

Module 1 — Introduction à Claude Code

Jour 1 — Découverte et premiers pas

The Bearded Bear

Février 2026



Table des matières

Module 1 : Introduction à Claude Code 2.1.45	3
Objectifs	3
1. Qu'est-ce que Claude Code?	3
Définition	3
Version actuelle : 2.1.45	3
Positionnement	5
Cas d'usage principaux	5
Modèles disponibles	5
2. Installation	6
Prérequis	6
Installation via NPM	6
Configuration de la clé API	6
Options CLI avancées	6
3. Interface et Commandes	7
Commandes essentielles	7
Nouvelles commandes (2.1.45)	7
Navigation dans le contexte	8
Interaction avec le code	8
4. Plan Mode	8
Qu'est-ce que le Plan Mode?	8
Utilisation	8
Classification Plan Mode dans Claude-Craft	9
Avantages	10
5. Extended Thinking	10
Qu'est-ce que l'Extended Thinking?	10
Hiérarchie des niveaux	10
Utilisation	10
Quand utiliser chaque niveau	11
6. MCP (Model Context Protocol)	11
Qu'est-ce que le MCP?	11
Commande /mcp	11
Variable d'environnement MCP_TIMEOUT	11
Cas d'usage	11
7. Permissions 3-tier	12
Système de permissions	12
Les 3 niveaux	12
Wildcards	12
Configuration	12
Ordre de priorité	13
8. Gestion du Contexte et Tokens	13
Qu'est-ce qu'un token?	13

Limites de contexte	13
Stratégies d'optimisation	13
Estimation des coûts	13
9. Background Tasks	14
Concept	14
Utilisation	14
Cas d'usage	14
Task Management System (v2.1.19+)	14
10. Keybindings	15
Configuration	15
Raccourcis par défaut	15
Personnalisation	16
11. VSCode Integration (v2.1.21+)	16
Python Virtual Environment	16
13. Bonnes Pratiques	17
DO	17
DON'T	17
Exercice Pratique	17
Objectifs de l'exercice	17
Ressources	18
Points Clés à Retenir	18

Module 1 : Introduction à Claude Code 2.1.45

Objectifs

À la fin de ce module, vous serez capable de : - Expliquer ce qu'est Claude Code et ses différences avec d'autres outils - Installer et configurer Claude Code - Utiliser les commandes de base et avancées (plan mode, extended thinking, MCP, permissions) - Comprendre la gestion du contexte et des tokens

1. Qu'est-ce que Claude Code ?

Définition

Claude Code est l'interface en ligne de commande (CLI) officielle d'Anthropic pour interagir avec Claude, l'assistant IA. C'est un outil de développement assisté par IA conçu pour le pair programming et les tâches de codage.

Version actuelle : 2.1.45

Nouveautés majeures incluent (par ordre de version) :

Fonctionnalités établies

- **Extended Thinking** : hiérarchie think < think hard < think harder < ultrathink
- **MCP Integration** : Model Context Protocol, /mcp, MCP_TIMEOUT
- **13 Hooks** : PreToolUse, PostToolUse, PostToolUseFailure, PermissionRequest, UserPrompt-Submit, Stop, SubagentStop, SubagentStart, Notification, PreCompact, SessionStart, SessionEnd, Setup
- **Permissions 3-tier** : Deny > Allow > Ask, wildcards
- **Sub-agents** : Task tool, parallel execution
- **Plan Mode** pour explorer sans exécuter
- **Background Tasks** pour les opérations longues
- **Commandes** : /compact, /doctor, /mcp, /teleport, /release-notes, /config, /keybindings, /tasks
- **Raccourcis** : Shift+Enter (newline), Ctrl+B (background), Shift+Tab (plan mode)
- **Flag --add-dir** : Charger des CLAUDE.md depuis des répertoires additionnels
- **Modèle par défaut** : Sonnet 4.5

Nouveautés v2.1.20 - v2.1.31

- **Task status deleted** (v2.1.20) : Suppression définitive de tâches via TaskUpdate
- **Background Agent Permissions** (v2.1.20) : Les agents demandent les permissions AVANT le lancement
- **File Tools Preference** (v2.1.21) : Claude préfère Read/Edit/Write aux équivalents bash

