Développement Assisté par IA

DE L'IDÉE À LA PRODUCTION

Le guide complet tout-en-un pour créer votre propre site web, application ou startup à partir de zéro. Maîtrisez les agents et outils IA, intégrez les paiements, déployez votre serveur, et apprenez les techniques marketing pour croître de manière organique.

apifreellm.com

Version 1.0 — Février 2026

Table des Matières

1. Introduction

> *Vous lisez ceci en ce moment parce que quelque part, d'une manière ou d'une autre, une combinaison de stratégies marketing, de techniques SEO et de positionnement intelligent a amené ce cours sur votre écran. Cela seul devrait vous dire quelque chose : les tactiques de ce guide fonctionnent vraiment. Et oui, vous allez toutes les apprendre.*

Le monde a changé

Le développement logiciel n'est plus ce qu'il était. Il y a quelques années, créer une application web nécessitait des mois de travail, une équipe de développeurs et un budget conséquent. Aujourd'hui, une seule personne avec les bons outils et les bonnes connaissances peut construire et lancer une application prête pour la production en quelques jours, pas en quelques mois.

Ce cours a été écrit par des professionnels qui travaillent dans l'industrie et utilisent quotidiennement des outils IA agentiques pour construire de vrais produits. Nous n'enseignons pas la théorie tirée d'un manuel. Nous enseignons ce qui fonctionne réellement dans le monde réel, en ce moment.

Pourquoi ce cours existe

L'IA et les LLM sont désormais de meilleurs et plus rapides exécutants que les humains. Ils peuvent écrire du code, le déboguer, le refactoriser et le déployer à une vitesse qu'aucun humain ne peut égaler. Mais il y a quelque chose qui leur manque fondamentalement : les **idées**.

> Les modèles d'IA sont des exécutants extraordinaires, mais ils ne sont pas des innovateurs. Ils ne voient pas un manque sur le marché et ne se disent pas « je pourrais construire quelque chose pour résoudre ça. » C'est votre rôle. Votre mission est d'être l'architecte des idées. L'IA est votre constructeur.

Quand vous donnez de mauvaises instructions à un LLM, il produira quand même quelque chose. Il ne s'arrêtera pas pour expliquer ce qui serait réellement mieux. La qualité de ce que vous construisez est directement proportionnelle à la qualité de vos connaissances. Ce cours comble cette lacune.

Ce n'est pas seulement un cours de développement. C'est un plan d'action complet pour passer d'une idée à un produit en ligne qui génère des revenus. Y compris des tactiques marketing et SEO extrêmement efficaces — les mêmes techniques qui vous ont amené sur cette page.

2. La Stack de Développement IA-First

> *Les outils que vous choisissez définiront la vitesse à laquelle vous avancez. Dans ce chapitre, nous décortiquons les agents de codage IA disponibles aujourd'hui, nous vous aidons à choisir le bon, et nous vous enseignons les stratégies de prompting qui séparent les amateurs des professionnels.*

> **Note :** Ce guide est écrit sous Windows, mais tous les outils et étapes fonctionnent de manière identique sur macOS et Linux. Google Antigravity, Claude Code, et toutes les autres applications abordées dans ce cours sont entièrement multiplateformes. Si vous êtes sur Mac ou Linux, suivez simplement les mêmes étapes — les interfaces et les flux de travail sont les mêmes.

Le paysage des agents IA

L'espace des outils de codage IA a explosé. Mais tous les outils ne se valent pas. Certains ne sont que des moteurs d'autocomplétion améliorés. D'autres sont de véritables agents autonomes capables de lire l'intégralité de votre base de code, de planifier une stratégie, d'exécuter des modifications sur plusieurs fichiers, de lancer des tests et de corriger les erreurs par eux-mêmes. Comprendre la différence est crucial.

Il existe deux catégories fondamentales d'outils de codage IA :

- **Les assistants inline** — Ils se placent dans votre éditeur et suggèrent du code pendant que vous tapez. Pensez au GitHub Copilot original. Ils sont réactifs : ils attendent que vous écriviez, puis essaient de deviner ce qui vient ensuite. Utiles, mais limités.
- **Les outils agentiques** — Ceux-ci sont d'une tout autre nature. Vous leur donnez une tâche en langage naturel, et ils planifient, écrivent, éditent, déboguent et itèrent de manière autonome. Ils ne suggèrent pas simplement une ligne de code. Ils construisent des fonctionnalités. C'est là que réside la vraie puissance.

Voici les outils qui comptent en ce moment :

Claude Code (par Anthropic)

Claude Code est, d'après notre expérience, l'agent de codage IA le plus puissant disponible aujourd'hui. C'est un agent en terminal qui opère directement dans le répertoire de votre projet. Vous lui donnez des instructions en langage naturel, et il lit vos fichiers, écrit du code, exécute des commandes, crée des commits et corrige les erreurs de manière autonome. Il a un accès complet à votre système de fichiers et à votre shell, ce qui le rend incroyablement efficace pour le vrai travail de développement. Vous pouvez l'installer via npm (`npm install -g @anthropic-ai/claude-code`) ou l'utiliser comme extension VS Code, ce que nous aborderons sous peu.

Google Antigravity

Antigravity est un environnement de développement de Google qui intègre le chat IA et les capacités agentiques directement dans votre flux de travail. Imaginez-le comme un espace de travail intelligent où vous pouvez interagir avec des agents IA via des interfaces de chat, et depuis chaque conversation d'agent vous pouvez ouvrir un éditeur VS Code intégré. C'est essentiel : Antigravity vous offre la couche d'IA conversationnelle, tandis que VS Code fournit la puissance d'édition de code. La combinaison est fluide — vous discutez de ce que vous voulez construire avec l'agent, puis vous plongez directement dans le code sans changer de fenêtre.

Cursor

Cursor est un fork de VS Code avec l'IA profondément intégrée. Il propose à la fois des suggestions inline et un mode agentique « Composer » où vous pouvez décrire des modifications sur plusieurs fichiers. L'interface est familière si vous utilisez déjà VS Code, et il supporte plusieurs modèles (Claude, GPT, etc.). C'est un outil solide, surtout si vous préférez un flux de travail entièrement visuel. Cependant, ses capacités agentiques, bien que bonnes, ne sont pas aussi autonomes que Claude Code. Vous devrez souvent examiner et approuver les modifications individuellement, étape par étape.

> Notre configuration recommandée : **Google Antigravity + Claude Code comme extension VS Code.** Utilisez les chats d'agents d'Antigravity pour planifier et discuter de votre travail, puis ouvrez le VS Code intégré et laissez Claude Code gérer l'exécution lourde. Cela vous donne le meilleur des deux mondes : une conversation intelligente pour la planification, et une exécution de code autonome pour la construction.

Choisir & configurer votre agent

Installer et lancer un outil agentique est la partie facile. En tirer le meilleur parti nécessite de comprendre son fonctionnement, de choisir le bon abonnement et de le configurer correctement.

L'abonnement Claude : commencez par Pro

Claude Code nécessite un compte Anthropic. Nous recommandons de commencer avec l'**abonnement Claude Pro**, qui est le premier palier payant. Il est abordable et vous donne accès à Claude Code avec toutes ses fonctionnalités. C'est le point de départ parfait pour expérimenter, apprendre le flux de travail et construire votre premier projet simple.

Sachez cependant que le plan Pro a une **utilisation limitée**. Si vous utilisez Claude Code de manière intensive, vous atteindrez le plafond d'utilisation relativement vite. Pour l'apprentissage et les petits projets, c'est largement suffisant. Mais si vous savez déjà que vous voulez utiliser Claude Code comme outil de développement principal et prévoyez d'y travailler plusieurs heures par jour, envisagez de passer à l'un des plans supérieurs (comme Max) dès le départ. Les plans supérieurs offrent une utilisation nettement plus élevée, ce qui signifie moins d'interruptions et un flux de travail plus fluide pour les projets plus importants.

Le modèle : Claude Opus 4.6

Nous recommandons actuellement d'utiliser **Claude Opus 4.6** comme modèle de développement principal. C'est, à l'heure actuelle, le meilleur modèle pour construire des sites web et des applications. Il comprend les architectures complexes, écrit du code propre et prêt pour la production, gère les modifications multi-fichiers avec une précision remarquable, et a rarement besoin de corrections dès le premier essai. Quand vous travaillez avec Claude Code, définissez Opus 4.6 comme modèle par défaut. La différence de qualité de sortie par rapport aux modèles plus petits est immédiatement perceptible.

Configurer Antigravity

À partir de ce point, ce guide se poursuit en utilisant **Google Antigravity** comme environnement de développement principal. Voici comment tout préparer.

Étape 1 : Créez votre dossier de projet. Avant d'ouvrir Antigravity, créez un dossier sur votre ordinateur où vivra votre premier projet. Par exemple, sous Windows vous pourriez créer `C:\Projects\my-first-app`, ou sur Mac/Linux `~/Projects/my-first-app`. C'est le dossier qu'Antigravity ouvrira comme espace de travail. Il peut être vide pour l'instant — nous le remplirons de code plus tard dans le cours.

Étape 2 : Installez et lancez Antigravity. Téléchargez Google Antigravity, installez-le et ouvrez-le. Connectez-vous avec votre compte Google quand demandé.

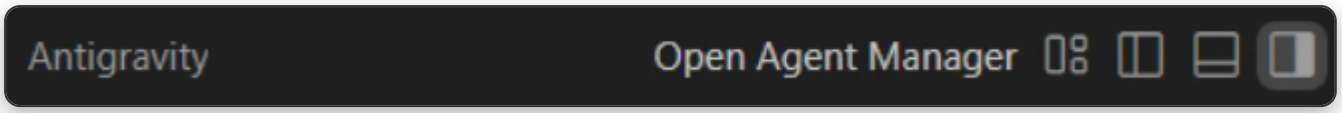
Pendant le processus d'installation, Antigravity vous demandera de configurer quelques politiques. Pour un flux de développement rapide et autonome, nous recommandons de sélectionner **configuration personnalisée** et de définir ces options :

- **Politique d'exécution du terminal** → Toujours continuer
- **Politique de révision** → Toujours continuer
- **Exécution JavaScript** → Toujours continuer

Avec ces paramètres, les agents d'Antigravity pourront exécuter des commandes, écrire des fichiers et exécuter du code de manière autonome sans demander de confirmation à chaque étape. C'est ce qui permet l'expérience de développement rapide et fluide que nous visons dans ce cours.

