Eigenflow

GUIDE · 14 NOVEMBRE 2026

Cinq outils pour coder comme un développeur senior avec Claude Code

Agent Skills, Context7, Graphify, le MCP de 21st.dev et Strix : ce que fait vraiment chacun, l'installation exacte, ce qui reste payant, notre test et la loi

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté SENIOR, voici les liens des cinq outils de la vidéo :

- Agent Skills : https://github.com/addyosmani/agent-skills
- Context7 : https://github.com/upstash/context7 et le site https://context7.com
- Graphify : https://github.com/Graphify-Labs/graphify
- Le MCP de 21st.dev : https://21st.dev/mcp et le dépôt https://github.com/21st-dev/magic-mcp
- Strix : https://github.com/usestrix/strix

Pour chacun : ce qu'il est vraiment (tous ne sont pas des plugins au sens de Claude Code), à quoi il te sert, la commande d'installation copiée de sa doc, la première chose à lui demander et ce qui se paie. Pour Context7, on te raconte notre test. Pour Strix, on passe par la loi avant de lancer quoi que ce soit.

Dépôts, docs, pages de prix et Légifrance relus le 14 novembre 2026, entre 5 h 28 et 5 h 35 (heure de Paris). On n'a installé ni lancé aucun des cinq outils ; seul Context7 a servi, pour une recherche, par le connecteur déjà branché chez nous. Aucun lien de ce guide n'est affilié.

À savoir avant de lire

- Claude Code est payant. Sur la page française des tarifs d'Anthropic, le 14/11/2026, il n'est pas dans l'offre Free et il est inclus dès Pro : 18 € par mois, ou 15 € par mois à l'année (180 € facturés d'avance). Ces prix sont hors taxes (« Les prix affichés n'incluent pas les taxes applicables »), soit 21,60 € TTC par mois avec la TVA à 20 %. Claude Code puise dans le même quota que tes conversations.
- Dans Claude Code, un plugin est un paquet de skills, d'agents, de hooks ou de serveurs MCP qui s'installe d'un bloc avec la commande /plugin. Agent Skills, Context7 et 21st en sont. Graphify et Strix sont des programmes à installer dans ton terminal, qui ajoutent des skills à Claude Code.
- Les cinq ont une version gratuite, avec des limites : 1 000 appels par mois pour Context7, 2 composants par jour pour 21st. Strix fait tourner un modèle d'IA que tu paies à l'usage, sauf avec un abonnement ChatGPT ou un modèle installé sur ta machine.
- Un plugin peut lancer des commandes sur ton ordinateur. La doc de Claude Code demande de lire sa fiche avant de l'installer. Essaie chaque outil sur un projet de test d'abord.
- Strix attaque vraiment ce que tu lui donnes. Ton code, tes propres sites, ou un site pour lequel tu as une autorisation écrite, rien d'autre (partie 5).
- Ce qu'il faut sur ta machine selon l'outil : Node.js pour les commandes npx et npm, Python 3.10 ou plus récent et uv pour Graphify, Docker pour Strix.

1. Agent Skills : la méthode d'un développeur senior, en 25 skills

Ce que c'est

Agent Skills est un vrai plugin Claude Code, sous licence MIT, avec 117 206 étoiles sur GitHub le 14/11. Il contient 25 skills, des fichiers de méthode que Claude lit quand il en a besoin : 24 pour les étapes du développement, plus `using-agent-skills`, qui aiguille vers les autres. La vidéo dit vingt-quatre, le README aussi (« 24 lifecycle skills plus the using-agent-skills meta-skill »). S'y ajoutent 9 commandes, 4 agents relecteurs (code, tests, sécurité, performance web) et 7 aide-mémoire.

Son créateur, Addy Osmani, a passé plus de 14 ans chez Google, en dernier comme directeur chez Google Cloud AI. Sa page le présente aujourd'hui ainsi : « Member of Technical Staff at Anthropic. He works on Claude Code. » Le README dit reprendre les pratiques d'ingénierie de Google, dont le livre Software Engineering at Google.