- **VSCode Python venv** (v2.1.21) : Setting `claudeCode.usePythonEnvironment`
- **spinnerVerbs** (v2.1.23) : Configuration personnalisable du texte spinner
- **PR Integration** (v2.1.27) : `--from-pr`, auto-link, indicateurs de statut PR

Nouveautés v2.1.30

- **PDF Page Range** : Paramètre pages pour Read tool sur les PDFs, référence légère pour les PDFs >10 pages
- **OAuth Client Credentials MCP** : Flags `--client-id` / `--client-secret` pour `claude mcp add`
- **/debug** : Commande de troubleshooting de session (complète /doctor)
- **Task Tool Metrics** : Token count, tool uses, duration dans les résultats du Task tool
- **Reduced Motion Mode** : Option `reducedMotion: true` pour minimiser les animations

Nouveautés v2.1.31

- **Session Resume Hint** : Hint de reprise de session affiché à la sortie de Claude Code
- **PDF Limits Clarification** : Messages d'erreur affichant les vraies limites (100 pages, 20MB)
- **Enhanced File Tools Preference** : System prompts renforcés pour guider vers les outils natifs (Read, Edit, Glob, Grep)
- **Reduced Layout Jitter** : Moins de sautilllements UI du spinner dans le terminal
- **Japanese IME Support** : Support de l'espace pleine largeur (zenkaku) dans les checkboxes
- **Third-party Pricing Fix** : Tarification Anthropic supprimée du sélecteur pour les utilisateurs Bedrock/Vertex/Foundry

Nouveautés v2.1.32

- **Claude Opus 4.6** : Nouveau modèle flagship - 200K context (1M beta), 128K output, adaptive thinking
- **Agent Teams** (Research Preview) : Coordination multi-agents avec Teammate/SendMessage tools, tâches partagées
- **Automatic Memory Recording** : Claude enregistre automatiquement la mémoire de session après ~10K tokens
- **Summarize from Here** : Résumé partiel de conversation depuis un point spécifique
- **Auto Skill Loading** : Skills depuis `--add-dir` auto-découverts
- **Skill Character Budget Scaling** : Budget skills = 2% de la fenêtre de contexte du modèle
- **-resume Agent Inheritance** : `--agent` auto-hérité lors de `--resume`
- **VSCode Session Loading Spinner** : Spinner de chargement pendant restauration de session

Nouveautés v2.1.33

- **Agent Memory Frontmatter** : Champ memory (user/project/local) pour la mémoire persistante des agents
- **TeammateIdle & TaskCompleted Hooks** : 2 nouveaux événements hooks pour workflows multi-agents (11 → 13 hooks)

- **Agent Type Restrictions** : Syntaxe Task (agent_type) pour contrôler les sous-agents autorisés
- **Plugin Name in Skills** : Nom du plugin visible dans le menu /skills
- **VSCode Remote Sessions** : Browse/resume depuis claude.ai
- **VSCode Session Picker** : Branche git et nombre de messages affichés

Nouveautés v2.1.36-v2.1.45

- **Fast Mode** (v2.1.36) : /fast pour Opus 4.6 jusqu'à 2.5x plus rapide, meme intelligence
- **Skills Directory Protection** (v2.1.45) : Ecritures dans .claude/skills bloquées en sandbox
- **Heredoc JS Fix** (v2.1.45) : Plus d'erreur "Bad substitution" avec les template literals JS
- **Plan Mode Crash Fix** (v2.1.45) : Correction crash avec config projet dans ~/.claude.json
- **temperatureOverride Fix** (v2.1.45) : temperatureOverride fonctionne maintenant dans le streaming API
- **VSCode Fixes** (v2.1.37/v2.1.45) : Scroll, Tab key, duplicate sessions
- **LSP Compatibility** (v2.1.45) : Compatibilité avec les serveurs LSP stricts