> **Avertissement de sécurité :** Quand vous autorisez les agents à procéder de manière autonome, Antigravity et Claude Code peuvent exécuter des actions sur votre système sans approbation manuelle. Cela signifie que vous devez être attentif à ce que vous leur exposez. Ne pointez pas les agents vers des bases de code susceptibles de contenir des malwares, et soyez prudent avec les liens ou ressources provenant de sources non fiables. À l'ère de l'IA, il existe de nouveaux vecteurs d'attaque — des acteurs malveillants peuvent créer du contenu spécialement conçu pour tromper les LLM et les amener à exécuter des commandes nuisibles. Gardez toujours un œil sur ce que font vos agents. Nous recommandons la configuration « Toujours continuer » pour la rapidité, mais restez vigilant et examinez régulièrement les résultats.

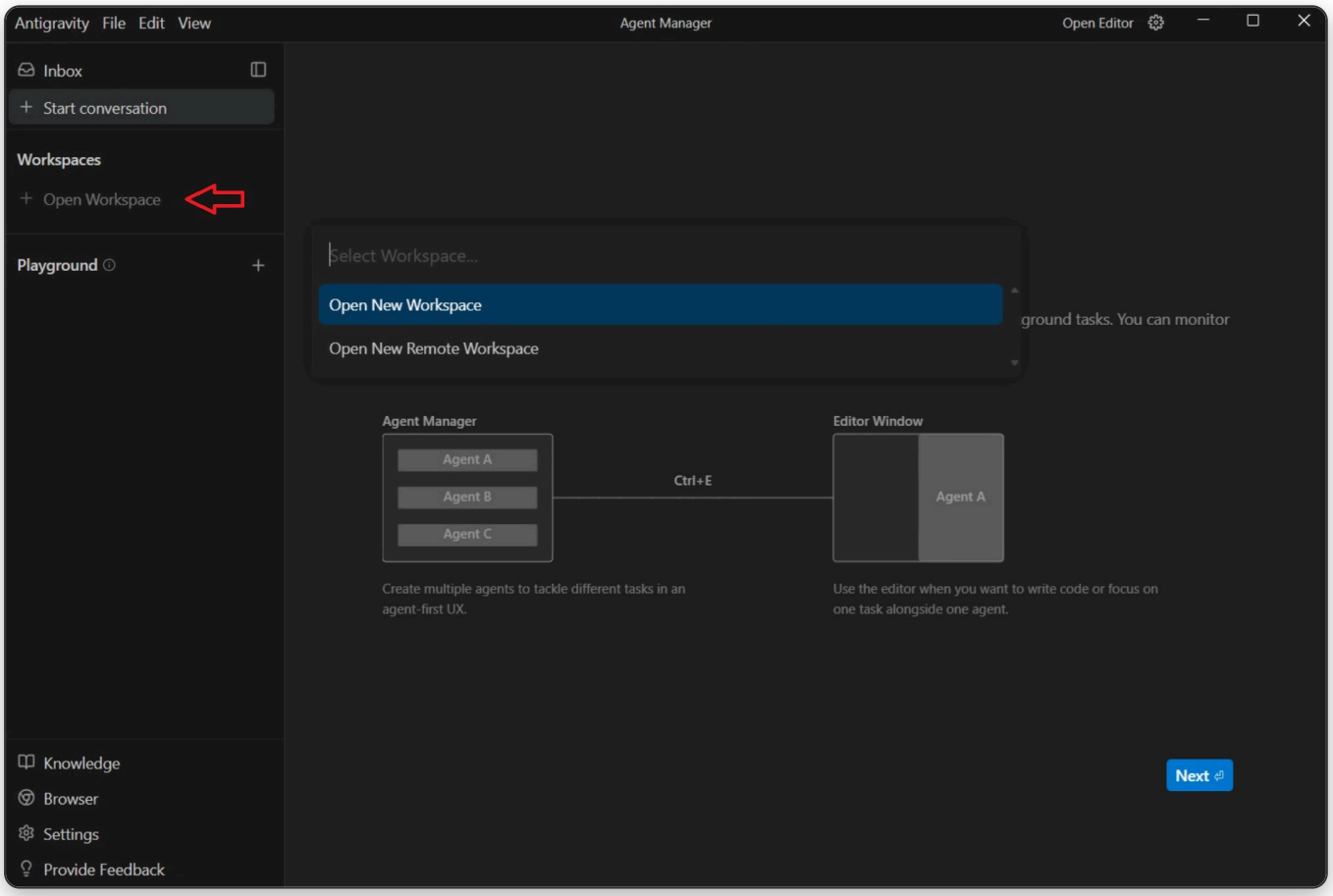
Étape 3 : Ouvrez le Gestionnaire d'agents. Une fois Antigravity lancé, cliquez sur **« Open Agent Manager »** dans la barre supérieure de la fenêtre.



Le bouton « Open Agent Manager » dans la barre supérieure d'Antigravity.

Le **Gestionnaire d'agents** est le centre névralgique d'Antigravity. C'est là que vous gérez vos projets, démarrez des conversations IA et ouvrez vos espaces de travail de développement.

Étape 4 : Créez votre premier espace de travail. Dans le Gestionnaire d'agents, regardez la barre latérale gauche. Sous la section **« Workspaces »**, cliquez sur **« + Open Workspace »**. Un menu déroulant apparaîtra — sélectionnez **« Open New Workspace »**.

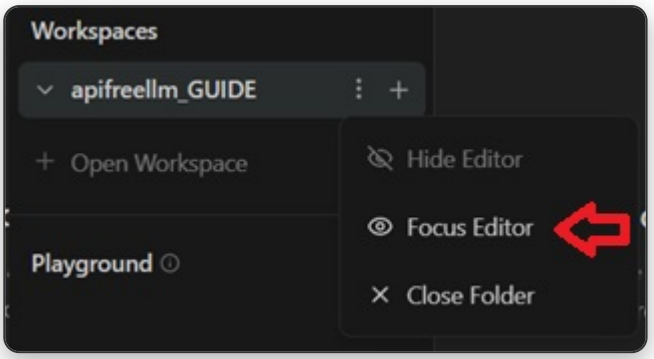


Cliquez sur « + Open Workspace » dans la barre latérale gauche, puis sélectionnez « Open New Workspace ».

Antigravity vous demandera de sélectionner un dossier. Naviguez jusqu'au dossier de projet que vous avez créé à l'Étape 1 et sélectionnez-le. Votre nouvel espace de travail apparaîtra dans la barre latérale gauche sous « Workspaces », avec le nom du dossier que vous avez sélectionné. Considérez chaque espace de travail comme une session de développement individuelle — un par projet. Vous pouvez créer autant d'espaces de travail que nécessaire et passer de l'un à l'autre depuis le Gestionnaire d'agents à tout moment.

Étape 5 : Ouvrez l'éditeur. Une fois votre espace de travail créé, vous verrez son nom dans la barre latérale. Cliquez sur les **trois points verticaux** (⋮) à côté du nom de l'espace de travail, puis

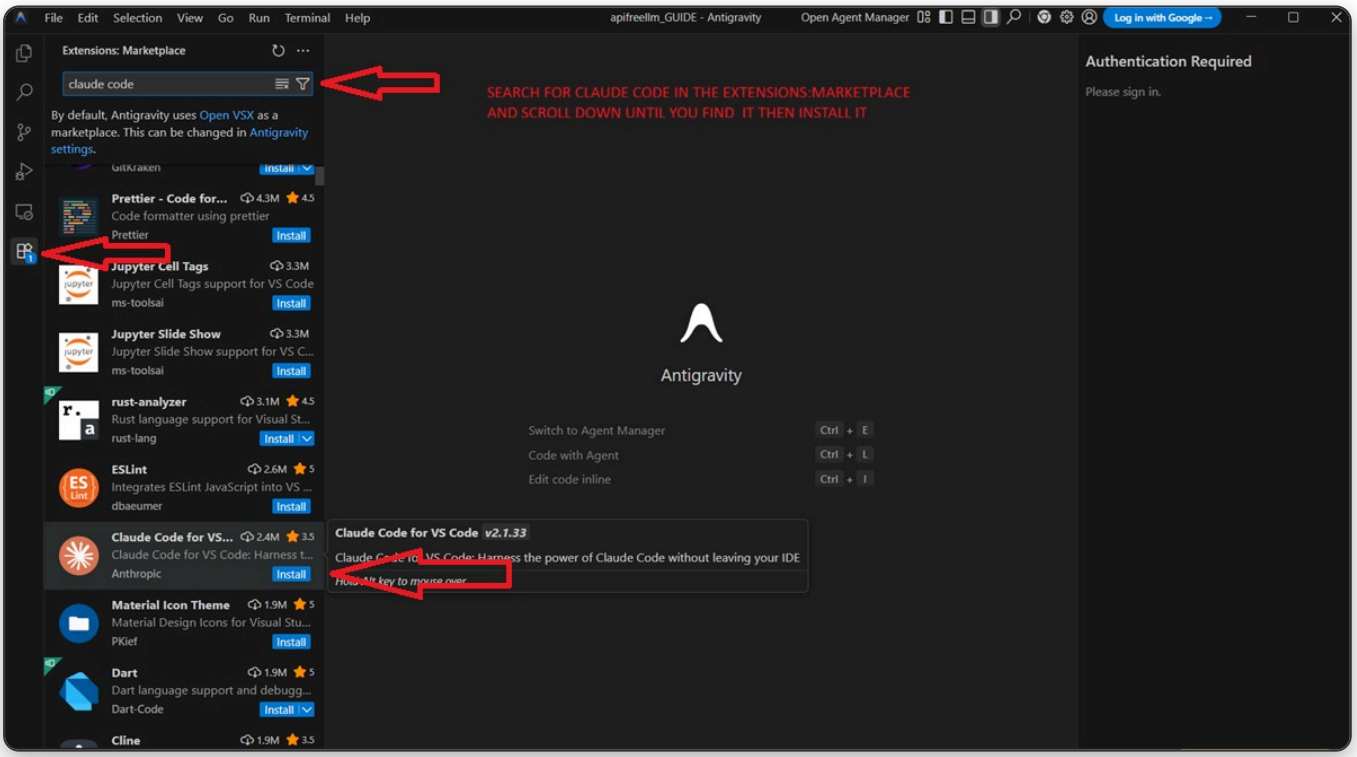
sélectionnez **« Focus Editor »**. Cela ouvre l'environnement VS Code complet pour cet espace de travail, où vous écrirez et modifierez votre code.