À quoi ça te sert

Le README part d'un constat : un agent de code prend le chemin le plus court, et saute souvent la spécification, les tests et la relecture de sécurité. Les skills le font travailler dans l'ordre d'un développeur expérimenté. Chaque étape a sa commande :

- Définir : `/spec` écrit la spécification avant la première ligne de code. La skill interview-me te pose une question à la fois jusqu'à savoir ce que tu veux vraiment.
- Planifier : `/plan` découpe le travail en petites tâches vérifiables.
- Construire : `/build` code une tranche à la fois, tests d'abord.
- Vérifier : `/test`, avec le débogage et les tests dans le navigateur.
- Relire : `/review` (code, sécurité, performance), `/code-simplify` pour simplifier, `/webperf` pour la vitesse d'une page.
- Livrer : `/ship`, avec git, la doc et le suivi en production.

« Active automatiquement » : qui décide

La vidéo dit que la bonne compétence s'active toute seule. Dans Claude Code, c'est Claude qui choisit : il lit la description de chaque skill et charge celle qui colle à ta demande (« Claude uses skills when relevant », dit la doc de Claude Code). Le plugin n'ajoute pas d'aiguillage maison. Son script de démarrage le précise en tête, « Not wired by the plugin », pour ne pas doubler celui de Claude Code.

Rien ne garantit donc que la bonne skill part à chaque fois, et on ne l'a pas testé. Pour être sûr, tape la commande de l'étape.

Autre détail de la doc de Claude Code : le nom et la description de chaque skill que Claude peut lancer seul restent dans son contexte à chaque échange, même quand rien ne sert. Avec 25 skills, chaque message coûte un peu plus de quota.

L'installer

Dans Claude Code, copié du README (section « Claude Code (recommended) ») :

À COLLER

```
/plugin marketplace add addyosmani/agent-skills
/plugin install agent-skills@addy-agent-skills
```

La première ligne ajoute la place de marché du créateur, la seconde installe le plugin. Claude Code t'ouvre alors la fiche du plugin : lis-la et choisis où l'installer (pour toi seul ou pour ce projet). Si l'ajout échoue sur une erreur SSH, le README donne l'adresse en HTTPS :

À COLLER

```
/plugin marketplace add https://github.com/addyosmani/agent-skills.git
/plugin install agent-skills@addy-agent-skills
```

La première chose à lui demander

Ouvre Claude Code dans le dossier de ton projet et commence par la spécification, avec tes mots. Par exemple :

À COLLER

```
/spec Une page où mes clients réservent un rendez-vous de 30 minutes et
reçoivent un e-mail de confirmation.
```

Si tu ne sais pas encore ce que tu veux, demande-lui « interview me » : la skill te questionne une question à la fois. Une fois la spécification validée, `/build auto` écrit le plan puis code toutes les tâches à la suite, après une seule validation de ta part. Chaque tâche reste testée et enregistrée à part dans git, et il s'arrête sur un échec ou une étape risquée, dit le README.

Un piège : `/review` existe déjà dans Claude Code. Pour celle du pack, tape `/agent-skills:review`.

Limites et ce qui se paie

- Gratuit, licence MIT. Tu ne paies que ton abonnement Claude.
- Plus de méthode veut dire plus d'étapes : spécification, tests, relecture. Compte plus de temps et plus de quota que pour un « fais-moi un site ».
- `/build auto` et `/ship` enregistrent des commits. Apprends les bases de git avant de les lancer sur un vrai projet.

Source : README de addyosmani/agent-skills et page https://addyosmani.com relus le 14/11/2026 à 5 h 28.