Positionnement

Outil	Type	Force	Limitation
Claude Code	CLI + Agent	Contexte large, hooks, plan mode	Pas d'IDE intégré natif
GitHub Copilot	Extension IDE	Autocomplétion rapide	Contexte limité
ChatGPT	Web/API	Polyvalent	Pas spécialisé code
Cursor	IDE complet	Intégration UI	Dépendance à l'IDE

Cas d'usage principaux

1. **Pair Programming** : Claude comme partenaire de développement
2. **Génération de code** : Scaffolding, boilerplate, features
3. **Refactoring** : Amélioration de code existant
4. **Debug** : Analyse et résolution de bugs
5. **Documentation** : Génération de docs, commentaires
6. **Code Review** : Revue assistée par IA
7. **Tâches autonomes** : Via background tasks et agents

Modèles disponibles

Modèle	Caractéristique	Usage recommandé
Claude Sonnet 4.5	Rapide, modèle par défaut	Usage quotidien
Claude Opus 4.6	Flagship, 1M context (beta), 128K output, adaptive thinking	Tâches complexes, agents
Claude Opus 4.5	Puissant, raisonnement avancé	Tâches complexes (legacy)
Claude Haiku	Léger, économique	Tâches simples, background

2. Installation

Prérequis

- Node.js 20+ installé
- Compte Anthropic avec clé API
- Terminal bash/zsh

Installation via NPM

```
# Installation globale
npm install -g @anthropic-ai/claude-code

# Vérification
claude --version
# Devrait afficher : 2.1.45 ou supérieur
```

Configuration de la clé API

```
# Option 1 : Variable d'environnement (recommandé)
export ANTHROPIC_API_KEY="sk-ant-..."

# Option 2 : Fichier de configuration
claude config set api_key sk-ant-...

# Vérification
claude config get api_key
```

Options CLI avancées

```
# Lancer Claude dans le répertoire courant
cd mon-projet
```

claude

Claude va indexer le projet et démarrer une session interactive

Charger des CLAUDE.md depuis des répertoires additionnels (v2.1.20+)

claude **--add-dir** /chemin/vers/autre-projet

Utile pour partager des instructions entre plusieurs projets

3. Interface et Commandes

Commandes essentielles

Commande	Description
/help	Affiche l'aide complète
/clear	Efface le contexte
/model <nom>	Change de modèle (sonnet, opus, haiku)
/exit ou /quit	Quitte Claude
/cost	Affiche le coût de la session
/history	Historique des conversations

Nouvelles commandes (2.1.45)

Commande	Description
/plan	Active le mode exploration (sans exécution)
/compact	Compacter le contexte
/doctor	Diagnostiquer les problèmes
/debug	Troubleshoot session en cours (v2.1.30+)
/mcp	Gérer les serveurs MCP
/config	Configuration
/teleport	Téléporter la session
/release-notes	Notes de version
/skills	Liste les skills disponibles
/keybindings	Personnaliser les raccourcis clavier

Commande	Description
/tasks	Système de gestion de tâches (Task Management)

Navigation dans le contexte

```
# Ajouter un fichier au contexte
/add src/Controller/UserController.php

# Ajouter un répertoire
/add src/Entity/

# Lister les fichiers en contexte
/context

# Retirer un fichier
/remove src/Controller/UserController.php
```

Interaction avec le code

```
# Demander une modification
"Ajoute une méthode getFullName() à l'entité User"

# Demander une explication
"Explique comment fonctionne ce repository"

# Demander une génération
"Crée un service pour gérer l'envoi d'emails"

# Utiliser un skill
/testing "Écris les tests pour OrderService"
```

4. Plan Mode

Qu'est-ce que le Plan Mode ?