Cliquez sur les trois points à côté du nom de votre espace de travail et sélectionnez « Focus Editor ».

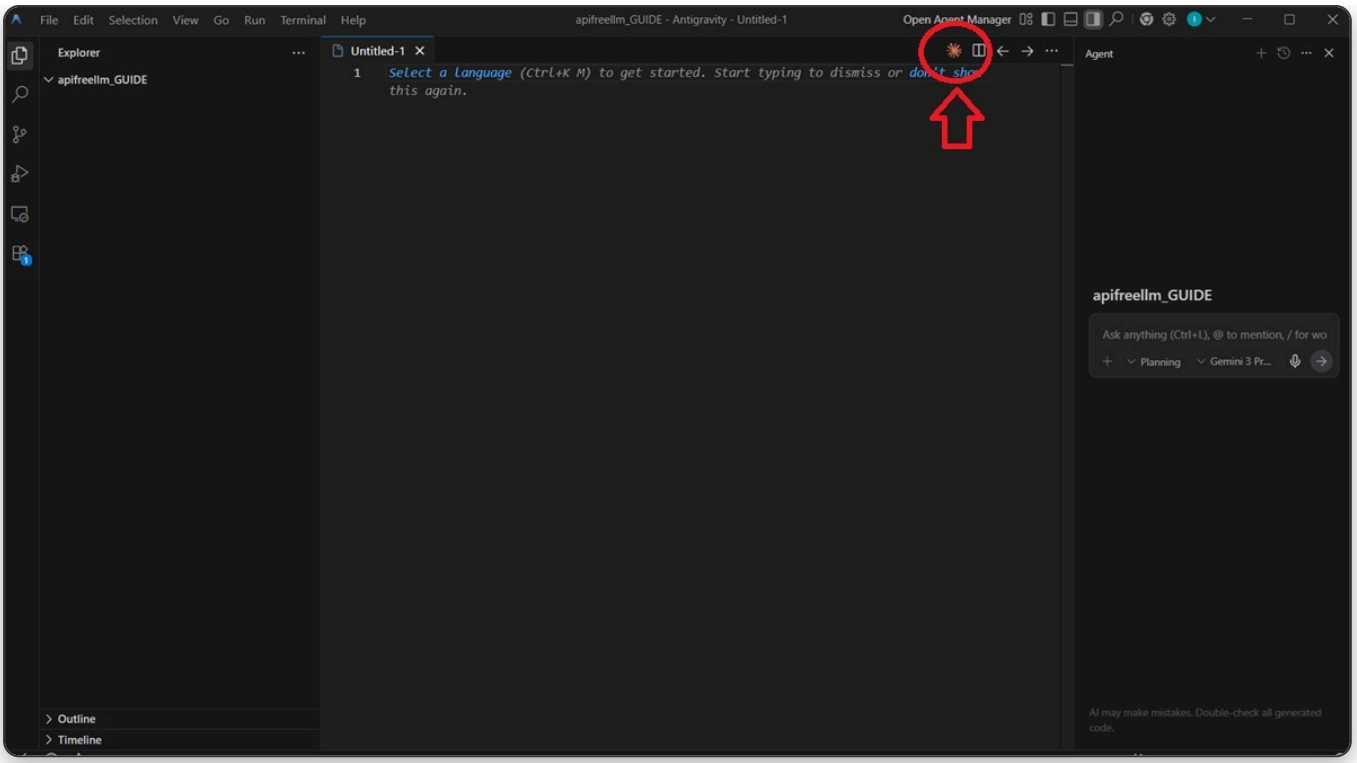
Installer Claude Code comme extension VS Code

Maintenant que l'éditeur est ouvert, il est temps d'installer Claude Code. Comme l'éditeur est basé sur VS Code, vous avez accès au marketplace d'extensions. Cliquez sur l'**icône Extensions** dans la barre latérale gauche (ou appuyez sur `Ctrl+Shift+X`). Dans la barre de recherche, tapez **« claude code »**. Vous devrez peut-être faire défiler les résultats vers le bas — cherchez **« Claude Code for VS Code »** par Anthropic. Une fois trouvé, cliquez sur le bouton **Installer**.



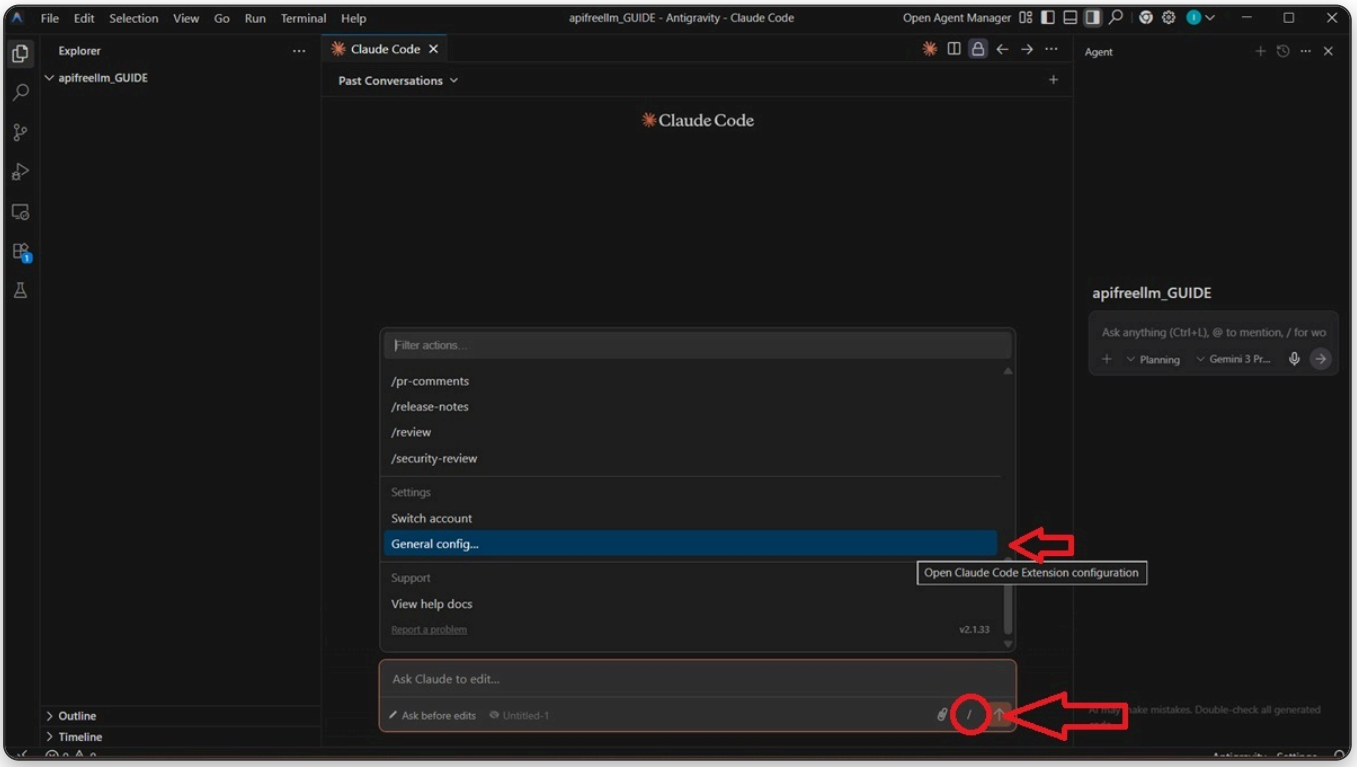
Recherchez « claude code » dans le marketplace d'extensions, faites défiler pour le trouver et cliquez sur Installer.

Une fois installé, fermez le panneau Extensions et ouvrez un nouveau fichier (ou n'importe quel fichier de votre projet). Vous remarquerez une petite **icône Claude Code** apparaissant dans la zone en haut à droite de l'éditeur — elle ressemble à un petit symbole orange. Cliquez dessus pour ouvrir le panneau de chat Claude Code.