---

2. Context7 : la documentation du jour dans ta conversation

Ce que c'est

Context7 est un service d'Upstash. La pièce principale est un serveur MCP hébergé, un connecteur au sens de Claude Code, qui donne deux outils à Claude : `resolve-library-id` trouve la bibliothèque dont tu parles, `query-docs` va chercher la page de doc et les exemples de code qui répondent à ta question. Il existe aussi en plugin Claude Code, qui branche ce serveur, et en version « CLI + skill » avec la commande `ctx7`. Il figure dans la place de marché officielle d'Anthropic.

Le code du serveur MCP est sous licence MIT (64 770 étoiles le 14/11). Le moteur qui lit et range les docs est privé, précise le README.

À quoi ça te sert

Claude apprend sur des données arrêtées à une date, et tes bibliothèques changent de version sans arrêt. Sans doc à jour, il peut t'écrire du code pour une version d'avant, ou appeler une fonction qui n'existe plus. Avec Context7, Claude lit la doc de la version que tu utilises avant de coder.

Notre test : « 16 » contre v16.4.5

Le 14 novembre à 5 h 04, on a posé une question simple à Claude seul : Claude Code 2.1.325, modèle Opus 5.5, dans un dossier vide, sans outil de recherche. « Quelle est la dernière version stable de Next.js ? Réponds par le numéro de version seul, sans rien chercher. »

Sa réponse : « 16 ». Suivie d'une précision honnête : « Ma réponse se fonde sur mes connaissances, qui s'arrêtent à mi-2026. Une version plus récente a pu sortir depuis. »

À 5 h 15, on a cherché Next.js dans Context7, par le connecteur déjà branché chez nous. Réponse : la bibliothèque /vercel/next.js, 7 338 exemples de code, des versions listées jusqu'à la v16.4.5.

Puis on a regardé npm, le registre officiel des paquets JavaScript. La dernière version stable de Next.js y est la 16.4.6, publiée le 9 novembre 2026 à 21 h 33 (heure de Paris). La 16.4.5 date du 3 novembre.

Claude seul connaît la version majeure et prévient qu'il peut avoir du retard. Context7 donne le numéro précis, avec cinq jours de retard sur le dernier correctif. On a revérifié à 5 h 32 sur sa page Next.js, mise à jour « 5 hours ago » : la liste s'arrêtait toujours à v16.4.5. Pour le numéro exact du jour, la source reste le registre du paquet, npm pour JavaScript, PyPI pour Python.

La vidéo promet que tu ne coderas « plus jamais » avec des versions périmées. Context7 réduit beaucoup l'écart, sans le supprimer. Son README prévient lui-même que les docs viennent de la communauté et qu'il ne peut pas en garantir l'exactitude.

L'installer

Le plus simple, sans rien installer sur ton ordinateur : le plugin. Dans Claude Code (doc de Context7, page Claude Code) :

À COLLER

```
/plugin marketplace add upstash/context7
/plugin install context7@context7-marketplace
```

Redémarre Claude Code, tape `/mcp`, choisis Context7 et connecte-toi dans le navigateur. « No API key is required », dit la doc.

Autre voie, si tu as Node.js 22.13 ou plus récent : la commande de configuration, dans ton terminal.

À COLLER

```
npx ctx7 setup --claude
```

Elle te connecte, crée une clé d'API gratuite et installe une skill qui se déclenche dès que tu parles d'une bibliothèque, sans avoir à écrire « use context7 ».

Avec le plugin, ajoute la règle proposée par le README dans le fichier CLAUDE.md de ton projet, pour que Claude pense à Context7 sans que tu le demandes :

À COLLER

```
Always use Context7 when I need library/API documentation, code generation,
setup or configuration steps without me having to explicitly ask.
```

La première chose à lui demander

Un exemple de la doc de Context7, en français :

À COLLER

```
use context7 : montre-moi comment configurer un proxy dans Next.js 16
```

Si tu connais l'identifiant de la bibliothèque, donne-le : Context7 saute l'étape de recherche, et ça fait un appel de moins.

