Eigenflow

GUIDE · 13 NOVEMBRE 2026

Le setup épuré en 3 prompts

Lire ton setup Claude Code, trier tes skills, écrire tes modèles et tes préférences : les trois prompts testés sur notre setup, avec ce que Claude nous a rendu

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté SETUP, voici le guide complet : les trois prompts de la vidéo, prêts à copier, et ce qu'ils ont donné quand on les a lancés sur notre propre Claude Code, le 13 novembre 2026.

Le premier fait l'audit de ton setup et range chaque skill, chaque plugin et chaque fichier de consignes dans une colonne : garder, fusionner ou couper. Le deuxième écrit un modèle pour chaque tâche qui revient. Le troisième t'interviewe et note tes préférences en quinze lignes au plus. Avant eux, deux commandes pour lire ton setup toi-même. Après, ce qu'Anthropic conseille de laisser tomber, une grille pour trancher seul et de quoi garder le tout propre.

On a relu les pages officielles le 13 novembre, avec Claude Code 2.1.280 installé. La doc décrivait déjà des fonctions de la version 2.1.305 : si une commande a bougé le jour où tu lis, la doc officielle a raison.

Eigenflow n'a aucun lien avec Anthropic ni avec Vercel. Pas d'affiliation, pas de partenariat.

À savoir avant de lire

- Une skill coûte avant même de servir. Son nom et sa description partent dans le contexte à chaque message ; le reste du fichier ne se charge que quand elle tourne. La doc de Claude Code l'écrit ainsi : « Every skill in the skill listing adds to your context on every turn, whether or not Claude ever uses it. »
- La liste des skills a un budget : 1 % de la fenêtre de contexte du modèle. Quand elle déborde, Claude Code retire des descriptions, en commençant par les skills que tu appelles le moins. Il ne reste que le nom, et la doc prévient que ça « retire les mots-clés dont Claude a besoin pour reconnaître ta demande ». Chez nous, 224 skills sur 260 n'avaient plus que leur nom.
- Les fichiers de consignes pèsent aussi. La page mémoire d'Anthropic vise moins de 200 lignes par CLAUDE.md : « Longer files consume more context and reduce adherence. » Boris Cherny, le créateur de Claude Code, est allé plus loin dans une vidéo de Y Combinator publiée le 27 juillet 2026 : tous les six mois, supprime ton CLAUDE.md, tes skills et tes hooks, regarde ce que le modèle fait sans eux, et ne remets que ce qui manque.
- Le chiffre de Vercel, « In 56% of eval cases, the skill was never invoked », date du 27 janvier 2026 et vient d'un cas précis : une seule skill (la doc de Next.js 16), sur leurs propres évals, sans consigne qui demande de l'utiliser. L'agent testé n'est pas nommé. Avec une ligne de consigne, la skill était appelée dans plus de 95 % des cas.
- Ne supprime pas tout. Anthropic conseille toujours les skills pour tes procédures à toi, celles que Claude ne peut pas deviner, et précise qu'une skill pèse moins que la même procédure collée dans CLAUDE.md. Le but est de garder ce qui sert, et seulement ça.
- Aucun prompt de ce guide n'efface de fichier. Le premier rend une liste et des commandes, et c'est toi qui valides, ligne par ligne.

0. Lire ton setup : /skill-doctor et /context

Commence par regarder ce que tu as. Deux commandes de Claude Code suffisent, et aucune ne modifie quoi que ce soit.

/context : ce qui est chargé avant ton premier message

Dans une session, tape /context. Tu vois, catégorie par catégorie, combien de tokens sont déjà pris avant que tu aies écrit un mot : prompt système, agents, fichiers de mémoire, skills, messages. Ce socle repart avec chaque requête. Le cache le rend moins cher à relire, mais il compte dans ton usage.

À COLLER

```
/context
```

Pour garder le résultat dans un fichier texte, à donner ensuite au prompt 1, lance-le depuis le terminal, dans le dossier où tu travailles :

À COLLER

```
claude -p "/context" > context.txt
```

Chez nous, dans le dossier de travail de l'agence : 39,8 k tokens. 20 k pour les trois fichiers de consignes (le CLAUDE.md global, celui du projet et la mémoire automatique), 9,9 k pour la liste des skills, 4,5 k pour les agents, 3,1 k de messages déjà présents au démarrage, 2,2 k de prompt système. Le même Claude Code sans aucune personnalisation, dans un dossier vide, en prend 11,2 k. Trois fois et demie moins.