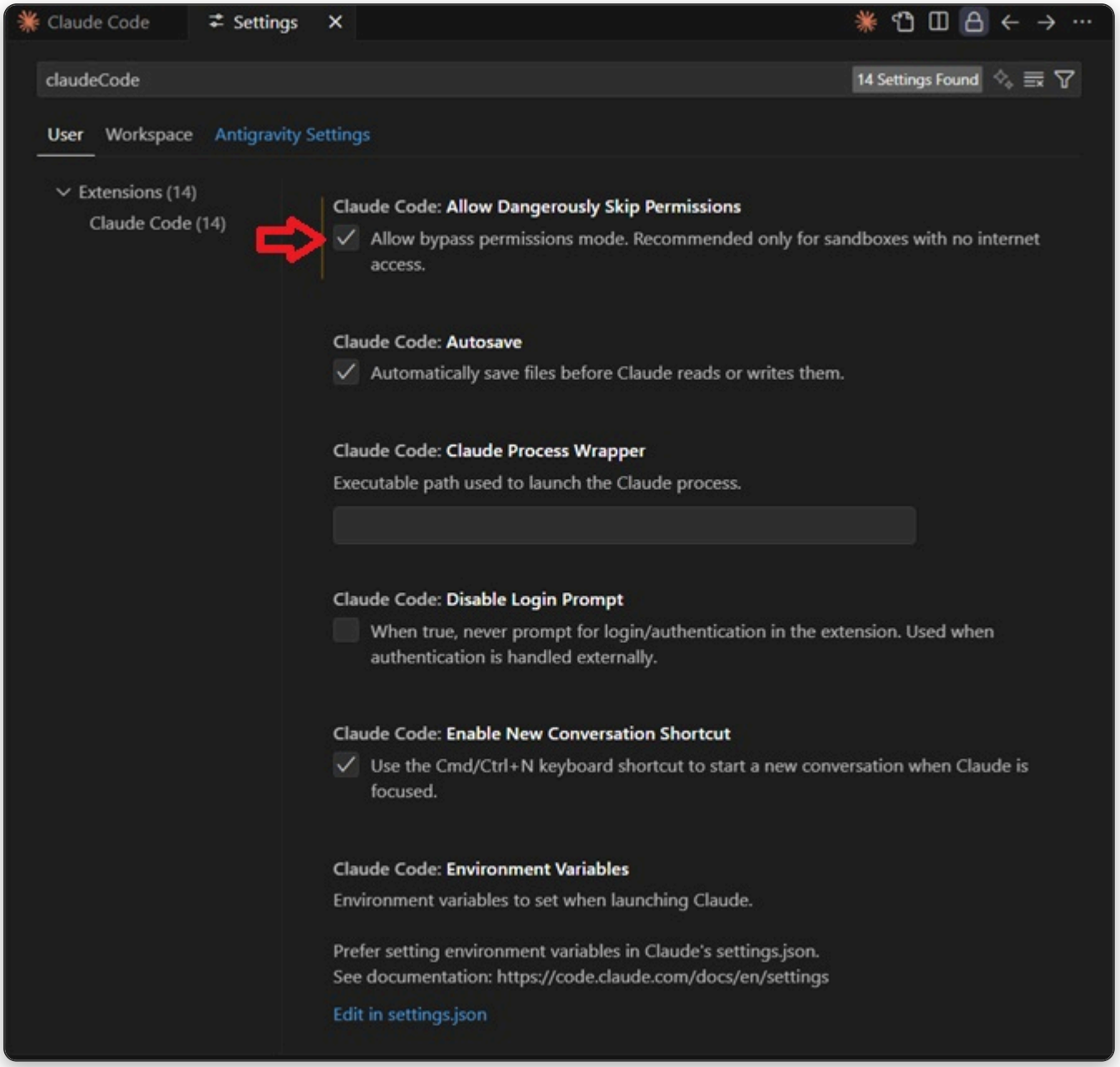
Le bouton Claude Code (entouré) apparaît en haut à droite de l'éditeur. Cliquez dessus pour ouvrir le chat Claude Code.

Claude Code vous demandera de vous connecter avec votre compte Anthropic (celui avec votre abonnement Pro). Après la connexion, vous devez le configurer. Dans le panneau de chat Claude Code, cliquez sur le bouton **/** en bas du panneau. Un menu apparaîtra avec différentes commandes. Faites défiler jusqu'à la section **Settings** et cliquez sur **« General config... »** pour ouvrir la configuration de l'extension Claude Code.



Cliquez sur le bouton « / » en bas, puis faites défiler jusqu'à Settings et sélectionnez « General config... » pour ouvrir la configuration.

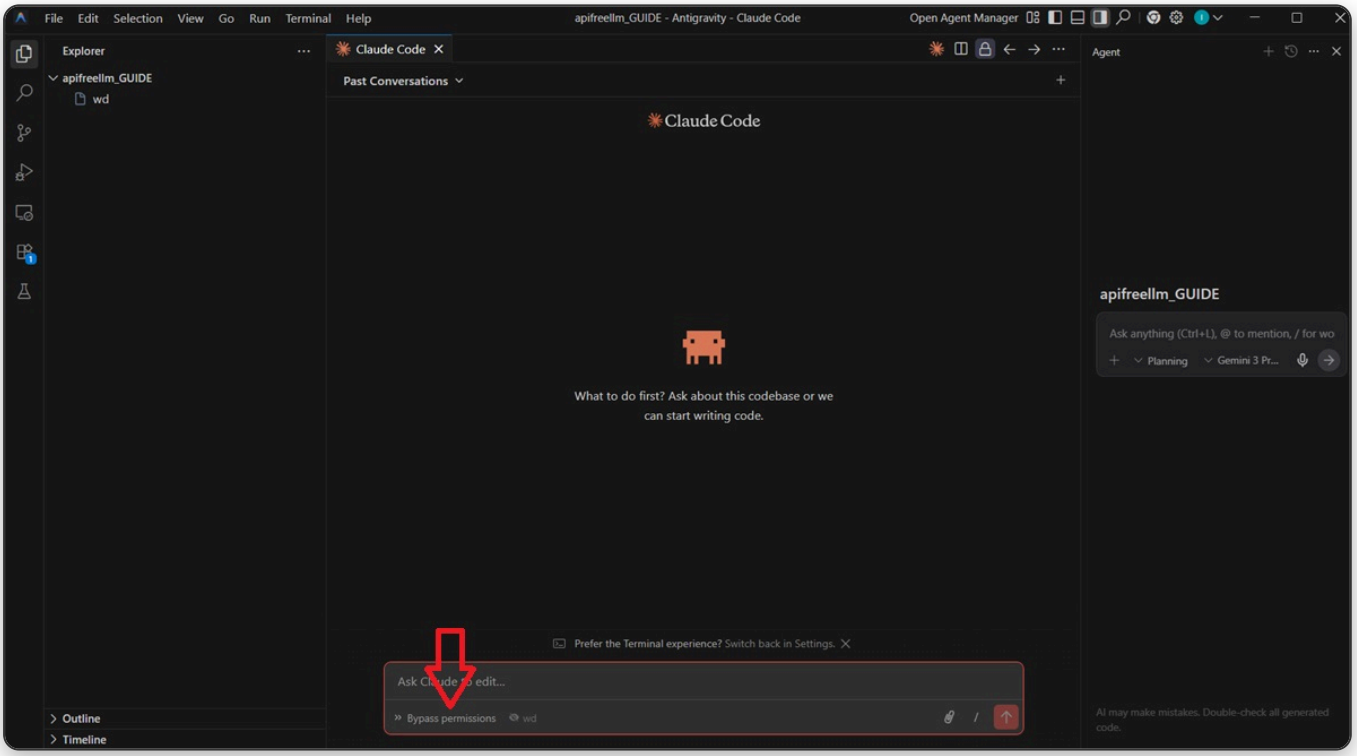
Dans le panneau de configuration qui s'ouvre, cherchez l'option appelée **« Allow Dangerously Skip Permissions »**. Activez cette option. Une fois activée, vous pourrez sélectionner **« Bypass permissions »** comme mode de permissions, ce qui permet à Claude Code de fonctionner de manière totalement autonome — lecture de fichiers, écriture de code, exécution de commandes terminal et modifications sans demander de confirmation à chaque étape.



Activez l'option « Allow Dangerously Skip Permissions » dans les paramètres de Claude Code, puis sélectionnez « Bypass permissions ».

> **Important :** Avec le contournement des permissions activé, Claude Code exécutera des actions sur votre système sans approbation manuelle. C'est essentiel pour un flux de développement fluide, mais cela implique une responsabilité. Soyez prudent avec le code que vous lui demandez d'exécuter, et ne le pointez jamais vers des dépôts non fiables ou des URLs suspectes. Les agents IA peuvent être exploités par injection de prompts — du contenu malveillant caché dans des fichiers ou sites web qui trompe l'agent pour qu'il exécute des commandes nuisibles. Examinez toujours ce que fait Claude Code, surtout lorsque vous travaillez avec des ressources externes.

Nous recommandons également d'activer le paramètre qui fait de **« Bypass permissions » la sélection par défaut** pour les nouveaux chats, afin de ne pas avoir à le sélectionner manuellement à chaque nouvelle conversation. Maintenant fermez Claude Code et rouvrez-le (en cliquant à nouveau sur le bouton de l'icône Claude Code). À partir de maintenant, vous verrez l'option **« Bypass permissions »** disponible dans le bouton de sélection des permissions en bas du panneau de chat Claude Code. Cliquez dessus pour l'activer.



Sélectionnez « Bypass permissions » depuis le bouton indiqué dans la capture d'écran.

Avec Bypass permissions actif, Claude Code exécutera des commandes, créera des fichiers, éditera du code et lancera des opérations terminal de manière complètement autonome — sans s'arrêter pour demander votre approbation à chaque étape. C'est ce qui vous permet d'**automatiser 100 % du flux de développement** : vous décrivez ce que vous voulez construire, et Claude Code le construit de manière autonome du début à la fin. Plus besoin de cliquer sur « Accepter » pour chaque modification de fichier ou exécution de commande. Vous donnez les instructions, et l'agent s'occupe du reste.

Gérer votre utilisation intelligemment

Une chose que vous devez savoir dès le départ : chaque interaction avec Claude Code consomme des **tokens**, et votre abonnement a un plafond d'utilisation. La bonne nouvelle, c'est que vous pouvez surveiller exactement combien vous avez utilisé. Dans le panneau de chat Claude Code, cliquez sur le bouton **/** — parmi les commandes disponibles vous trouverez votre **utilisation actuelle**, montrant combien de tokens vous avez consommés et quelle capacité il vous reste.

Tous les modèles ne consomment pas les tokens au même rythme. **Claude Opus 4.6** est le modèle le plus intelligent et le plus capable, mais il consomme aussi le plus de tokens par interaction. Les modèles plus petits comme **Sonnet** ou **Haiku** sont moins puissants mais ont des plafonds d'utilisation plus élevés et consomment nettement moins de tokens. Notre recommandation : utilisez **Opus pour les tâches complexes** qui nécessitent un raisonnement approfondi, des modifications multi-fichiers ou des décisions architecturales — c'est là que son intelligence fait une vraie différence. Pour les tâches plus simples comme les corrections rapides, les petites modifications ou les questions directes, passez à un modèle plus léger pour préserver vos tokens Opus pour quand ils comptent vraiment.

Il existe une autre stratégie pour économiser vos tokens Claude : **utilisez l'agent intégré d'Antigravity** pour les questions qui ne nécessitent pas le niveau d'intelligence de Claude. Antigravity supporte plusieurs modèles, mais nous recommandons d'utiliser **Gemini** pour ces questions rapides — puisqu'Antigravity est un produit Google, Gemini a le plafond d'utilisation le plus élevé et est aussi le modèle le plus rapide disponible sur la plateforme. Besoin d'un rappel CSS rapide ? Envie de connaître la syntaxe d'une commande Git ? Curieux du fonctionnement d'une bibliothèque ? Demandez à Antigravity au lieu de Claude Code. Ainsi, vous gardez vos tokens Claude pour le gros du travail de développement où ils ont le plus d'impact.