À COLLER

```
use context7 with /supabase/supabase : comment créer un compte avec e-mail et
mot de passe ?
```

Limites et ce qui se paie

- Offre gratuite, relue sur context7.com/plans le 14/11 : 0 $, dépôts publics, 1 000 appels par mois. Une fois la limite atteinte, tu es bloqué, avec 20 appels de secours par jour jusqu'au mois suivant. Une recherche complète compte deux appels (trouver la bibliothèque, puis lire

la doc), donc environ 500 recherches par mois. C'est notre calcul, pas un chiffre de Context7.
- Pro : 10 $ par siège et par mois (9,05 € au cours du 13/11), 2 000 appels inclus, puis 5 $ les 1 000. Lire la doc de tes dépôts privés demande Pro, à 5 $ par million de jetons.
- Ce qui part chez Context7 : d'après sa FAQ, ta question de doc et le nom de la bibliothèque, pas ton code. Évite quand même d'y glisser un nom de client ou un mot de passe.

Source : README de upstash/context7, pages Claude Code et plans de Context7, registre npm, relus le 14/11/2026 entre 5 h 29 et 5 h 32.

---

3. Graphify : une carte de ton code que Claude consulte avant de relire

Ce que c'est

Graphify est un programme en ligne de commande écrit en Python, paquet `graphifyy` (avec deux y), sous licence Apache 2.0, avec 147 348 étoiles le 14/11. Il ne s'installe pas avec /plugin : la commande `graphify install` ajoute une skill à Claude Code, et `graphify claude install`, lancée dans ton projet, ajoute les hooks qui poussent Claude vers le graphe. L'éditeur vend aussi une version hébergée, branchée par MCP. Ce guide parle de la version gratuite, sur ta machine.

À quoi ça te sert

Sur un projet qui grossit, Claude passe du temps et du quota à fouiller les fichiers pour comprendre qui fait quoi. Graphify lit ton code une fois et en tire un graphe : quelle fonction en appelle une autre, quel fichier importe quel autre, quelle classe hérite de laquelle, dans une quarantaine de langages. Tu obtiens un dossier graphify-out/ avec trois fichiers :

- graph.html, la carte de ton projet, cliquable dans ton navigateur ;
- GRAPH_REPORT.md, le résumé : les concepts clés, les liens surprenants, des questions à poser ;
- graph.json, le graphe complet, que Claude interroge au lieu de relire tes fichiers.

Chaque lien porte une étiquette : EXTRACTED quand il est écrit dans le code, INFERRED quand Graphify l'a déduit. Un lien déduit peut être faux, garde-le en tête quand Claude s'appuie dessus.

Sur Claude Code, un hook se déclenche avant que Claude cherche ou lise un fichier source, et le pousse vers le graphe. Il peut toujours lire tes fichiers ; il commence par la carte. C'est ce que la vidéo veut dire par « pas besoin de tout relire encore et encore ». À une condition : le graphe doit être à jour. `graphify hook install` le reconstruit à chaque commit et à chaque changement de branche ; après un `git pull`, lance `graphify update .`.

Le README met en avant « 80 % » de jetons en moins. Ce chiffre parle de Graphify Cloud, l'offre hébergée. Pour la version locale, la doc écrit elle-même « Headline numbers aren't your numbers » et te donne `graphify benchmark` pour mesurer sur ton propre projet.

L'installer

Il te faut Python 3.10 ou plus récent et uv (ou pipx). Sur Mac, la doc de Graphify installe uv avec Homebrew :

À COLLER

```
brew install uv
```

Puis, dans ton terminal, copié du README :

À COLLER

```
uv tool install graphifyy      # install the CLI (or: pipx install graphifyy)
graphify install               # detects your assistant and registers the skill
```

Attention au nom, avec deux y : le README prévient que les autres paquets graphify de PyPI n'ont rien à voir avec l'éditeur. Si le terminal ne trouve pas la commande graphify après l'installation, la doc donne `uv tool update-shell`, puis un nouveau terminal.