Pour faire la même comparaison chez toi, ouvre un dossier vide et lance Claude Code en mode sans échec. Il démarre sans CLAUDE.md, sans skills, sans plugins, sans hooks, sans serveurs MCP et sans mémoire automatique ; ton compte et ton modèle restent les mêmes.

À COLLER

```
claude --safe-mode
```

Une grosse ligne « MCP tools (deferred) » n'a rien d'inquiétant. Chez nous elle affiche 45 k, mais ces outils sont différés : tant que Claude ne s'en sert pas, Claude Code ne charge que leurs noms et les instructions des serveurs, et ces 45 k ne comptent pas dans le total.

/skill-doctor : ce que chaque skill coûte, et combien de fois elle a servi

Il faut Claude Code 2.1.252 ou plus récent. Vérifie ta version, et mets à jour si besoin :

À COLLER

```
claude --version
```

À COLLER

```
claude update
```

Dans une session, /skill-doctor ouvre son rapport dans l'onglet Stats de /plugin. Depuis le terminal, il l'imprime en texte, plus pratique pour le prompt 1 :

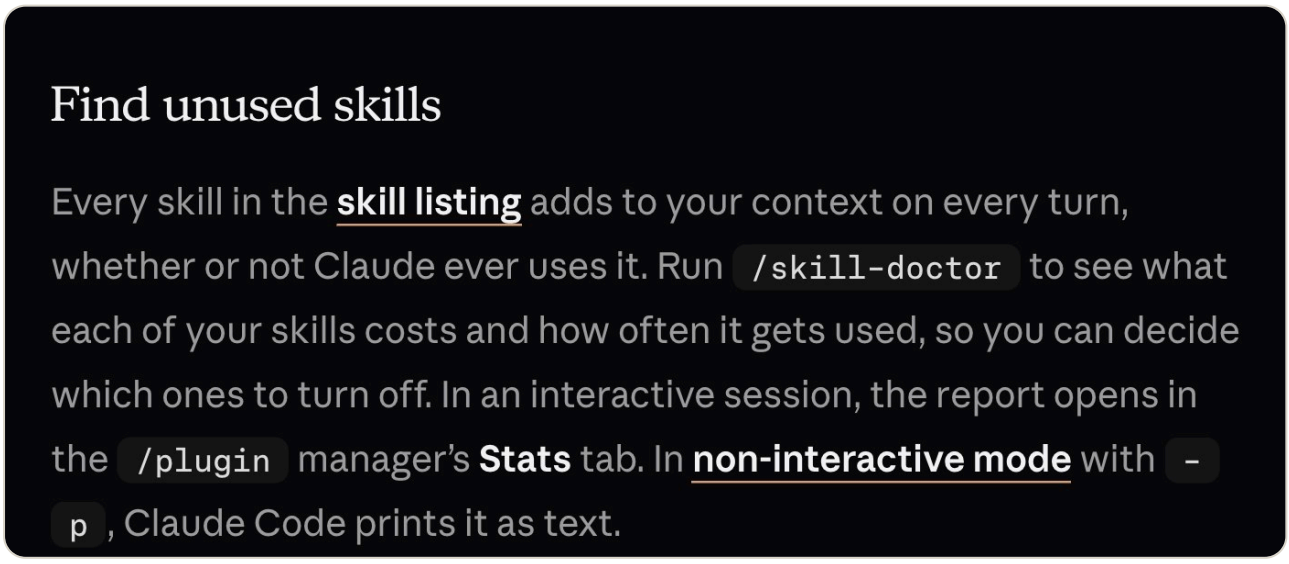
À COLLER

```
claude -p "/skill-doctor" > skill-doctor.txt
```

Une ligne par skill, quatre colonnes. `context`, ce que sa ligne pèse dans la liste à chaque message (« < 20 » veut dire qu'il ne reste que le nom). `7d tokens`, ce qu'elle a consommé ces 7 derniers jours. `uses`, le nombre d'appels. `last used`, le dernier. En bas, le rapport compte les skills jamais appelées selon leur origine (les tiennes, celles synchronisées depuis claude.ai, celles des plugins), dit où couper chacune et liste les plugins qui n'ont pas servi depuis longtemps. Les skills livrées avec Claude Code n'y figurent pas.

Notre rapport du 13 novembre :

- 260 skills chargées : 196 viennent de 23 plugins, 24 de la synchro avec claude.ai, 40 sont à nous.
- 216 jamais appelées. Le résumé du bas en signale 189 : il laisse de côté les skills inutilisées des plugins dont une autre skill a servi, puisqu'un plugin se coupe en entier.
- 27 appelées dans les 30 derniers jours.
- 224 réduites à leur nom dans la liste envoyée à Claude.
- Les plus appelées : humanizer (45 fois), agent-reach (20), brainstorming (17), find-skills (12), systematic-debugging (9), pdf (8), apple-design et chrome-browser (7 chacune).