Comme vous pouvez le voir dans la capture d'écran précédente, nous recommandons de garder le **chat Claude Code à gauche** et le **chat de l'agent Antigravity à droite**. Cette disposition côte à côte vous donne un accès instantané aux deux agents en permanence.

> Le flux de travail intelligent : **Opus pour construire, les modèles légers pour les tâches rapides, Antigravity pour les questions générales.** Ainsi vous maximisez la valeur de votre abonnement Claude. Si vous êtes à court sur votre plafond d'utilisation Claude, rappelez-vous que l'agent Gemini d'Antigravity est juste à côté de vous pour les questions plus simples — utilisez-le pour économiser vos tokens Claude pour les tâches de développement qui en ont vraiment besoin.

Configuration essentielle

Quelle que soit la manière dont vous installez Claude Code, ces étapes de configuration amélioreront considérablement vos résultats :

- **Utilisez un fichier CLAUDE.md** — Placez un fichier `CLAUDE.md` à la racine de votre projet. Ce fichier est automatiquement lu par Claude Code au début de chaque session. Utilisez-le pour décrire la structure de votre projet, les conventions de code, la stack technique et toutes les

règles que l'agent doit suivre. Considérez-le comme une documentation d'intégration pour votre développeur IA.

- **Gardez votre projet organisé** — Les agents IA fonctionnent considérablement mieux avec des bases de code propres et bien structurées. Si votre code est un fouillis, l'agent produira un résultat brouillon. Une bonne structure de dossiers, des conventions de nommage claires et des patterns cohérents font une différence massive.
- **Utilisez le contrôle de version** — Travaillez toujours avec Git initialisé. Cela vous donne un filet de sécurité. Si l'agent fait une erreur, vous pouvez revenir en arrière instantanément. Cela permet aussi à l'agent de créer des commits pour vous, ce qui est étonnamment utile pour suivre ce qui a changé et pourquoi.

Outils essentiels : Git & GitHub CLI

Avant de commencer à construire quoi que ce soit, il y a deux outils qui doivent être installés sur votre système : **Git** et le **GitHub CLI (gh)**. Ils sont fondamentaux pour tout flux de développement moderne, et ce sont eux qui permettent à vos agents IA de gérer vos dépôts de code de manière autonome.

Pourquoi Git et GitHub CLI sont importants

Git est le système de contrôle de version qui suit chaque modification dans votre projet. C'est votre filet de sécurité : si Claude Code fait une erreur, vous pouvez revenir en arrière instantanément. Il permet aussi à l'agent de créer des commits, gérer des branches et maintenir un historique propre de l'évolution de votre projet — le tout automatiquement.

GitHub CLI (gh) est un outil en ligne de commande qui donne un accès direct à GitHub depuis le terminal. C'est la clé de l'automatisation complète : une fois `gh` installé et authentifié, Claude Code peut créer des dépôts, pousser du code, gérer des pull requests, configurer les paramètres du dépôt, mettre en place GitHub Actions pour le déploiement, ajouter des secrets, et bien plus — le tout depuis son terminal, sans que vous ayez à ouvrir GitHub dans un navigateur.

Installer Git et GitHub CLI

La façon la plus simple d'installer ces outils est de **demander à Claude Code ou Antigravity de le faire pour vous**. Dites simplement à votre agent : *« Installe Git et le GitHub CLI sur mon système. »* L'agent détectera votre système d'exploitation et exécutera les commandes d'installation appropriées. Sous Windows, il utilisera typiquement `winget` ou téléchargera les installateurs ; sur macOS, il utilisera `brew` ; sur Linux, `apt` ou le gestionnaire de paquets de votre système.

Si vous préférez les installer manuellement, vous pouvez télécharger Git depuis le site officiel et le GitHub CLI depuis sa page de releases sur GitHub. Mais laisser l'IA s'en charger est plus rapide et évite les erreurs d'installation courantes.

Authentifier GitHub CLI

Après l'installation, vous devez vous connecter pour que `gh` puisse accéder à votre compte GitHub. Encore une fois, vous pouvez simplement demander à votre agent : *« Connecte-moi au*

GitHub CLI. » L'agent exécutera `gh auth login` et vous guidera à travers le processus d'authentification, qui implique généralement l'ouverture d'un lien dans le navigateur et la saisie d'un code. Une fois authentifié, l'agent a un accès complet à vos dépôts GitHub.

> **C'est un vrai game-changer.** Avec `gh` authentifié, vous pouvez dire à Claude Code des choses comme : *« Crée un nouveau dépôt privé appelé my-app, initialise le projet et pousse le code. »* Ou plus tard : *« Mets en place une GitHub Action qui déploie sur mon serveur à chaque push sur main. »* L'agent gère tout — création de fichiers, configuration des secrets, mise en place des workflows — sans que vous quittiez jamais l'éditeur. C'est ça, la vraie automatisation du développement.

3. Votre Premier Projet : De Zéro au Déploiement

> *Votre environnement est prêt. Claude Code est ouvert, Antigravity tourne, Git et GitHub CLI sont configurés. Il est temps de construire quelque chose de concret. À partir de maintenant, ce cours passe de l'installation à la stratégie — des techniques pratiques et des perspectives qui vous donneront un vrai avantage quand vous construirez avec des agents IA.*

Ne partez jamais de zéro

C'est le conseil le plus important de tout ce cours : **ne construisez jamais à partir de zéro.**

Même si Claude Opus est un modèle incroyablement avancé et intelligent, il fera des erreurs. Tous les modèles d'IA en font. Il peut mal comprendre la structure de votre projet, utiliser une bibliothèque obsolète, créer une arborescence de fichiers incohérente, ou générer du code qui ne s'assemble pas bien quand le projet grandit. Partir d'un dossier vide signifie que l'IA doit prendre des centaines de décisions sans point de référence — et certaines de ces décisions seront inévitablement mauvaises.

Voici la réalité : **99,9 % de ce que vous voulez construire existe déjà** sous une forme ou une autre. Que ce soit une boutique e-commerce, un tableau de bord SaaS, un site portfolio, une application de réseau social ou une plateforme de réservation — quelqu'un a déjà construit quelque chose de similaire. Et beaucoup de ces projets sont open-source, disponibles comme templates sur GitHub, prêts à être clonés et personnalisés.

Votre flux de travail devrait toujours commencer de la même manière : **cherchez d'abord un template.** Allez sur GitHub et cherchez des projets open-source qui correspondent à ce que vous voulez construire. Trouvez-en un qui soit proche de votre vision, clonez-le dans votre dossier de projet, puis pointez Claude Code dessus. Maintenant, au lieu de construire à partir de zéro, l'agent

modifie, améliore et personnalise une base de code existante et fonctionnelle. La différence en qualité et en vitesse est énorme.

> **Les templates ne sont pas de la triche — c'est de la stratégie.** Les développeurs professionnels utilisent des boilerplates et des starter kits en permanence. Vous ne copiez pas le produit de quelqu'un d'autre. Vous utilisez une fondation structurelle et vous construisez votre propre produit unique par-dessus. L'IA fonctionne considérablement mieux quand elle a des patterns existants à suivre plutôt que de tout inventer à partir de rien.

Choisir la bonne stack

Si votre recherche de template ne donne rien et que vous devez vraiment démarrer un nouveau projet, la technologie que vous choisissez peut faire une différence — surtout quand vous travaillez avec des agents IA. Cela dit, il n'y a pas de choix unique obligatoire. La meilleure stack est celle qui vous amène le plus vite à un produit fonctionnel.

Claude Code, comme tous les LLM, est fondamentalement meilleur pour écrire du **code web** : HTML, CSS, JavaScript et TypeScript. C'est le langage d'internet, et c'est ce sur quoi ces modèles ont été le plus entraînés. Plus votre projet reste proche des technologies web, mieux l'IA performera.

Pour les **applications web**, **Next.js** est un excellent choix si vous en trouvez un. C'est un framework React moderne avec du rendu côté serveur intégré, ce qui est crucial pour le SEO. Claude Code fonctionne exceptionnellement bien avec les projets Next.js : il comprend le routage basé sur les fichiers, les routes API, les composants serveur et tout l'écosystème. Vous trouverez un nombre énorme de templates Next.js sur GitHub pour pratiquement tout type d'application.

Pour les **applications de bureau** (exécutables pour Windows, macOS ou Linux), **Electron** est une excellente option. Electron vous permet de construire des applications de bureau en utilisant HTML, CSS et JavaScript — les mêmes technologies web dans lesquelles Claude excelle. Comme l'interface est essentiellement une page web rendue dans une fenêtre native, l'IA peut construire de belles applications de bureau fonctionnelles avec la même facilité que pour un site web.