Ensuite, dans Claude Code ouvert dans ton projet, construis le graphe :

À COLLER

```
/graphify .
```

Une fois le graphe construit, lance une fois dans le dossier du projet la commande qui fait consulter le graphe d'office (section « Make your assistant always use the graph » du README) :

À COLLER

```
graphify claude install
```

La première chose à lui demander

Une fois le graphe construit, l'exemple de la doc Claude Code de Graphify, en français :

À COLLER

```
Utilise Graphify pour m'expliquer comment marche la connexion des utilisateurs.
Cite les fichiers sources et distingue les liens écrits dans le code des liens
déduits.
```

Ouvre aussi graphify-out/graph.html dans ton navigateur : quand Claude a écrit la moitié de ton projet, la carte aide à comprendre ce que tu as entre les mains.

Limites et ce qui se paie

- Le code est lu sur ta machine, sans modèle d'IA, sans clé et sans compte : c'est gratuit, et « Nothing leaves your machine », dit le README pour le code.
- Les docs, PDF et images passent par un modèle : celui de ta session Claude Code (donc ton quota) ou une clé d'API. La doc prévient : « A README and a folder of screenshots are enough to trigger it. » Pour ne lire que le code : `graphify extract . --code-only`.
- Chaque question au graphe est notée dans un journal sur ta machine, ~/.cache/graphify-queries.log. Pas de télémétrie, d'après le README.
- Version hébergée : gratuite jusqu'à 3 dépôts, sans carte ; Pro à 15 $ par mois, ou 10 $ par mois à l'année (120 $). Un espace gratuit se met en pause après 7 jours sans usage.

Source : README de Graphify-Labs/graphify, docs.graphify.com (installation, Claude Code, réduire les jetons, prix), relus le 14/11/2026 à 5 h 31.

---

4. Le MCP de 21st.dev : des composants d'interface prêts à poser

Ce que c'est

21st.dev est une bibliothèque de composants d'interface : tableaux de prix, formulaires, menus. Sa page annonce « 12,000+ components », chacun publié par son auteur. Le 21st MCP, ancien Magic MCP, est un serveur MCP hébergé qui donne à Claude de quoi chercher dans cette bibliothèque et récupérer le code d'un composant. Il existe aussi en plugin Claude Code et en ligne de commande, `21st`. Le dépôt historique, 21st-dev/magic-mcp, compte 6 482 étoiles (licence ISC).

Les composants sont écrits en React, avec Tailwind et shadcn, d'après les mots-clés du plugin. Si ton site tourne sous WordPress ou Wix, cet outil ne te servira pas.

À quoi ça te sert

Pour une page d'accueil, un tableau de prix ou un tableau de bord, Claude part d'un composant déjà dessiné au lieu de l'inventer. L'exemple de la page 21st.dev/mcp : tu demandes un tableau de prix avec un bouton mensuel/annuel ; Claude cherche dans 21st, choisit un composant, l'écrit dans components/ui/pricing.tsx, installe ses dépendances, remplace les couleurs par les tiennes et vérifie que le projet compile.

L'installer

Il te faut un compte 21st (gratuit, sans carte) et ta clé d'API, affichée sur https://21st.dev/mcp une fois connecté. Ensuite, dans ton terminal, lance la commande de l'onglet « Claude » de cette page :

À COLLER

```
claude mcp add --transport http 21st https://21st.dev/api/mcp --header "x-api-key: $API_KEY_21ST"
```

Remplace $API_KEY_21ST par ta clé, ou range ta clé dans une variable d'environnement de ce nom. Vérifie avec `claude mcp list` : 21st doit apparaître connecté.

