CLAUDE CODE ULTIMATE GUIDE

Claude Code : De Zéro à Productif

Les fondamentaux pour démarrer efficacement

Florian BRUNIAUX

octobre 2026

v1.5.4

Guide v3.44.1



Table des matières

1 Qu'est-ce que Claude Code 4
1.1 Installation 5
2 Quickstart : Votre première session 7
2.1 Lancer Claude Code 7
2.2 Le workflow 7
2.3 Exemple concret 7
2.4 Commandes essentielles 8
2.5 Raccourcis clavier 8
3 Les 5 Règles d'Or 10
3.1 Le Mindset Clé : Système de Contexte, pas Chatbot 10
3.2 La formule WHAT/WHERE/HOW/VERIFY 11
3.3 Gestion du Contexte 11
4 La Série de Whitepapers 13
4.1 #1 : Prompts qui Marchent (~15 min) 13
4.2 #2 : Personnaliser Claude (~20 min) 13
4.3 #3 : Sécurité en Production (~25 min) 14
4.4 #4 : L'Architecture Démystifiée (~20 min) 14
4.5 #5 : Déployer en Équipe (~20 min) 14
4.6 #6 : Privacy & Compliance (~15 min) 15
4.7 #7 : Le Guide de Référence (~2h) 15
5 Parcours Recommandés 16
5.1 Solo vs Équipe vs Enterprise 16
6 Ressources 17
6.1 Liens essentiels 17
6.2 Conventions de Fiabilité 17
6.3 Fiche Récap : Introduction à Claude Code 19
7 La Série 20

📌 **Pour qui ce whitepaper ?**
Audience principale : Tous, développeurs, tech leads, managers

Prérequis : Aucun, point d'entrée de la série

Temps de lecture : 15 min

Ce que vous allez retenir :

- Claude Code = agent autonome avec accès à vos outils, pas un chatbot
- La formule WHAT/WHERE/HOW/VERIFY structure chaque interaction efficacement
- CLAUDE.md est la mémoire persistante du projet; le créer en premier

En 5 lignes : Claude Code n'est pas un assistant de chat avec des super-pouvoirs. C'est un agent autonome qui lit vos fichiers, exécute vos commandes et modifie votre code directement. La différence structurelle avec les outils précédents (Copilot, ChatGPT) est que Claude Code agit, il ne suggère pas. Ce guide explique comment transformer ce potentiel en productivité mesurable, de la première session à l'adoption en équipe.

1 Qu'est-ce que Claude Code

> *Ce whitepaper s'adresse principalement aux **développeurs** qui démarrent avec Claude Code. Si vous êtes non-développeur (manager, knowledge worker), consultez le Claude Cowork Guide dédié aux usages sans code.*

La plupart des développeurs qui adoptent Claude Code font la même erreur les premières semaines : ils l'utilisent comme un chatbot amélioré, avec des prompts vagues, et s'étonnent des résultats médiocres. Ce guide existe pour éviter ça.

Claude Code est l'outil CLI officiel d'Anthropic pour le développement assisté par IA. En une phrase : **un agent de codage autonome qui opère directement dans votre terminal, lit votre codebase, propose des modifications, et exécute des commandes avec votre approbation explicite**.

> **Après cette introduction, vous saurez** : installer Claude Code, lancer votre première session, appliquer les 5 règles d'or, et choisir votre parcours dans la série.

Caractéristique	Description
CLI Native	Pas d'IDE imposé, fonctionne dans n'importe quel terminal
Agentic	Analyse, planifie, exécute, pas juste de l'autocomplétion
Extensible	MCP (Model Context Protocol) pour connecter des outils externes
Personnalisable	Des Agents, Skills, Commands et Hooks personnalisables pour adapter Claude à vos workflows

Depuis fin 2025, plusieurs fonctionnalités agentic sont devenues disponibles et changent significativement ce que Claude Code peut faire en autonomie :