Le **Plan Mode** permet à Claude d'explorer et d'analyser sans exécuter de modifications. Idéal pour : - Comprendre un codebase - Planifier une refactorisation - Évaluer les impacts d'un changement

Utilisation

```
# Activer le mode plan
/plan
```

```
# Ou via le raccourci clavier
# Shift+Tab pour basculer en plan mode

# Claude peut maintenant :
# - Lire les fichiers
# - Analyser le code
# - Proposer des plans
# - MAIS ne peut PAS modifier de fichiers

# Exemple d'utilisation
"Analyse le module Order et propose un plan de migration vers Clean
  ↳ Architecture"

# Claude répond avec un plan détaillé sans modifier de code

# Quitter le mode plan pour exécuter
/plan off
# Ou accepter le plan proposé
```

Classification Plan Mode dans Claude-Craft

Les commandes Claude-Craft incluent une guidance Plan Mode intégrée qui classifie automatiquement le niveau requis :

Niveau	Description	Exemples
MANDATORY	Plan mode s'active automatiquement avant l'exécution. Claude analyse le code impacté et propose un plan à valider.	/workflow:implement, /qa:tdd, /qa:recette, commandes generate-*
RECOMMENDED	Plan mode recommandé pour les scénarios complexes. S'active quand le scope couvre plusieurs modules.	/workflow:plan, /workflow:design, /common:architecture- decision
CONDITIONAL	Plan mode s'active automatiquement quand le scope est large (analyse multi-modules).	Commandes check-*, /workflow:analyze, /common:research- context7

Astuce : Pas besoin de retenir ces règles — chaque commande indique son niveau de Plan Mode dans sa documentation intégrée.

Avantages

1. **Sécurité** : Pas de modification accidentelle
 2. **Compréhension** : Explorer avant d'agir
 3. **Collaboration** : Valider le plan avec l'équipe
 4. **Coût** : Moins de tokens consommés (pas de génération de code)
-

5. Extended Thinking

Qu'est-ce que l'Extended Thinking ?

L'**Extended Thinking** permet de demander à Claude un raisonnement plus profond et plus long avant de répondre. Cela se traduit par des réponses mieux réfléchies, notamment pour les problèmes complexes.

Hiérarchie des niveaux

Niveau	Commande	Profondeur	Usage recommandé
1	think	Réflexion basique	Questions simples nécessitant un peu de recul
2	think hard	Réflexion approfondie	Problèmes moyennement complexes
3	think harder	Réflexion poussée	Architecture, design patterns complexes
4	ultrathink	Réflexion maximale	Problèmes critiques, debugging profond

Utilisation

Il suffit de préfixer votre prompt avec le niveau de réflexion souhaité :

Niveau 1 : Réflexion basique

"think - Comment nommer cette variable ?"

Niveau 2 : Réflexion approfondie

"think hard - Quelle architecture choisir pour ce microservice ?"

Niveau 3 : Réflexion poussée

"think harder - Analyse les implications de sécurité de cette migration
↪ OAuth"

```
# Niveau 4 : Réflexion maximale
"ultrathink - Identifie la cause racine de ce bug de concurrence"
```

Quand utiliser chaque niveau

- **think** : Nommage, choix simples, reformulations
 - **think hard** : Design d'API, choix d'architecture entre 2-3 options
 - **think harder** : Refactoring complexe, analyse de sécurité, migration de données
 - **ultrathink** : Bugs de concurrence, optimisation de performance critique, architecture distribuée
-

6. MCP (Model Context Protocol)

Qu'est-ce que le MCP ?

Le **Model Context Protocol (MCP)** est un protocole ouvert qui permet à Claude Code de se connecter à des serveurs externes fournissant des outils, des ressources et des données supplémentaires. Cela étend les capacités de Claude au-delà de son contexte local.

Commande /mcp

```
# Lister les serveurs MCP configurés
/mcp

# Ajouter un serveur MCP
/mcp add <nom-du-serveur>

# Voir les outils disponibles via MCP
/mcp status
```

Variable d'environnement MCP_TIMEOUT

```
# Configurer le timeout des serveurs MCP (en millisecondes)
export MCP_TIMEOUT=30000 # 30 secondes

# Utile quand les serveurs MCP sont lents ou distants
```

Cas d'usage

- **Accès aux bases de données** : Interroger directement une BDD via un serveur MCP
- **Intégration navigateur** : Claude in Chrome pour tester des applications web

- **Outils externes** : Connecter des outils spécifiques (Jira, Notion, Slack, etc.)
 - **Ressources distantes** : Accéder à de la documentation, des API, des fichiers distants
-

7. Permissions 3-tier

Système de permissions

Claude Code 2.1.45 introduit un système de permissions à 3 niveaux pour contrôler finement les actions que Claude peut exécuter.