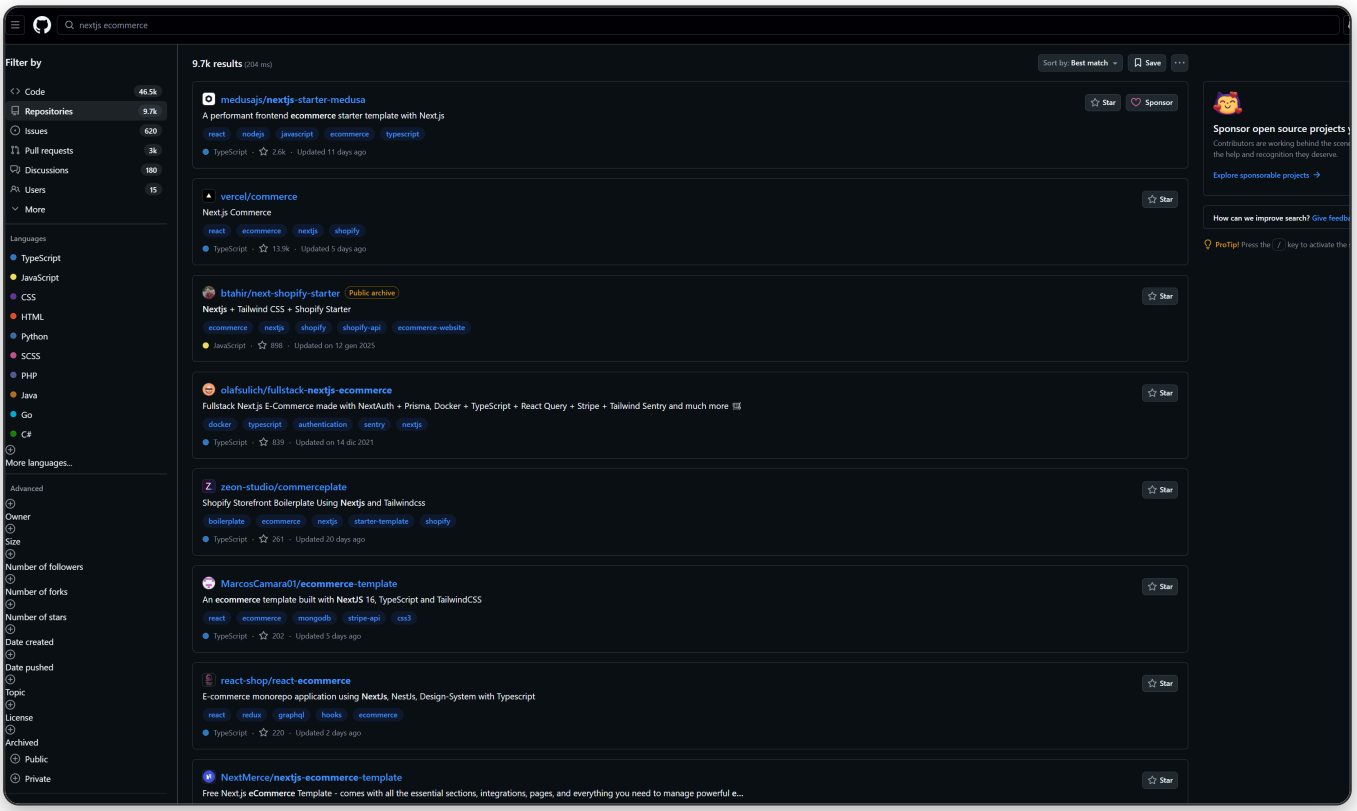
Mais voici la nuance importante : **un template bien construit dans une stack plus ancienne est presque toujours meilleur que de partir de zéro avec une stack moderne.** Vous pourriez trouver un projet PHP, jQuery ou Laravel qui correspond exactement à vos besoins — complet, bien structuré, éprouvé, avec toutes les fonctionnalités déjà implémentées. Dans ce cas, utilisez-le. Les agents IA peuvent travailler avec n'importe quelle technologie, et le temps que vous économisez en ayant une fondation solide et prête à l'emploi dépasse largement les avantages théoriques d'un framework plus récent. Claude Code peut comprendre et modifier du PHP, Python, Ruby ou toute autre base de code sans problème.

La priorité est simple : **trouvez le meilleur template disponible.** Si vous trouvez deux templates de qualité similaire et que l'un utilise Next.js tandis que l'autre utilise PHP, optez pour Next.js. Mais si le template PHP est plus complet, plus riche en fonctionnalités et mieux maintenu — choisissez-le sans hésiter. La qualité et la complétude du point de départ comptent plus que la modernité de la stack.

> **La règle d'or :** utilisez ce que vous trouvez qui vous rapproche le plus de votre objectif. Visez les technologies web modernes (Next.js, Electron, React Native) quand elles sont disponibles, mais ne rejetez jamais un excellent template simplement parce qu'il utilise une stack plus ancienne. Le temps économisé grâce à une fondation solide a toujours plus de valeur que la pureté de la stack.

Exemple pratique : trouver un template

Disons que vous voulez construire une boutique e-commerce. Au lieu de dire à Claude Code « *Construis-moi un site e-commerce à partir de zéro* », ouvrez GitHub et cherchez quelque chose comme **« nextjs ecommerce »**. Vous trouverez des dizaines de projets prêts à l'emploi avec des listes de produits, des paniers, des flux de paiement et l'intégration des paiements déjà construits.



Une recherche rapide sur GitHub pour « nextjs ecommerce » montre déjà plusieurs templates prometteurs.

Un point important à garder à l'esprit : de nombreux templates sur GitHub sont des **projets freemium** — le noyau est open-source et gratuit, mais certaines fonctionnalités ou versions premium nécessitent un paiement. Vérifiez toujours le README et la licence du dépôt avant de vous engager avec un template. Évitez tout ce qui nécessite des abonnements payants ou des coûts cachés dont vous n'avez pas besoin.

En regardant les résultats de recherche, on peut repérer **fullstack-nextjs-ecommerce** — et c'est un excellent point de départ. Analysons pourquoi. Le projet utilise Next.js avec TypeScript, ce qui est exactement ce que nous voulons pour le développement assisté par IA. Mais ce qui se distingue vraiment, c'est ce qui est déjà intégré : **Stripe pour les paiements**, **PostgreSQL avec**

Prisma comme base de données, et **NextAuth pour l'authentification**. Ce sont trois des parties les plus critiques — et les plus sujettes aux erreurs — de toute application web.

Avoir **Stripe déjà intégré** est particulièrement précieux. Le traitement des paiements implique des webhooks, la gestion des sessions, la gestion des erreurs et des cas limites qui peuvent être étonnamment difficiles à bien gérer. Quand quelque chose ne va pas avec les paiements, ce sont vos clients qui en souffrent — des prélèvements échoués, des paiements en double ou des flux de paiement cassés sont le genre de bugs qui détruisent la confiance et vous coûtent de l'argent réel. Partir d'un template qui a déjà une intégration Stripe fonctionnelle vous épargne ces maux de tête.

Le projet utilise aussi **PostgreSQL** comme base de données, ce qui est un choix solide. Postgres est fiable, bien documenté, offre des paramètres de sécurité par défaut légèrement meilleurs que les alternatives comme MySQL, et peut être facilement hébergé sur Amazon AWS, Hetzner ou des fournisseurs similaires — nous couvrirons la configuration du serveur et le déploiement dans le prochain chapitre. Le fait que **NextAuth** soit déjà câblé signifie que l'authentification des utilisateurs est gérée nativement, une autre pièce complexe que vous n'avez pas à construire à partir de zéro.

C'est la puissance de partir avec le bon template : au lieu de passer des jours (ou des semaines) à configurer les paiements, la base de données et l'authentification — des choses faciles à mal faire — vous partez d'un projet où ces systèmes critiques fonctionnent déjà. Vous pouvez alors vous concentrer entièrement sur la personnalisation du produit selon votre vision : changer le design, ajouter vos produits, modifier la logique métier et construire les fonctionnalités qui rendent votre boutique unique.

Une fois que vous avez trouvé votre template, l'étape suivante est simple : **téléchargez-le et placez-le dans votre dossier d'espace de travail.** Ouvrez ce dossier avec Claude Code (ou Antigravity) et commencez à travailler. Vous pouvez dire à Claude Code de modifier directement le template — changer le branding, ajouter de nouvelles pages, retravailler la mise en page, intégrer de nouvelles fonctionnalités — ou vous pouvez utiliser le template comme **référence** pour votre propre projet. Même si vous construisez quelque chose de légèrement différent, avoir le template dans le même espace de travail signifie que Claude Code peut le consulter, étudier les patterns du code et les utiliser comme fondation pour ce qu'il écrit.

C'est un point crucial : **donnez toujours à Claude Code quelque chose à référencer.** Quand l'agent a une base de code solide et fonctionnelle à consulter, il écrit du code nettement meilleur. Il suit les mêmes patterns, utilise les mêmes conventions et produit un résultat cohérent et fiable. Quand il n'a rien à référencer et doit tout générer à partir de zéro, c'est là que les erreurs surviennent — des structures de fichiers incohérentes, des mauvaises versions de bibliothèques, du code qui ne s'assemble pas. Un template agit comme un ancrage qui maintient l'IA en phase et lui fait produire des résultats de haute qualité et cohérents.

Et si le template n'a pas de paiements ni de base de données ? Ce n'est pas un problème. Ce cours inclut des fichiers d'intégration Stripe prêts à l'emploi que vous pouvez donner directement à Claude Code pour intégrer les paiements dans n'importe quel projet. Pour la base de données, nous recommandons PostgreSQL ou la base de données déjà utilisée par le template — ne combattez pas les choix du template sauf s'il y a une bonne raison. Idem pour l'authentification : si le template utilise NextAuth, Clerk ou tout autre système d'auth, travaillez avec. L'objectif est de tirer parti de ce qui est déjà construit, pas de tout remplacer.

4. Du Code à la Production

> *Votre produit est construit. Maintenant il doit accepter des paiements, tourner sur un serveur, et être rapide et sécurisé pour les utilisateurs du monde entier. Ce chapitre couvre les services essentiels qui transforment votre projet de code local en une entreprise en ligne prête pour la production.*

Une note sur ce chapitre. Nous restons concis et allons droit au but ici. Notre objectif est de vous dire **ce que** vous devez faire et **pourquoi** — pas d'écrire de longs tutoriels qui vous font perdre du temps. N'importe quel LLM (Claude, Gemini, ChatGPT) peut expliquer les détails, vous guider à chaque étape et répondre à vos questions bien mieux qu'une page statique ne le pourra jamais. Dès que quelque chose n'est pas clair, demandez simplement à votre agent : *« Je suis un cours qui dit que je dois [faire X]. Peux-tu m'expliquer ce que ça signifie et me guider ? »* C'est plus rapide, plus personnalisé et toujours à jour.

Paiements avec Stripe

Si votre produit vend quoi que ce soit, vous avez besoin d'un processeur de paiement. **Stripe** est le standard de l'industrie : fiable, bien documenté, et supporté par pratiquement tous les agents IA en raison de son utilisation répandue.

Voici ce que vous devez faire :

- **Créez un compte Stripe** sur stripe.com. Vous obtiendrez des **clés API** (une clé publiable pour le frontend, une clé secrète pour le backend) et accès au **mode test** (faux paiements pour le développement) et au **mode live** (argent réel).
- **Configurez les webhooks** dans le tableau de bord Stripe. Les webhooks sont des URLs sur votre serveur que Stripe appelle quand quelque chose se passe — un paiement réussit, un abonnement se renouvelle, un prélèvement échoue. C'est comme ça que votre application sait quand activer un abonnement, confirmer une commande ou gérer un échec. Vous avez

besoin de webhooks pour les **tests locaux** (Stripe a un outil CLI qui redirige les événements vers votre localhost) et la **production** (pointant vers votre serveur live). Faites toujours fonctionner tout en local avant de passer en production.