Autre voie, sur la même page : le plugin, qui ajoute le serveur et les skills de 21st d'un coup. Son README demande d'exporter la clé dans API_KEY_21ST avant.

À COLLER

```
/plugin marketplace add 21st-dev/claude-code-plugin
/plugin install 21st@21st
```

Et la voie en ligne de commande, celle de la « seule commande » de la vidéo :

À COLLER

```
npm i -g @21st-dev/cli
21st login
21st search "pricing table"
21st add efferd/pricing
```

La dernière ligne ajoute un composant à ton projet en une commande, une fois le CLI installé et la connexion faite.

La première chose à lui demander

À COLLER

```
ajoute un tableau de prix avec un bouton mensuel/annuel depuis 21st
```

Sur le compte gratuit, ce composant compte pour l'un de tes 2 du jour. La recherche ne coûte rien : demande d'abord à Claude de te montrer ce que 21st propose, puis ne récupère que celui que tu gardes.

Limites et ce qui se paie

- Compte gratuit, relu sur 21st.dev/pricing le 14/11 : recherche gratuite, 2 composants par jour (site, MCP et CLI confondus), 5 revues de l'outil Design Bug Bot.
- Builder : 8 $ par mois facturé au trimestre (24 $), ou 6 $ par mois facturé à l'année (72 $). Copies illimitées. En euros : 7,24 € ou 5,43 € par mois au cours du 13/11.
- Builder + AI : 20 $ par mois au trimestre, dès 15 $ par mois à l'année, avec 500 crédits d'IA par mois pour générer des composants. Sans cette offre, l'outil de génération n'apparaît pas.
- « Payments are non-refundable », dit la page.
- Le code posé dans ton projet t'appartient (« yours to edit »). Il vient d'auteurs différents : relis-le comme tout code que tu n'as pas écrit.

Source : README de 21st-dev/magic-mcp et de 21st-dev/claude-code-plugin, pages mcp et pricing de 21st.dev, relus le 14/11/2026 entre 5 h 29 et 5 h 35.

5. Strix : un test d'intrusion par des agents IA

Ce que c'est

Strix est un outil de test d'intrusion open source, sous licence Apache 2.0, avec 75 428 étoiles le 14/11. Il lance une équipe d'agents IA dans un conteneur Docker sur ton ordinateur : ils lisent ton code, font tourner ton appli, essaient de l'attaquer et prouvent chaque faille par une démonstration qui marche. Il s'installe dans ton terminal. Pour que Claude Code sache le piloter, Strix fournit 9 skills.

À quoi ça te sert

Avant de mettre en ligne une appli écrite avec Claude, tu veux savoir ce qu'un attaquant pourrait en faire. Strix cherche les failles de la liste OWASP et au-delà : lire les données d'un autre client en changeant un numéro dans l'adresse, injections SQL, failles XSS, connexion contournée, paiement manipulé. Les résultats arrivent dans strix_runs/ : un fichier par format (JSON, CSV, SARIF) et un Markdown par faille. `strix view` les affiche dans un tableau de bord local.

Trouver, puis corriger : qui fait quoi

La version gratuite trouve, prouve, et donne pour chaque faille comment la corriger, avec un correctif proposé (« actionable findings with remediation guidance », « generate patches »). La correction en un clic, qui ouvre une demande de fusion prête à valider, appartient à Strix Cloud, payant. Dans Claude Code, avec les skills de Strix, c'est Claude qui corrige : la skill fix-security-vulnerabilities-with-strix trie les failles, corrige la cause et relance Strix pour vérifier chaque correction.

Un point à connaître avant le premier lancement. Quand tu donnes un dossier à Strix, il le monte en écriture, et ses agents modifient tes vrais fichiers. Sa doc le dit : « Commit or stash first. » Enregistre ton travail dans git avant chaque analyse.