Fonctionnalité	Depuis	Description
Tasks API	v2.1.16	Listes persistantes avec dépendances ; activation explicite requise sur les modèles récents
Auto-memories	v2.1.32	Capture automatique du contexte cross-session dans `~/.claude/projects/`
Agent Teams	v2.1.32	Multi-agents en parallèle via `TeamCreate`/`SendMessage`
LSP Tool	v2.0.74	Diagnostics après chaque modification et navigation par symbole, activés par un plugin officiel de code intelligence par langage
Remote Control	v2.1.51	Piloter une session locale depuis mobile ou navigateur (Pro/Max/Team/Enterprise)
MCP Elicitation	v2.1.76	Les serveurs MCP peuvent demander des données structurées en cours de tâche

Fonctionnalité	Depuis	Description
Contexte 1M tokens	v2.1.75	1M natif sur les modèles actuels Opus, Sonnet, Haiku et Fable ; anciennes variantes soumises à conditions
`/loop`	v2.1.71+	Intervalle fixe si précisé ; sinon Claude choisit le délai
`/branch`	v2.1.78+	Fork de session en parallèle (`--fork-session`) pour explorer plusieurs approches
`--bare`	v2.1.78+	Mode scripted sans hooks/LSP/plugins, idéal pour CI/CD
Auto mode (Max)	v2.1.100+	Classification des permissions ; ne sélectionne pas le modèle
`/ultrareview`	v2.1.114+	Revue de code multi-agents en parallèle dans le cloud ; aussi disponible en CI/CD via `claude ultrareview` (v2.1.120)
`/goal`	v2.1.139+	Boucle d'exécution autonome : définissez une condition, Claude travaille jusqu'à ce qu'un évaluateur séparé (Haiku par défaut sur l'API Claude) confirme qu'elle est atteinte (plus besoin de taper « continue »)
Opus 5.5	Modèle actuel	Défaut du service direct, 1M natif ; tarif standard $4/$20 et fast mode $8/$40 par Mtok
`--safe-mode`	v2.1.169	Flag de dépannage : désactive toutes les personnalisations (CLAUDE.md, plugins, skills, hooks, serveurs MCP)
Fable 5.1	Modèle actuel	Modèle API direct sur comptes éligibles ; les alias des fournisseurs peuvent différer
Sub-agents imbriqués	v2.1.219	Un agent peut déléguer à d'autres agents jusqu'à 3 niveaux de profondeur par défaut (`CLAUDE_CODE_MAX_SUBAGENT_SPAWN_DEPTH`)

1.1 Installation

Sonnet 5.5 (`claude-sonnet-5-5`) remplace Sonnet 5 sur l'API directe depuis Claude Code v2.1.284, le 28 septembre 2026, au même tarif standard $2/$10 par million de tokens. Sonnet 5 reste un modèle historique ; vérifiez le modèle effectif avec `/model`. Configuration des modèles.

Deux options d'installation. L'installeur natif est la méthode recommandée par Anthropic ; npm reste fonctionnel si vous avez des contraintes spécifiques.

```
# Méthode recommandée : installeur natif (pas de dépendance Node.js)
curl -fsSL https://claude.ai/install.sh | bash

# Alternative : via npm (non recommandé, toujours fonctionnel)
npm install -g @anthropic-ai/claude-code

# Vérifier l'installation
claude --version
```

```
# Première authentification
claude
# Suivre les instructions pour lier votre compte Anthropic
```

Prérequis : un abonnement Claude éligible ou un accès API/fournisseur. Node.js 18+ est nécessaire uniquement pour l'installation npm.

Vérifiez le mode d'authentification : abonnement Claude éligible ou accès API/fournisseur. Un abonnement Claude ne nécessite pas systématiquement une clé API.

Pour aller plus loin : §1.1 Installation complète: troubleshooting, prérequis détaillés, configuration initiale

2 Quickstart : Votre première session

2.1 Lancer Claude Code

Claude Code se lance toujours depuis la racine du projet sur lequel vous travaillez. C'est ce répertoire de départ qui détermine quels fichiers seront analysés et quel `CLAUDE.md` sera chargé; ce fichier texte placé à la racine de votre projet sert de mémoire persistante et de règles de travail pour Claude (voir la section « Règles d'Or » ci-dessous, et la règle n°5).

```
cd mon-projet
claude
```

Claude analyse automatiquement votre projet et attend vos instructions.