La section « Find unused skills » de la page Skills de la doc Claude Code, capturée le 13 novembre 2026 : chaque skill de la liste « adds to your context on every turn », /skill-doctor dit ce qu'elle coûte et combien de fois elle sert.



La section « Skill descriptions are cut short », même page, capturée le 13 novembre 2026 : la liste garde tous les noms, mais au-delà de 1 % de la fenêtre de contexte, Claude Code retire des descriptions, en commençant par les skills les moins appelées.

Anthropic propose aussi un bilan plus large, /doctor (alias /checkup) : il compare tes skills, serveurs MCP et plugins inutilisés à ce qu'ils coûtent en contexte, puis propose des corrections qu'il n'applique qu'après ton accord. Et /doctor prompt-audit, à partir de la version 2.1.283, relit tes CLAUDE.md, tes skills et tes agents pour trouver les consignes écrites pour d'anciens modèles ou qui se contredisent. On ne les a pas lancés pour ce guide.

---

1. Prompt 1, l'audit : garder, fusionner, couper

Quand le lancer : avant d'installer quoi que ce soit d'autre, puis une fois par mois.

Ce qu'il faut lui donner, dans le même message :

- la sortie de /skill-doctor et celle de /context (partie 0) ;
- la liste de tes plugins avec leur état, telle que l'imprime `claude plugin list` ;
- tes serveurs MCP, avec `claude mcp list` ;
- les noms des dossiers de ~/.claude/skills, ~/.claude/commands, ~/.claude/agents, et du dossier .claude/skills de ton projet s'il existe.

Le prompt, à coller tel quel :

À COLLER

```
Fais l'audit de mon setup Claude Code avant que j'installe quoi que ce soit
d'autre.

Je te colle la sortie de /skill-doctor, celle de /context et la liste de mes
dossiers.

1. Range chaque skill, plugin, serveur MCP et chaque fichier de consignes
(CLAUDE.md, mémoire) dans une de ces trois colonnes :
   GARDER : servi ce mois-ci, et tu ne saurais pas le faire seul.
   FUSIONNER : fait doublon avec autre chose.
   COUPER : jamais appelé, ou corrige un défaut que les modèles actuels n'ont
plus.
2. Chiffre ce qui part, en tokens chargés à chaque message.
3. Propose une arborescence propre : un dossier par sujet, des noms qui disent
ce qu'il y a dedans, pour que tes agents sachent où tout se trouve.
4. Ne supprime rien. Donne-moi la liste et les commandes exactes, je valide
ligne par ligne.
```

Ce que Claude nous a rendu

Lancé avec Claude Opus 5.5, sans accès à nos fichiers, sur nos deux rapports et la liste de nos dossiers. Réponse en 21 minutes.

- Garder : humanizer (45 appels), agent-reach (20), apple-design (7), les skills de nos formats vidéo, superpowers (« tu le gardes parce que tu t'en sers »), deux skills Higgsfield servies le jour même.
- Fusionner : une commande et deux skills d'écriture dans humanizer ; creator-breakdown dans reel-autopsy, « même geste : disséquer une vidéo » ; huit skills Higgsfield jamais appelées rangées comme références d'une seule ; les 17 skills d'un plugin SEO, appelé une fois, remplacées par une seule skill à nous.
- Couper : 13 plugins jamais appelés (sales et ses 36 skills, figma 16, engineering 10, data 10, legal 9, finance 8…), 14 skills synchronisées depuis claude.ai jamais appelées, et nos propres skills restées à zéro.
- Les fichiers de consignes restent, mais maigrissent. Partent d'office les consignes pour des défauts disparus (« réfléchis étape par étape », « n'invente pas »), les MAJUSCULES répétées, les descriptions de skills déjà présentes dans la liste, l'historique et les doublons entre fichiers. Reste ce que Claude ne peut pas deviner : chemins, comptes, conventions, préférences.
- Le chiffrage : la liste des skills passe de 9,9 k à 5,3 k tokens (calculé), les agents perdent environ 2 k (estimé), les consignes environ 15 k (un objectif, Claude n'avait pas lu les fichiers). Total visé : environ 18 k au lieu de 39,8 k. Ce sont ses calculs ; rien n'est encore appliqué chez nous.