L'installer

Il te faut Docker, lancé, et une clé d'API chez un fournisseur de modèle. Le README, copié tel quel :

À COLLER

```
# Install Strix
curl -sSL https://strix.ai/install | bash

# Configure your AI provider
export STRIX_LLM="vercel_ai_gateway/zai/glm-5.3"   # Vercel AI Gateway
# export STRIX_LLM="openrouter/z-ai/glm-5.3"       # OpenRouter
export LLM_API_KEY="your-api-key"

# Run your first security assessment
strix --target ./app-directory
```

La première ligne télécharge un script et l'exécute. Ouvre https://strix.ai/install dans ton navigateur pour le lire avant, ou passe par `pipx install strix-agent`, l'autre voie de la doc.

Puis les skills, pour Claude Code :

À COLLER

```
npx skills add usestrix/strix
```

La doc de Strix ne prévoit pas ton abonnement Claude. Avec Claude, il faut une clé d'API, et en plus le Cyber Verification Program d'Anthropic : sans lui, dit la doc de Strix, « the safety classifiers block most offensive tasks ». Le modèle conseillé par Strix est GLM-5.3, via Vercel AI Gateway ou OpenRouter. Sans facture à l'usage : ton abonnement ChatGPT Plus ou Pro (README), ou un modèle sur ta machine (Ollama, LM Studio, dans la doc).

À COLLER

```
strix auth login chatgpt             # sign in with your ChatGPT account
export STRIX_LLM="chatgpt/gpt-5.4"   # chatgpt/<model> runs on the subscription
strix auth status                    # show the active sign-in, or logout to
forget it
```

La première chose à lui demander

Dans Claude Code, une fois les skills installées, l'exemple de la doc de Strix, en français :

À COLLER

```
Teste ce dépôt avec Strix (mode quick, budget de 10 $) et résume ce que tu
trouves.
```

S'il trouve des failles :

À COLLER

```
Corrige toutes les failles critiques et élevées de la dernière analyse Strix,
puis relance une analyse pour vérifier.
```

Ce que coûte une analyse

Il n'y a pas de prix fixe : tu paies à ton fournisseur les jetons consommés par le modèle, selon la taille de ton projet et la profondeur de l'analyse. Le mode quick prend quelques minutes, standard de 30 minutes à 1 heure, deep de 1 à 4 heures. Deep est le mode par défaut. Pour un premier essai, précise toujours `--scan-mode quick`.

Fixe un plafond en dollars avec `--max-budget`. La doc donne un exemple à 25 $: `strix --target https://example.com --max-budget 25 --max-turns 300`. Le compteur est une estimation tirée des jetons, et la dépense finale peut dépasser un peu le plafond. Mets aussi une limite de dépense chez ton fournisseur de modèle.

Pour un ordre d'idée du prix des jetons, GLM-5.3 coûtait sur OpenRouter, le 14/11, 0,084 $ par million de jetons envoyés et 3,60 $ par million de jetons générés. On n'a pas lancé Strix, donc on n'a pas de coût mesuré à te donner.

Strix Cloud, la version hébergée, coûte 29 $ par siège et par mois (26,24 € au cours du 13/11), avec les tests facturés à part et 7 jours d'essai.

Le cadre légal

Strix attaque pour de vrai ce que tu lui désignes. Son README le rappelle en avertissement : « Authorized use only. Strix actively tests the targets you point it at, so only run it against systems you own or have explicit, written permission to test, and stay within the agreed scope. »

En France, le Code pénal punit l'intrusion dans un système informatique. Article 323-1, version en vigueur depuis le 26 janvier 2023 (loi du 24 janvier 2023), relu sur Légifrance le 14/11/2026 :

- accéder ou se maintenir frauduleusement dans tout ou partie d'un système de traitement automatisé de données : trois ans d'emprisonnement et 100 000 € d'amende ;
- s'il en résulte la suppression ou la modification de données, ou une altération du fonctionnement du système : cinq ans et 150 000 € ;
- contre un système de données personnelles mis en œuvre par l'État : sept ans et 300 000 €.

