passer ne prouve rien : vérifier qu'il échoue bien quand on casse le code.

05

Corriger un bug à partir d'un ticket

POUR QUI

Développeurs, support
de niveau 3

OUTIL CONSEILLÉ

Claude Code dans le
terminal

DONNÉES À FOURNIR

Le ticket (description,
étapes pour reproduire,
journaux nettoyés), le
dépôt

LE PROMPT, À COPIER ET ADAPTER

Voici le ticket [référence] : « [description du bug] », étapes pour reproduire : [étapes].

1. Trouve le code en cause et explique la cause probable avant de modifier quoi que ce soit.
 2. Écris d'abord un test qui reproduit le bug et vérifie qu'il échoue.
 3. Corrige le bug avec la modification la plus petite possible, puis relance tous les tests.
 4. Rédige le message de commit et le commentaire pour le ticket, compréhensible par le support.
- Si tu trouves plusieurs causes possibles, arrête-toi et demande-moi.

Ce que vous obtenez

Un correctif minimal, prouvé par un test, et une explication claire pour le support.

Point de vigilance

Les journaux fournis à l'IA sont nettoyés de toute donnée client et de tout jeton d'accès.

06

Relire une pull request avec Claude Code

POUR QUI

Relecteurs, tech leads

OUTIL CONSEILLÉ

Claude Code dans le terminal, ou Claude relié à GitHub (serveur MCP officiel)

DONNÉES À FOURNIR

La pull request, le ticket associé, le CLAUDE.md

LE PROMPT, À COPIER ET ADAPTER

Relis la pull request [numéro] par rapport au ticket [référence] et aux règles du CLAUDE.md.

1. Résume le changement en cinq lignes.
 2. Classe tes remarques en trois niveaux : bloquant, à corriger, suggestion.
 3. Vérifie en priorité : sécurité (entrées non validées, droits, secrets), cas limites, tests absents, migrations de base de données.
 4. Rédige chaque commentaire relié à sa ligne, sur un ton de collègue.
- Ne poste rien : je relis et je publie moi-même.

Ce que vous obtenez

Une première revue structurée en quelques minutes, que le relecteur complète et publie.

Point de vigilance

La revue humaine reste obligatoire avant la fusion, quel que soit l'auteur du code.

07

Documenter un module et tenir le README à jour

POUR QUI

Développeurs, tech writers

OUTIL CONSEILLÉ

Claude Code dans le terminal

DONNÉES À FOURNIR

Le module, ses tests, le README actuel

LE PROMPT, À COPIER ET ADAPTER

Compare le README et la documentation de [module] avec le code actuel.

1. Liste ce qui est obsolète, faux ou manquant dans la documentation.
2. Réécris les sections concernées pour un développeur qui découvre le module, avec un exemple d'utilisation testé.
3. Ajoute les commentaires utiles dans le code là où l'intention n'est pas évidente, sans paraphraser le code.
4. Prépare la modification sur une branche [nom], avec un message de commit clair.

Ne modifie aucune ligne de code exécutable.

Ce que vous obtenez

Une documentation qui colle au code, mise à jour dans la même branche que la fonctionnalité.

Point de vigilance

Faire relire par l'auteur du module : la documentation engage autant que le code.

08

Préparer les notes de version

POUR QUI

Tech leads, product owners, support

OUTIL CONSEILLÉ

Claude Code ou Claude relié à GitHub

DONNÉES À FOURNIR

La liste des pull requests fusionnées depuis la dernière version, les tickets associés

LE PROMPT, À COPIER ET ADAPTER

Liste les pull requests fusionnées depuis la version [numéro de la version précédente].

1. Classe-les : nouveautés, améliorations, corrections, changements techniques internes.
2. Rédige les notes de version pour les utilisateurs, sans jargon, une phrase par changement visible.
3. Rédige à part la version technique pour l'équipe, avec les migrations et les points de vigilance du déploiement.
4. Signale les changements qui demandent une action des utilisateurs ou du support.