2.2 Le workflow

Claude Code propose une boucle de validation humaine. Les demandes d'approbation dépendent du mode de permission, des règles enregistrées et de la politique de l'organisation. Les lectures et actions déjà autorisées peuvent s'exécuter sans nouvelle demande.

```
Vous décrivez → Claude analyse → Claude propose → Vous validez → Claude exécute
```

Principe clé : choisissez délibérément le mode de permission et relisez les modifications avant publication.

2.3 Exemple concret

Exemple avec validation des éditions : Claude décompose la tâche en étapes visibles. Selon les permissions déjà accordées, certaines actions peuvent s'exécuter sans nouvelle demande.

```
> Ajoute une validation email au formulaire de login

Claude va :
```

```
1. Lire src/components/LoginForm.tsx
2. Proposer une modification avec diff
3. Attendre votre "y" pour appliquer
4. Optionnellement lancer les tests
```

2.4 Commandes essentielles

Ces commandes couvrent 90% des interactions quotidiennes. `claude`, `/status` et `/compact` forment le trio minimal à connaître dès le premier jour ; les autres s'ajoutent naturellement au fil de la pratique.

```
claude          # Lancer Claude Code
/init           # Générer un CLAUDE.md initial pour votre projet
/status         # Vérifier l'état de session et du compte
/compact        # Compresser le contexte (dès 70%)
/usage          # Coûts, stats et graphiques de session (v2.1.118, remplace /cost et /stats)
/context        # Répartition des tokens ; utiliser @file dans le prompt
/clear          # Repartir à zéro
/plan           # Mode plan (lecture seule, exploration sûre)
/doctor         # Diagnostiquer la config (sanity check)
```

2.5 Raccourcis clavier

Les raccourcis clavier accélèrent la navigation entre les modes de Claude Code sans quitter le terminal. Le plus utilisé est `Shift+Tab`, qui bascule entre le mode normal, l'auto-accept (pour les modifications de confiance), et le mode plan (lecture seule).

Raccourci	Action
`Shift+Tab`	Cycle : default → auto-accept → mode plan
`Alt+T`	Basculer le thinking si pris en charge ; aucun effet sur Opus 5.5, Sonnet 5.5, Haiku 5.5 ni Fable
`Ctrl+B`	Exécuter une commande en arrière-plan
`Ctrl+R`	Historique des commandes
`@fichier`	Référencer un fichier spécifique
`!command`	Exécuter une commande shell
`Ctrl+C`	Annuler l'opération en cours
`Esc` x2	Ouvrir le retour arrière et choisir un checkpoint

📌 Piège à éviter : Permission Fatigue

Le **piège #1 signalé par la communauté** : approuver les prompts de permission sans les lire, puis céder à la tentation de `--dangerously-skip-permissions` sur une machine non isolée. Résultat : une commande malformée ou une injection de prompt peut exécuter du code sans contrôle.

Choisir le bon mode selon le contexte :

Mode	Quand l'utiliser
Default (validation explicite)	Développement quotidien, nouveaux projets
Auto-accept (`Shift+Tab`)	Scripts de confiance, CI/CD contrôlé
bypassPermissions	Sandboxes isolées uniquement
`--dangerously-skip-permissions`	À éviter, risque élevé sans sandbox

Règle simple : si vous n'êtes pas dans un sandbox isolé, gardez la validation manuelle.

3 Les 5 Règles d'Or

Ces règles condensent les erreurs les plus fréquentes observées lors des premières semaines d'utilisation de Claude Code. Les appliquer dès le départ évite la majorité des frustrations signalées par la communauté.