- **Intégrez Stripe dans votre projet.** Si votre template a déjà Stripe, branchez simplement vos clés API. Sinon, ce cours inclut des fichiers d'intégration Stripe prêts à l'emploi — déposez-les dans votre espace de travail et dites à Claude Code de les intégrer. Stripe supporte les paiements uniques et les abonnements récurrents, les deux utilisant des pages de paiement hébergées qui gèrent la validation de carte, le 3D Secure et la conformité PCI pour vous.

Une règle cruciale : **vérifiez toujours les paiements côté serveur via les webhooks**, ne faites jamais confiance au frontend. Le webhook est Stripe qui dit directement à votre serveur que l'argent a réellement changé de mains — c'est la seule source de vérité fiable.

Hébergement : AWS, Hetzner & au-delà

Votre application doit vivre quelque part. La bonne nouvelle : **AWS vous offre un Free Tier** quand vous vous inscrivez — pendant **un an complet**, vous obtenez un **serveur t2.micro** et une **base de données micro** entièrement gratuitement. C'est suffisant pour héberger votre premier projet pendant que vous validez l'idée et commencez à avoir des utilisateurs.

Vous ne savez pas comment créer un compte AWS ou utiliser le Free Tier ? Demandez simplement à Claude ou Antigravity — ils vous guideront à chaque étape.

Quand vous dépassez le free tier, voici ce que vous devez savoir sur le dimensionnement des serveurs :

- **t3.small** est le meilleur compromis pour la plupart des applications — bon CPU, assez de RAM, et un prix raisonnable (~16 $/mois sur AWS). Le « t » fait référence à la génération d'instance ; les générations plus anciennes (t2, etc.) peuvent parfois coûter plus pour moins de performance, donc restez sur la plus récente disponible.
- **Hetzner** est une alternative européenne *nettement* moins chère — vous pouvez obtenir des performances comparables à un t3.small pour environ **4 $/mois**. C'est environ 4 fois moins cher qu'AWS pour des specs similaires.
- **Toutes les applications n'ont pas besoin d'un serveur.** Si votre projet est un site statique ou une application JAMstack, vous n'avez peut-être pas besoin d'un serveur dédié du tout. Des plateformes comme Vercel, Netlify ou même Cloudflare Pages peuvent l'héberger gratuitement ou presque. Demandez toujours à votre LLM : *« Quel est le meilleur hébergement pour mon application spécifique ? »*

Notre recommandation : **commencez avec le AWS Free Tier** pour votre première année, puis évaluez si Hetzner ou un autre fournisseur a plus de sens pour votre budget et la localisation de votre audience. Si la plupart de vos utilisateurs sont en Europe, les serveurs allemands de Hetzner vous donneront une latence plus faible. Si votre audience est mondiale ou basée aux États-Unis, les régions AWS pourraient être plus adaptées.

Clés SSH & gestion de serveur par IA. Pour permettre à Claude Code de se connecter et de gérer votre serveur à distance, vous devrez configurer une **clé SSH**. Demandez à Claude d'en générer une et de la configurer sur votre serveur. Une fois connecté, Claude peut déployer du code, gérer des services, résoudre des problèmes — le tout depuis votre terminal. Pour les tâches de gestion de serveur, nous recommandons d'utiliser **Opus** pour les meilleurs résultats, bien que Sonnet fonctionne aussi si vous voulez économiser des tokens (avec un risque d'erreur légèrement plus élevé).

Cloudflare : Performance & Protection

Une fois votre application sur un serveur, vous avez besoin de quelque chose en amont pour la protéger et l'accélérer. C'est **Cloudflare**. La bonne nouvelle : **le plan gratuit est largement suffisant** pour la grande majorité des sites web. Vous n'avez pas besoin d'un plan payant.

Mais d'abord, vous aurez besoin d'un **nom de domaine**. Nous recommandons d'en acheter un directement chez **Cloudflare** ou **Namecheap** — les deux sont fiables et à prix correct. Une fois votre domaine obtenu, vous devrez configurer le **DNS** pour le faire pointer vers l'adresse IP de votre serveur AWS ou Hetzner. Demandez simplement à votre LLM : *« J'ai acheté un domaine sur [Cloudflare/Namecheap]. Comment configurer le DNS pour le faire pointer vers mon serveur à [votre IP] ? »* Il vous guidera pas à pas.

Pourquoi Cloudflare est-il si important ? Il protège votre site des **attaques DDoS**, de diverses **failles de sécurité**, et des vulnérabilités web courantes. Il fournit aussi un **cache global**, ce qui signifie que votre contenu est servi depuis des serveurs proches de vos utilisateurs, rendant votre site plus rapide dans le monde entier. Sans un service comme Cloudflare, vous exposez votre site à des risques sérieux — surtout maintenant, à l'ère de l'IA, où n'importe qui peut utiliser des outils IA pour trouver des vulnérabilités, exploiter des mauvaises configurations, ou même voler des données sur des sites web mal protégés. Ne sautez pas cette étape.

Astuce rapide : mode SSL Flexible. Lors de la configuration de Cloudflare, vous pouvez choisir le mode SSL **Flexible**. Cela signifie que la connexion entre Cloudflare et votre serveur utilise du HTTP simple, mais la connexion entre Cloudflare et vos utilisateurs finaux est en HTTPS — donc les visiteurs voient un cadenas sécurisé. Cela vous évite de configurer des certificats SSL sur votre serveur, ce qui est idéal pour démarrer rapidement. Pour les applications en production qui gèrent des données sensibles, vous devriez éventuellement passer au mode **Full** (chiffré de bout en bout). Mais pour vos premiers tests et lancements, Flexible convient parfaitement et fait gagner beaucoup de temps de configuration. Demandez à votre LLM d'expliquer les différences si vous n'êtes pas sûr de quel mode convient à votre projet.

Cloudflare offre bien plus que de la simple protection. Le plan gratuit inclut des fonctionnalités puissantes que beaucoup de développeurs ne connaissent même pas :

- **Cloudflare Pages** — hébergement statique gratuit pour les apps frontend (React, Next.js static export, Vue, etc.). Vous poussez sur GitHub, Cloudflare compile et déploie

automatiquement. Zéro configuration, zéro coût. Parfait pour les landing pages, portfolios et apps JAMstack.

- **Edge Functions (Workers)** — exécutez du code serverless à la périphérie, proche de vos utilisateurs, avec le plan gratuit. Idéal pour les routes API, redirections, A/B testing et la logique backend légère sans serveur dédié.
- **CDN & Caching** — vos ressources statiques (images, CSS, JS) sont mises en cache globalement, rendant votre site ultra rapide depuis n'importe où dans le monde.

La question clé est : **votre app a-t-elle vraiment besoin d'un serveur dédié, ou Cloudflare peut-il la gérer gratuitement ?** Demandez à Claude : *"Je construis [décrivez votre app]. Ai-je besoin d'un serveur dédié sur AWS/Hetzner, ou puis-je utiliser Cloudflare Pages et Workers gratuitement ?"* Claude analysera votre cas spécifique et vous indiquera la meilleure option. Vous pourriez être surpris de voir combien de projets peuvent fonctionner entièrement sur le plan gratuit de Cloudflare.

Clés API : Automatisez Tout

Voici un conseil qui change la donne et que la plupart des tutoriels ne mentionnent pas : **créez des clés API pour vos fournisseurs d'hébergement** et donnez-les à Claude. Cela permet à Claude de gérer votre infrastructure directement depuis le terminal — sans cliquer sur des dashboards, sans copier-coller, sans travail manuel.

- **Clé API Cloudflare** — allez dans votre dashboard Cloudflare → My Profile → API Tokens → créez un token. Avec cela, Claude peut configurer automatiquement les enregistrements DNS, créer des projets Pages, gérer les Workers, mettre à jour les paramètres SSL et bien plus. Dites à Claude : *"Voici mon token API Cloudflare. Configure le DNS pour mon domaine pointant vers l'IP de mon serveur."* Fait en quelques secondes.
- **AWS Access Keys** — allez dans AWS IAM → créez une access key. Avec cela, Claude peut gérer les instances EC2, configurer les security groups, créer des bases de données RDS, gérer les buckets S3 et gérer toute votre infrastructure AWS de manière programmatique. Dites à Claude : *"Voici mes identifiants AWS. Lance une instance EC2 t3.small dans us-east-1 et configure les security groups pour une app web."*
- **Hetzner API Token** — allez dans votre Hetzner Cloud Console → projet → Security → API Tokens → générez-en un. Claude peut alors créer des serveurs, configurer des pare-feu, gérer des snapshots et gérer toute votre infrastructure Hetzner. Dites à Claude : *"Voici mon token API Hetzner. Crée un serveur CX22 à Nuremberg avec Ubuntu."*

> **Note de sécurité sur les clés API.** Ces clés API sont puissantes — traitez-les comme des mots de passe. **Ne les commitez jamais dans Git**, ne les partagez pas publiquement et ne les collez pas dans des interfaces de chat auxquelles vous ne faites pas confiance. Stockez-les dans un fichier `.env` ou passez-les directement à Claude dans votre session terminal. Si une clé est compromise, révoquez-la immédiatement depuis le dashboard du fournisseur et générez-en une nouvelle. De plus, lors de la création de tokens API, **utilisez toujours le principe du moindre privilège** : n'accordez que les permissions réellement nécessaires, pas un accès admin complet.

La combinaison de ces clés API avec Claude est incroyablement puissante. Vous pouvez littéralement dire : *"J'ai une app Next.js. Déploie-la sur Cloudflare Pages, configure le domaine personnalisé et configure le DNS — voici mes clés API."* Et Claude fera tout automatiquement. Ou : *"Crée une instance EC2 sur AWS, installe Node.js et PM2, configure nginx, configure le DNS Cloudflare et déploie mon app."* Configuration complète de l'infrastructure en minutes, pas en heures.