Les 3 niveaux

Niveau	Description	Comportement
Deny	Interdit	L'action est bloquée, Claude ne peut pas l'exécuter
Allow	Autorisé	L'action est exécutée automatiquement sans confirmation
Ask	Demander	Claude demande confirmation avant d'exécuter

Wildcards

Les permissions supportent les wildcards pour des règles plus flexibles :

Autoriser toutes les commandes avec le flag -h (help)

Bash(*-h*) -> Allow

Bloquer toutes les commandes rm

Bash(*rm*) -> Deny

Demander confirmation pour les commandes git push

Bash(*git push*) -> Ask

Configuration

Les permissions se configurent dans les settings de Claude Code :

Voir les permissions actuelles

/config permissions

```
# Les permissions sont définies dans :  
# ~/.claude/settings.json (global)  
# .claude/settings.json (projet)
```

Ordre de priorité

1. **Deny** est toujours prioritaire : si une règle Deny correspond, l'action est bloquée
2. **Allow** autorise sans confirmation
3. **Ask** (défaut) demande confirmation à l'utilisateur

8. Gestion du Contexte et Tokens

Qu'est-ce qu'un token ?

- Unité de texte (mot, ponctuation, partie de mot)
- ~4 caractères = 1 token (en anglais)
- ~3 caractères = 1 token (en français, code)

Limites de contexte

Modèle	Contexte max	Recommandé
Sonnet 4.5	200K tokens	< 150K
Opus 4.6	200K (1M beta)	< 150K
Opus 4.5	200K tokens	< 150K
Haiku	200K tokens	< 100K

Stratégies d'optimisation

1. **Être sélectif** : N'ajouter que les fichiers pertinents
2. **Nettoyer régulièrement** : /clear quand le contexte devient trop grand
3. **Compacter** : /compact pour résumer le contexte sans le perdre
4. **Background tasks** : Décharger les tâches longues

Estimation des coûts

```
# Afficher le coût actuel  
/cost
```

```
# Tarification indicative (2026)
# Sonnet 4.5 : ~$3/million tokens input, ~$15/million tokens output
# Opus 4.6 : ~$5/million tokens input, ~$25/million tokens output
# Opus 4.5 : ~$15/million tokens input, ~$75/million tokens output
# Haiku : ~$0.25/million tokens input, ~$1.25/million tokens output
```

9. Background Tasks

Concept

Les **Background Tasks** permettent de lancer des opérations longues sans bloquer la session interactive.

Utilisation

```
# Lancer une tâche en background via le raccourci
# Ctrl+B pour envoyer une tâche en arrière-plan

# Lancer une tâche en background
/tasks run "Analyse tous les fichiers PHP et génère un rapport de qualité"

# Lister les tâches en cours
/tasks list

# Voir le statut d'une tâche
/tasks status <task-id>

# Récupérer le résultat
/tasks output <task-id>

# Arrêter une tâche
/tasks stop <task-id>
```

Cas d'usage

- Analyse de codebase complet
- Génération de documentation exhaustive
- Refactoring multi-fichiers
- Migration de code

Task Management System (v2.1.19+)

Claude Code 2.1.19 introduit un système complet de gestion de tâches pour organiser et suivre les opérations complexes multi-étapes.

Outils disponibles

Outil	Description
TaskCreate	Créer une tâche avec sujet, description et activeForm
TaskGet	Récupérer les détails complets d'une tâche
TaskUpdate	Mettre à jour le statut, les dépendances, le propriétaire
TaskList	Lister toutes les tâches avec leur statut

Cycle de vie d'une tâche

pending → in_progress → completed

↓
deleted

Gestion des dépendances Les tâches supportent les dépendances via `blocks` et `blockedBy` : - Une tâche bloquée ne peut pas démarrer tant que ses dépendances ne sont pas résolues - Permet de structurer des plans d'implémentation complexes

Cas d'usage

- **Planification multi-étapes** : Décomposer une feature en tâches ordonnées
- **Suivi de progression** : Visualiser l'avancement en temps réel
- **Orchestration parallèle** : Identifier les tâches exécutables simultanément

10. Keybindings

Configuration

Les raccourcis clavier sont personnalisables dans `~/.claude/keybindings.json`.