1. **Toujours relire les diffs** avant d'accepter les modifications
2. **Utiliser** `/compact` à une frontière utile de la tâche ; consulter `/context` (rappels ci-dessous)
3. **Être spécifique** dans les demandes (voir formule ci-dessous)
4. **Mode plan d'abord** pour les tâches complexes ou risquées
5. **Créer un CLAUDE.md** pour chaque projet; vous pouvez en avoir plusieurs : un global (`~/.claude/CLAUDE.md`), un par projet (racine), et un par sous-répertoire pour des règles plus fines

3.1 Le Mindset Clé : Système de Contexte, pas Chatbot

> **«Stop treating it like a chatbot. Give it structured context. Changes everything.»** (Robin Lorenz)

Claude Code n'est pas un chatbot amélioré, c'est un **système de contexte à 4 couches**. Chaque couche représente un niveau de mémoire persistante que vous construisez progressivement : la première (CLAUDE.md) se met en place en quelques minutes, les suivantes s'ajoutent au fil des semaines à mesure que vous identifiez les patterns récurrents de votre workflow.

Couche	Mécanisme	Quand l'ajouter
Mémoire	`CLAUDE.md`	Semaine 1
Connaissances	Skills	Semaine 2
Automatisation	Hooks	Semaine 3
Persistance	Project memory	En continu

Résultat : chaque couche rend la suivante plus efficace.

> **Auto Dream** (v2.1.78+) : après 24h d'inactivité et 5 sessions, Claude consolide automatiquement sa mémoire projet en 4 phases (orientation, collecte, consolidation, indexation). Accessible via `/memory`.

> 📌 **Scénario illustratif : Antoine, dev full-stack dans un cabinet de conseil tech**
> Antoine commence par un `CLAUDE.md` basique listant sa stack (Next.js, Prisma, PostgreSQL). Dès la première semaine, il constate que Claude respecte ses conventions sans qu'il ait besoin de les rappeler. En semaine 2, il ajoute un Skill « conventions API REST » qui encode ses patterns de validation. En semaine 3, un hook pre-commit vérifie automatiquement que les migrations Prisma sont générées. Résultat : ses prompts passent de 4 lignes de contexte à une simple instruction, et Claude produit du code conforme du premier coup.

3.2 La formule WHAT/WHERE/HOW/VERIFY

À appliquer pour chaque prompt envoyé à Claude Code, qu'il s'agisse d'un bugfix, d'une feature, ou d'un refactoring. Ces 4 éléments précisent le livrable, le périmètre, la méthode et la vérification ; ils ne garantissent pas une réponse correcte du premier coup :

Élément	Description
WHAT	Livrable concret attendu
WHERE	Chemins des fichiers concernés
HOW	Contraintes techniques, approche souhaitée
VERIFY	Critères de succès mesurables

Exemple :

```
Add input validation to login form.
WHERE: src/components/LoginForm.tsx
HOW: Use Zod, inline errors
VERIFY: Empty email shows error message
```

3.3 Gestion du Contexte

Le contexte est la mémoire de travail de Claude. Les modèles actuels Opus, Sonnet, Haiku et Fable disposent de 1M tokens natifs sur l'API directe ; Haiku 4.5, dernier modèle à 200K, est historique. Les fournisseurs et anciennes variantes peuvent différer. Chaque coéquipier a sa propre fenêtre, dimensionnée par son modèle. Consultez `/context` :

Zone	Contexte	Action
Vert	0-50%	Travailler librement

Zone	Contexte	Action
Jaune	50-70%	Être sélectif
Orange	70-90%	Envisager `/compact` à une frontière de tâche
Rouge	90%+	Sauvegarder les décisions, puis compacter ou préparer une passation

📌 Heures de Pointe (mars 2026)
L'annonce des heures de pointe de mars 2026 est historique, pas une garantie de quota actuel. Consultez `/usage` et les conditions du forfait actif avant de planifier une tâche lourde. Distinguez fenêtre de contexte, limites d'abonnement et facturation API. Référence des limites et coûts.