Ne mentionne aucune pull request que tu ne trouves pas dans l'historique.

Ce que vous obtenez

Deux versions des notes de version, utilisateurs et technique, prêtes le jour de la livraison.

Point de vigilance

Relire avec le product owner : une nouveauté annoncée trop tôt crée des tickets au support.

09

Résorber la dette technique par petites étapes

POUR QUI

Tech leads, architectes

OUTIL CONSEILLÉ

Claude Code dans le terminal

DONNÉES À FOURNIR

Le module à moderniser, ses tests, le plan de migration validé

LE PROMPT, À COPIER ET ADAPTER

Nous modernisons [module] : [objectif, par exemple passer de callbacks à async/await ou retirer une dépendance obsolète].

1. Vérifie que les tests existants couvrent le comportement actuel ; sinon, écris d'abord les tests manquants.
 2. Propose un découpage en étapes indépendantes, chacune livrable seule.
 3. Réalise uniquement l'étape 1, relance tous les tests et montre-moi le diff.
 4. Note dans [docs/migration.md] ce qui est fait et ce qui reste.
- Aucun changement de comportement visible pour l'utilisateur.

Ce que vous obtenez

Une modernisation qui avance étape par étape, toujours testée, jamais en big bang.

Point de vigilance

Une étape par pull request : un gros diff généré d'un coup ne se relit pas sérieusement.

10

Accueillir un nouveau développeur avec Claude Code

POUR QUI

Tech leads, nouveaux arrivants, ESN qui placent des consultants

OUTIL CONSEILLÉ

Claude Code dans le terminal, avec le CLAUDE.md du projet

DONNÉES À FOURNIR

Le dépôt, la carte du projet, un premier ticket simple

LE PROMPT, À COPIER ET ADAPTER

Je viens d'arriver sur ce projet. Aide-moi à être autonome sans écrire le code à ma place.

1. Explique-moi le projet à partir du CLAUDE.md et de la carte du projet, en partant de ce qu'un utilisateur fait.
2. Montre-moi comment lancer le projet et les tests, et vérifie avec moi que tout fonctionne.
3. Pour le ticket [référence], indique-moi les fichiers à lire et pose-moi des questions pour que je trouve la solution.
4. Liste les cinq conventions de l'équipe que je dois connaître dès cette semaine.

Réponds à mes questions en citant les fichiers concernés.

Ce que vous obtenez

Un nouvel arrivant qui comprend le projet et livre son premier ticket en apprenant, pas en recopiant.

Point de vigilance

Garder un binôme humain : l'IA explique le code, pas l'histoire du projet ni les attentes du client.

ALLER PLUS LOIN

De ces dix prompts à une équipe qui code avec l'IA, avec les mêmes règles pour tous.

Wavia est une agence IA fondée à Nantes en 2023, certifiée Qualiopi. Nous formons les équipes à l'IA générative sur leurs vrais dossiers, nous relierons l'IA à leurs outils et nous développons les agents qui manquent.

Former les développeurs à Claude Code

Explorer, implémenter, tester, relire, documenter : une formation pratique par groupe de dix au plus, sur vos dépôts ou sur un projet d'exemple. Finançable par votre OPCO.

Paramétrer Claude Code pour l'équipe

Offres et comptes, CLAUDE.md, permissions, contrôle des secrets, charte d'usage de l'IA dans le code : un cadre commun posé avec votre CTO.

Relier Claude à GitHub et à vos outils

Serveur MCP officiel de GitHub, connecteurs Notion et Atlassian, ou connecteur sur mesure pour votre outil interne, en lecture seule d'abord.

Un agent IA sur mesure

Revue automatique des pull requests, notes de version, documentation tenue à jour : l'agent prépare, le développeur valide.

Réserver un échange de 15 minutes

Ou par téléphone : 06 44 22 69 86

4,9/5 sur plus de 280 avis Google · plus de 3 000 collaborateurs formés · wav-ia.fr/ia-it-numerique