L'article 323-2 punit le fait d'entraver ou de fausser le fonctionnement d'un système : cinq ans et 150 000 €. Une analyse lancée sur le site d'un autre peut le ralentir. L'article 323-3-1 punit aussi la détention d'un outil conçu pour ces attaques quand elle se fait « sans motif légitime » ; le texte cite la recherche et la sécurité informatique comme motifs légitimes.

En pratique :

- ton propre code, sur ta machine ou dans ton dépôt : oui ;
- ton site en ligne : oui s'il est à toi, plutôt sur une copie de test que sur le site où passent tes clients ;
- le site d'un client : seulement avec une autorisation écrite qui dit quoi tester, jusqu'où et quand ;
- le site de quelqu'un d'autre, pour voir : non.

Ce guide n'est pas un avis juridique. Avant une mission chez un client, fais relire l'autorisation.

Source : README de usestrix/strix, docs.strix.ai (démarrage, CLI, modes, fournisseurs, agents de code), strix.ai/pricing, API publique d'OpenRouter, Légifrance (Code pénal, articles 323-1 à 323-3-1), relus le 14/11/2026 entre 5 h 29 et 5 h 33.

---

6. Dans quel ordre les installer

1. Context7 d'abord : un plugin, une connexion dans le navigateur, rien à installer sur ta machine. Il sert dès ta première question sur une bibliothèque.

2. Agent Skills quand tu attaques un vrai projet, pour que la première demande soit un `/spec` et pas une ligne de code.

3. 21st si ton projet est en React et que tu travailles l'interface. Le compte gratuit suffit pour essayer, à 2 composants par jour.

4. Graphify quand le projet dépasse quelques dizaines de fichiers et que Claude commence à fouiller partout.

5. Strix en dernier, avant la mise en ligne : sur ton code, travail enregistré dans git, en mode quick, avec un budget fixé.

---

7. La check-list

- Un abonnement Claude payant (Pro au minimum) et Claude Code installé.
- Un projet de test, enregistré dans git, avant d'installer quoi que ce soit.
- La fiche de chaque plugin lue avant l'installation.
- Context7 : plugin installé, connexion faite par /mcp, règle ajoutée dans CLAUDE.md.
- Agent Skills : plugin installé, et ta première demande commence par /spec.
- Graphify : Python 3.10 ou plus, uv, le paquet graphifyy (deux y), `graphify install` puis `graphify claude install` dans le projet, `/graphify .` lancé et graph.html ouvert une fois.
- 21st : compte créé, clé rangée dans API_KEY_21ST, 21st connecté dans `claude mcp list`. Tu sais que le gratuit s'arrête à 2 composants par jour.
- Strix : Docker lancé, une clé d'API avec une limite de dépense chez le fournisseur, travail enregistré dans git, `--scan-mode quick` et `--max-budget` à chaque essai.
- Strix seulement sur ton code, ou avec une autorisation écrite.

---

8. La méthode

Guide écrit le 14 novembre 2026. Dépôts lus par l'API de GitHub à 5 h 28 (étoiles, licences, README, manifestes de plugins) ; docs officielles, pages de prix, registre npm, API d'OpenRouter et Code pénal sur Légifrance relus jusqu'à 5 h 35 (heure de Paris). Dollars convertis au cours de référence de la BCE du 13 novembre 2026 (1 € = 1,1052 $) ; les pages en dollars ne disent pas si les taxes sont comprises.

Aucun des cinq outils n'a été installé ni lancé. Seul test : la question sur Next.js de la partie 2, posée à Claude Code 2.1.325 (Opus 5.5, sur abonnement, sans clé API, dossier vide, --safe-mode), puis à Context7 par son connecteur déjà branché, et comparée au registre npm. Commandes copiées des README et des docs du jour ; si un détail ne colle plus, les pages officielles font foi.