📌 Mémo: Gérer son contexte en 3 réflexes
Quoi : Garder le contexte pertinent pour la tâche active. Les seuils de pourcentage sont des rappels, pas des garanties de mémorisation.

Quand : Vérifier avec `/context` après chaque tâche majeure, ou quand les réponses deviennent vagues ou répétitives.

Comment : (1) `/compact` à une frontière utile de la tâche ; (2) Démarrer une nouvelle session pour chaque tâche indépendante ; (3) Utiliser `@fichier` pour charger uniquement les fichiers pertinents au lieu de laisser Claude tout scanner.

Vérifier : Si Claude perd une contrainte ou propose une réponse non étayée, rouvrir les preuves pertinentes. Le pourcentage de contexte ne suffit pas à identifier la cause.

Pour aller plus loin : TL;DR: The 5-Minute Summary, résumé visuel complet avec exemples

4 La Série de Whitepapers

Les bases sont posées: installation, premiers workflows, gestion du contexte. Cette introduction fait partie d'une série de 13 livres blancs techniques. Chaque whitepaper approfondit un aspect spécifique, de la rédaction de prompts jusqu'à l'orchestration multi-agents. Vous n'avez pas besoin de tout lire : la section « Parcours Recommandés » en fin de document vous oriente selon votre profil.

WP	Titre	Public	Ce que vous apprenez
#1	Prompts qui Marchent	Tous niveaux	Formuler des demandes précises et vérifiables
#2	Personnaliser Claude	Dev / setup	CLAUDE.md, Skills, Agents réutilisables
#3	Sécurité en Production	DevSecOps, Tech Leads	Permissions, hooks, protection des secrets
#4	L'Architecture Démystifiée	Architectes, curieux	Boucle agent, limites, calibrage des attentes
#5	Déployer en Équipe	Tech Leads, Managers	CI/CD, CLAUDE.md partagé, observabilité
#6	Privacy & Compliance	Conformité, enterprise	Données envoyées, RGPD, opt-out
#7	Le Guide de Référence	Tous	Référence complète + 5 chapitres inédits

4.1 #1 : Prompts qui Marchent (~15 min)

Public : Développeurs de tous niveaux.

- Appliquer la formule WHAT/WHERE/HOW/VERIFY pour définir un résultat vérifiable
- Structurer vos prompts pour que Claude comprenne le contexte sans relecture
- Identifier les anti-patterns (prompts vagues, demandes trop larges) et les corriger

> 📌 **Scénario illustratif : Léa, développeuse junior dans une startup SaaS**
> Léa demande « refactorise le code » et obtient des changements incohérents sur 12 fichiers. Après avoir lu le WP #1, elle reformule : « Extraire la logique de validation de `OrderService.ts` dans un module `validators/order.ts`, conserver les tests existants verts. » Résultat : un diff ciblé, accepté du premier coup.

4.2 #2 : Personnaliser Claude (~20 min)

Public : Développeurs souhaitant optimiser leur setup.

- Créer un `CLAUDE.md` projet efficace avec règles, stack tech et conventions de code

- Configurer des Agents spécialisés (test, review, documentation) réutilisables
- Brancher des Skills pour des connaissances métier persistantes entre sessions

4.3 #3 : Sécurité en Production (~25 min)

Public : DevSecOps, Tech Leads, équipes en environnement sensible.

- Configurer `permissions.deny` pour bloquer les commandes dangereuses en production
- Implémenter des hooks de validation pre/post-exécution pour auditabilité
- Protéger les secrets et clés API contre la fuite involontaire vers le contexte

> 📌 **Scénario : Karim, DevSecOps dans une fintech (50 devs)**
> Karim découvre qu'un développeur a involontairement chargé un `.env` contenant des clés Stripe dans le contexte Claude. Après avoir lu le WP #3, il déploie un hook `pre-tool-use` qui bloque automatiquement la lecture de tout fichier matchant `*.env*` ou `*secret*`. Résultat : les clés ne quittent plus le poste, l'équipe audit garde ses preuves de conformité.