CI/CD avec GitHub Actions

Vous vous souvenez quand nous avons configuré Git et GitHub CLI dans le Chapitre 2 ? C'est ici que tout porte ses fruits. Avec `gh` authentifié, vous pouvez dire à Claude Code de **pousser votre projet vers un dépôt GitHub privé**, mettre en place **GitHub Actions**, configurer tous les **secrets** nécessaires (la clé SSH de votre serveur, les variables d'environnement, etc.), et créer un pipeline de déploiement automatique — le tout depuis son terminal, sans que vous touchiez à l'interface de GitHub.

L'idée est simple : une fois GitHub Actions configuré, **chaque fois que Claude Code pousse du code vers votre dépôt, il se déploie automatiquement sur votre serveur.** Pas de SSH manuel, pas de copie de fichiers, pas de commandes à exécuter sur le serveur vous-même. Claude pousse, GitHub Actions le récupère, se connecte à votre serveur AWS ou Hetzner via SSH et déploie tout. Entièrement automatisé.

Dites simplement à Claude : *« Pousse ce projet vers un nouveau dépôt privé sur GitHub, mets en place une GitHub Action qui déploie sur mon serveur à chaque push sur main. Voici l'IP de mon serveur et ma clé SSH. »* Claude sait déjà comment faire — il créera le fichier workflow, ajoutera la clé SSH et l'IP du serveur comme secrets GitHub, et configurera l'ensemble du pipeline. C'est exactement pour ça que nous avons installé le GitHub CLI plus tôt.

> **Le piège de l'IP dynamique.** Les IPs statiques coûtent généralement un supplément sur AWS comme sur Hetzner, donc vous utiliserez probablement une **IP dynamique** pour économiser. Cela signifie que chaque fois que vous arrêtez et redémarrez votre serveur, l'IP change. Quand cela arrive, vous devrez la mettre à jour à **deux endroits** : l'enregistrement DNS Cloudflare et le secret GitHub `SERVER_IP`. Dites à Claude de stocker l'IP du serveur comme secret `SERVER_IP` dans GitHub Actions, pour que quand elle change vous n'ayez qu'une seule variable à mettre à jour. Dites aussi à Claude de vous rappeler de vérifier et mettre à jour l'IP si un déploiement échoue — une IP changée est presque toujours la raison.

5. Tactiques Marketing & SEO

> *Votre produit est en ligne. Maintenant le monde doit savoir qu'il existe. Le marketing le plus efficace pour les produits indépendants est organique — gratuit, créatif et étonnamment puissant quand il est bien fait. Ce chapitre couvre les tactiques exactes que nous utilisons.*

Stratégies de croissance organique

Soyons honnêtes : **le meilleur marketing ne ressemble pas à du marketing.** Internet est saturé de publicités, et les gens ont développé un réflexe pour ignorer tout ce qui ressemble à une promotion. Ce qui fonctionne vraiment, c'est le **marketing furtif** — du contenu qui ressemble à une vraie information, une question ou une recommandation plutôt qu'à une publicité. Les gens sont naturellement curieux et adorent aider avec des suggestions. Utilisez cela.

Reddit est l'une des plateformes les plus puissantes pour cela. Il est massif, très bien indexé par Google, et rempli de communautés de niche où votre audience cible traîne déjà. Mais voici la clé : **ne publiez jamais de marketing agressif.** N'écrivez pas *« Découvrez mon nouveau site incroyable ! »* — ça sera downvoté, supprimé ou banni. Écrivez plutôt quelque chose comme :

- *« Quelqu'un peut recommander des sites similaires à [concurrent connu] ou [votre site] ? Je cherche des alternatives. »*
- *« Quelqu'un a essayé [votre catégorie de produit] ? J'ai trouvé [votre site] mais je suis curieux des autres options. »*
- Répondez aux questions dans les subreddits pertinents et mentionnez naturellement votre produit là où il aide véritablement.

C'est comme ça que fonctionne la majorité du marketing indépendant réussi sur Reddit. Vous ne mentez pas — vous présentez votre produit comme une découverte au sein d'une conversation authentique. Les gens cliquent parce qu'ils sont curieux, pas parce qu'ils se sentent démarchés. Et voici un bonus : **les publications Reddit sont indexées par Google**, donc un fil bien placé peut vous apporter du trafic de recherche organique pendant des mois voire des années.

Vous pouvez aussi **sponsoriser une publication Reddit** avec un petit budget pour la booster temporairement. Cela la pousse plus haut dans les résultats de recherche, permet à Google de l'indexer plus vite, et lui donne une visibilité initiale. Demandez à votre LLM comment fonctionne la promotion de publications Reddit et quel budget est pertinent.

> **L'astuce de l'entonnoir.** Sur votre page d'accueil ou landing page, incluez quelque chose qui **attire les gens indépendamment de votre produit principal** — un outil gratuit, un sujet tendance, des informations utiles, ou quelque chose qui est actuellement populaire. Cela agit comme un **entonnoir** : les gens arrivent pour le contenu gratuit/intéressant, découvrent votre produit, et un pourcentage d'entre eux convertit. Considérez-le comme un appât qui apporte une vraie valeur tout en menant vers ce que vous vendez.

SEO, contenu & distribution

Avant de commencer tout marketing, mettez en place votre **analyse et suivi**. Vous avez besoin de deux choses :

- **Google Analytics** — ajoutez le code de suivi à votre site (demandez à Claude de l'intégrer). Il existe aussi une **application mobile** pour que vous puissiez vérifier votre trafic depuis votre téléphone à tout moment. Cela vous montre les visites totales, le comportement des utilisateurs, les sources de trafic et les données de conversion.
- **Google Search Console** — configurez-la et connectez-la à Google Analytics. Search Console suit spécifiquement votre **trafic organique Google** : quelles requêtes de recherche amènent des gens sur votre site, à quelle fréquence vous apparaissez dans les résultats de recherche, et vos taux de clics. Demandez à votre LLM comment configurer et connecter les deux outils.

Si vous avez du budget disponible, **Google Ads** peut vous donner un coup de pouce initial. Lancer des publicités brièvement aide à construire la confiance avec l'algorithme de Google et génère du trafic précoce pendant que votre SEO organique se développe. Mais les coûts s'accumulent vite, donc traitez-le comme un accélérateur à court terme, pas une stratégie à long terme. Votre LLM peut expliquer la configuration et le budget Google Ads en détail.

Pour le SEO en lui-même, commencez par les bases. Ouvrez les **DevTools** de votre navigateur (F12), allez dans **Lighthouse**, et générez un rapport. Cela vous donne des scores pour la Performance, l'Accessibilité, les Bonnes Pratiques et le **SEO**. Corrigez ce qu'il vous dit de corriger — et oui, vous pouvez demander à Claude Code de s'en charger.

Actions SEO clés à entreprendre :

- **Ajoutez des traductions** à vos pages. Le support multilingue augmente considérablement votre portée. Demandez à Claude Code de mettre en place l'internationalisation pour votre projet.
- **Ajoutez les méta-tags appropriés** — titre, description, balises Open Graph pour le partage social, données structurées. Ceux-ci affectent directement la façon dont votre site apparaît dans les résultats de recherche Google.

- **Créez et soumettez un sitemap.** Générez un sitemap à jour et uploadez-le sur Google Search Console. Cela dit à Google exactement quelles pages existent sur votre site et les aide à être indexées plus rapidement.

> **Le SEO demande de la patience.** N'attendez pas de trafic organique du jour au lendemain — il faut des semaines ou des mois pour que Google indexe et classe correctement vos pages. Mais quand ça décolle, c'est du **trafic gratuit qui continue d'arriver** sans que vous dépensiez un centime. Les clics pour lesquels vous paieriez autrement des centaines d'euros via Google Ads arrivent gratuitement par la recherche organique. En attendant, travaillez sur les plateformes sociales comme Reddit pour construire la visibilité initiale et les backlinks. Le SEO est un jeu de longue haleine, mais c'est l'investissement marketing le plus précieux que vous puissiez faire.

La tendance émergente : le trafic généré par l'IA

Il existe une nouvelle source de trafic en croissance rapide que la plupart des gens n'ont pas encore remarquée : **les LLM qui recommandent votre site.** ChatGPT, Gemini, Claude et d'autres assistants IA sont de plus en plus utilisés comme moteurs de recherche. Quand quelqu'un demande *« Quel est un bon site pour [votre catégorie de produit] ? »*, ces modèles peuvent et recommandent effectivement des sites spécifiques — et cela génère du vrai trafic. Une part significative de nos propres visites provient de ChatGPT et d'autres LLM.

Pour en profiter, allez dans votre **tableau de bord Cloudflare** et assurez-vous de **désactiver l'option qui bloque les crawlers IA**. De nombreux sites bloquent les bots IA par défaut, ce qui empêche les LLM d'apprendre l'existence de votre contenu. En autorisant les crawlers IA à accéder à votre site, vous permettez à ces modèles d'indexer vos pages, de comprendre ce que vous offrez, et de potentiellement vous recommander aux utilisateurs à l'avenir. C'est essentiellement du **SEO gratuit pour l'ère de l'IA** — et le coup de pouce peut être énorme.

> **Pensez au-delà de Google.** Le SEO traditionnel cible la recherche Google. Mais les recommandations par l'IA deviennent une source de trafic majeure — et cette tendance ne fait que s'accélérer. Assurez-vous que votre site est accessible aux crawlers IA, a un contenu clair et descriptif, et est bien structuré pour que les LLM puissent facilement comprendre ce que vous offrez. Les sites qui se positionnent maintenant pour la découverte par l'IA auront un avantage massif à mesure que ce canal se développe.

Merci !

Merci d'avoir acheté ce cours et de nous avoir fait confiance avec votre temps et votre argent. Nous espérons sincèrement que vous l'avez trouvé utile et qu'il vous aide à construire quelque chose de concret.

Nous serions ravis d'avoir de vos nouvelles. **Rejoignez notre communauté Discord** sur apifreellm.com — c'est là que nous traînons, partageons des mises à jour et nous entraidons.

Si vous avez un moment, **écrivez-nous un avis** et dites-nous ce que vous en pensez. Qu'avez-vous trouvé le plus utile ? Qu'est-ce qui manquait ? Qu'est-ce qui pourrait être amélioré ? Nous sommes sincèrement intéressés par vos retours — ce cours est un projet vivant et nous voulons continuer à l'améliorer en fonction de ce dont **vous** avez réellement besoin.

On se retrouve sur Discord. Maintenant allez construire quelque chose d'incroyable.