Raccourcis par défaut

Raccourci	Action
Ctrl+C	Annuler la génération
Ctrl+D	Quitter
↑ / ↓	Navigaton historique

Raccourci	Action
Tab	Autocomplétion
Ctrl+L	Effacer l'écran
Shift+Enter	Nouvelle ligne (saisie multi-lignes)
Ctrl+B	Envoyer en tâche de fond (background)
Shift+Tab	Basculer en mode plan

Personnalisation

Depuis la v2.1.20, la commande `/keybindings` permet de configurer les raccourcis directement dans Claude Code.

```
// ~/.claude/keybindings.json
{
  "bindings": {
    "ctrl+s": "submit",
    "ctrl+p": "plan",
    "ctrl+t": "tasks",
    "ctrl+shift+c": "clear"
  }
}
```

11. VSCode Integration (v2.1.21+)

Python Virtual Environment

Le setting `claudeCode.usePythonEnvironment` permet d'activer automatiquement l'environnement virtuel Python détecté par VSCode.

Configuration Dans les settings VSCode (`.vscode/settings.json`):

```
{
  "claudeCode.usePythonEnvironment": true
}
```

Comportement Quand activé : - Claude Code détecte automatiquement le venv actif dans VSCode
- Les commandes Python s'exécutent dans le bon environnement - Plus besoin de prefixer avec `source venv/bin/activate`

Cas d'usage

- Projets Python avec dépendances spécifiques
 - Multi-projets avec différentes versions de packages
 - Environnements conda ou poetry
-

13. Bonnes Pratiques**DO**

- Donner du contexte clair dans vos demandes
- Utiliser le Plan Mode avant les gros changements
- Utiliser Extended Thinking pour les problèmes complexes
- Valider le code généré avant de l'utiliser
- Nettoyer le contexte régulièrement avec `/compact`
- Spécifier le langage/framework quand pertinent
- Utiliser les hooks pour automatiser
- Configurer les permissions pour sécuriser les actions sensibles

DON'T

- Copier-coller aveuglément le code généré
 - Surcharger le contexte avec des fichiers inutiles
 - Ignorer les avertissements de Claude
 - Oublier de tester le code généré
 - Partager des données sensibles (credentials, etc.)
-

Exercice Pratique

Voir `exercices/exercice-01-premier-projet.md`

Objectifs de l'exercice

1. Installer Claude Code sur votre machine
 2. Configurer la clé API
 3. Lancer une première session
 4. Utiliser le Plan Mode
 5. Tester l'Extended Thinking avec différents niveaux
 6. Explorer les commandes de base et avancées
-

Ressources

- Documentation officielle Claude Code
 - Anthropic Cookbook
 - Pricing Anthropic
 - Release Notes 2.1.45
-

Points Clés à Retenir

1. **Claude Code 2.1.45** = CLI officiel avec Extended Thinking, MCP, Permissions 3-tier, hooks, plan mode, /debug, Agent Teams
 2. **4 modèles** : Sonnet 4.5 (par défaut), Opus 4.6 (flagship), Opus 4.5 (puissant), Haiku (léger)
 3. **Extended Thinking** : 4 niveaux (think < think hard < think harder < ultrathink)
 4. **MCP** : Protocole ouvert pour connecter des outils et ressources externes
 5. **Permissions 3-tier** : Deny > Allow > Ask avec wildcards
 6. **Plan Mode** : Explorer et planifier sans exécuter
 7. **Background Tasks** : Tâches longues sans bloquer
 8. **Hooks** : Automatisation via 13 événements
 9. **Contexte** : Gérer activement pour optimiser coûts et pertinence
-

Durée estimée : 1h30 **Prochain module** : Le Framework Claude-Craft