4.4 #4 : L'Architecture Démystifiée (~20 min)

Public : Architectes, développeurs curieux du fonctionnement interne.

- Comprendre la boucle agent (perception → planification → exécution → feedback)
- Savoir pourquoi Claude « oublie » entre les sessions et comment compenser avec CLAUDE.md
- Identifier les limites du système pour calibrer vos attentes et vos prompts

4.5 #5 : Déployer en Équipe (~20 min)

Public : Tech Leads, Managers, équipes en croissance.

- Configurer un CLAUDE.md partagé versionné en Git pour aligner toute l'équipe
- Brancher Claude Code sur votre pipeline CI/CD pour revues et checks automatisés
- Mesurer l'adoption et l'impact avec les outils d'observabilité disponibles

> 📌 **Scénario : Sophie, Tech Lead dans une PME e-commerce (12 devs)**
> Sophie constate que chaque développeur utilise Claude Code différemment : conventions de nommage incohérentes, tests parfois ignorés. Après le WP #5, elle met en place un `CLAUDE.md` partagé dans le monorepo avec les conventions d'équipe, et un agent `reviewer`

dans le pipeline CI. Résultat : les PR générées par Claude respectent les standards dès la première soumission.

4.6 #6 : Privacy & Compliance (~15 min)

Public : Responsables conformité, équipes en entreprise.

- Lister exactement quelles données partent chez Anthropic lors d'une session
- Configurer les règles de rétention et les opt-out disponibles pour votre organisation
- Évaluer la conformité RGPD et les options d'hébergement selon votre secteur

4.7 #7 : Le Guide de Référence (~2h)

Public : Tous profils, document de référence consolidé.

- Retrouver n'importe quelle commande, configuration ou workflow en quelques secondes
- Accéder aux 5 chapitres inédits (Migration, IDE, Git, Patterns, Méthodologies)
- Utiliser comme référence quotidienne : index complet avec numéros de ligne

5 Parcours Recommandés

Il n'existe pas de parcours unique : le bon ordre de lecture dépend de votre rôle, de votre niveau d'expérience, et de la contrainte la plus urgente. En résumé : un dev solo qui débute se concentre sur #0 puis #1, un lead d'équipe attaque par #5 puis #3, et un CTO/décideur priorise #6 puis #5.

Profil	Mon contexte	Parcours conseillé	Temps
Junior	Je découvre Claude Code	#0 → #1	~25 min
Senior	Setup complet + bonnes pratiques	#0 → #1 → #2 → #3	~1h15
Power User	Comprendre l'architecture	#0 → #4 → #3 → #5	~1h15
Tech Lead	Aligner mon équipe	#0 → #5 → #3 → #2	~1h15
PM/Manager	Adoption & ROI	#0 → #5 → #6	~45 min
Compliance	Question RGPD / audit	#0 → #6 → #3	~50 min
Urgence sécurité	Secrets exposés / prod sensible	#3 → #5	~45 min

Alternative : Le **#7 Guide de Référence** (~2h) consolide tous les whitepapers + 5 chapitres inédits (Migration, IDE, Git, Patterns, Méthodologies).

Le **#8 Agent Teams** couvre l'orchestration multi-agents expérimentale (v2.1.32+).

5.1 Solo vs Équipe vs Enterprise

Les besoins de configuration et de gouvernance évoluent avec la taille de l'équipe. Un développeur solo peut se contenter d'un `CLAUDE.md` projet, tandis qu'une équipe de 20+ personnes devra centraliser les permissions, les hooks de sécurité, et les règles de conformité. Ce tableau résume les différences clés pour calibrer votre investissement initial.

Dimension	Dev solo	Équipe (2-20)	Enterprise (20+)
Whitepaper clé	#1, #2	#5, #3	#6, #5, #3
Config minimale	`CLAUDE.md` projet	`CLAUDE.md` partagé en Git	`CLAUDE.md` + permissions centralisées
Sécurité	Bonnes pratiques #3	Hooks CI + deny list	Audit trail + conformité RGPD
CI/CD	Optionnel	Recommandé (agent reviewer)	Obligatoire
Temps de setup	30 min	2-4h	1-2 jours

6 Ressources

Le repository GitHub centralise le guide complet, les templates prêts à l'emploi, et la documentation officielle. C'est le point d'entrée pour approfondir n'importe quel sujet couvert dans cette série.

6.1 Liens essentiels

Ressource	Contenu	Lien
Documentation officielle	Vue d'ensemble Claude Code, installation, référence	docs.anthropic.com/en/docs/claude-code/overview
GitHub Claude Code	Code source, issues, releases officielles	github.com/anthropics/claude-code
Changelog officiel	Historique de toutes les versions et nouveautés	CHANGELOG.md
Guide Ultimate (ce repo)	26 648 lignes, 271 templates, workflows avancés	github.com/FlorianBruniaux/claude-code-ultimate-guide
Cheatsheet imprimable	1 page de référence quotidienne	guide/cheatsheet.md
Model Context Protocol	Spec officielle MCP, registre de serveurs	modelcontextprotocol.io
Anthropic Trust Center	SOC2, GDPR, DPA (pour les équipes compliance)	trust.anthropic.com

6.2 Conventions de Fiabilité

Toutes les informations de cette série ne se valent pas. La documentation officielle Anthropic est considérée comme source de vérité ; les observations issues de tests indépendants ou de retours communautaires sont marquées explicitement avec leur niveau de confiance.

Tier	Source	Fiabilité
Tier 1	Documentation Anthropic	Déclaration officielle ; version et fournisseur à vérifier
Tier 2	Tests vérifiés	Observation limitée aux conditions testées
Tier 3	Inférence communautaire	Hypothèse à valider localement

Les informations Tier 2/3 sont explicitement marquées.

Type	Licence
Contenu	CC BY-SA 4.0
Code	MIT

Version 3.44.1 | Octobre 2026

6.3 Fiche Récap : Introduction à Claude Code

Introduction à Claude Code — Système de contexte, pas chatbot 00

Points à retenir

- Claude Code = **système de contexte** : le contexte, le modèle et les contrôles influencent le résultat. Pas un assistant passif, un agent actif avec accès à vos fichiers, votre terminal, votre git.
- Les **5 Golden Rules** structurent chaque interaction : Be specific, Give context, Think in steps, Verify outputs, Iterate fast.
- La formule **WHAT / WHERE / HOW / VERIFY** précise le livrable, le périmètre et les contrôles : quoi faire, dans quels fichiers, comment vérifier.
- **CLAUDE.md** = mémoire persistante. Ce fichier définit le contexte projet qui sera relu à chaque session.
- Le guide couvre 10 chapitres progressifs, sans obligation de tout lire pour commencer à produire.

Par où commencer ?

1. Installer Claude Code et créer un premier **CLAUDE.md** dans votre projet
2. Appliquer la formule WHAT/WHERE/HOW/VERIFY sur votre prochaine tâche réelle
3. Tester /help pour découvrir les commandes natives
4. Lire le chapitre 2 et vérifier ces concepts sur une tâche réelle

7 La Série

Ce livre blanc fait partie de la série **Claude Code Ultimate Guide**, une ressource complète pour maîtriser Claude Code, de l'installation au déploiement en production.

13 livres blancs dans la série :

- **WP00** Introduction & Fondamentaux
- **WP01** Prompts Efficaces
- **WP02** Personnalisation & Configuration
- **WP03** Sécurité en Production
- **WP04** Architecture Interne
- **WP05** Adoption en Équipe
- **WP06** Privacy & Conformité
- **WP07** Guide de Référence Condensé
- **WP08** Teams d'Agents & Sub-Agents
- **WP09** Apprendre avec l'IA (UVAL)
- **WP10** Budget IA & ROI
- **WP11** Piloter une Équipe à l'Ère de l'IA
- **WP12** L'Ingénierie Logicielle à l'Ère des Agents Autonomes

Série complète disponible sur cc.bruniaux.com/whitepapers/