Sa phrase la plus utile : « Le vrai gain n'est pas la place : 40k, c'est 4 % de 1M. C'est la précision. » Chez nous, 216 descriptions jamais appelées faisaient concurrence aux quelques dizaines qui servent.

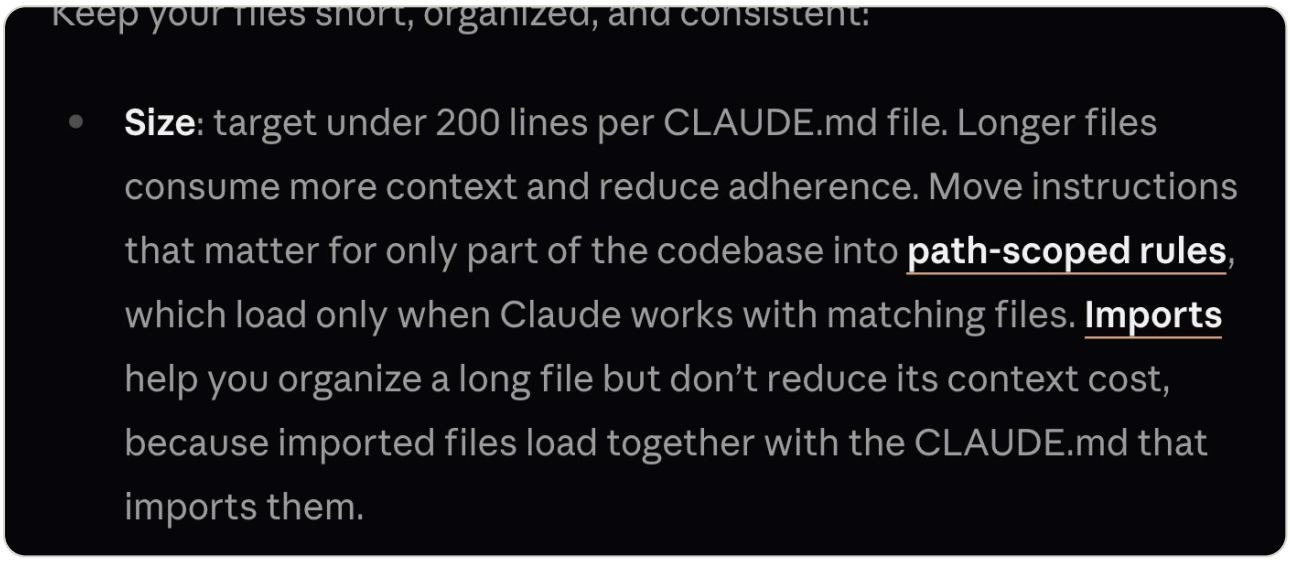
L'arborescence proposée :

- un CLAUDE.md global court (1,5 k tokens au plus) : qui tu es, ta langue, tes préférences générales ;

- les skills à plat dans ~/.claude/skills, avec un préfixe par sujet : texte-humanizer, video-reel-autopsy, design-title… ;
- les skills absorbées rangées dans un dossier references/ de celle qui les remplace, leur SKILL.md renommé en README.md pour qu'elles sortent de la liste ;
- un MEMORY.md qui ne sert plus que d'index (1 k au plus), et un fichier par sujet, lu à la demande ;
- une archive hors de ~/.claude, ~/claude-archive-2026-11-13/, où rien n'est chargé.

Pourquoi des préfixes plutôt que des sous-dossiers : la doc range chaque skill dans ~/.claude/skills/nom/SKILL.md, sur un seul niveau.

Enfin, les commandes, en blocs numérotés : un premier bloc en lecture seule pour vérifier ce qu'il ne pouvait pas savoir, puis les plugins à désactiver, les skills synchronisées à couper, l'archivage par déplacement, les fusions et les renommages.



La règle de taille de la page mémoire de la doc Claude Code, capturée le 13 novembre 2026 : moins de 200 lignes par CLAUDE.md, « Longer files consume more context and reduce adherence ».

Les pièges

- Garde le point 4. Le prompt ne supprime rien et rend des commandes : lis-les avant de les lancer, une par une.
- Archive plutôt que supprimer. Déplace les dossiers de skills hors de ~/.claude ; si l'une te manque, un déplacement la remet. Autre option sans toucher aux fichiers, pour tes skills à toi (pas celles des plugins) : dans /skills, la barre d'espace fait passer une skill de « on » à « name-only », « user-only » ou « off ».
- Les plugins ne se coupent pas skill par skill. On désactive le plugin entier, avec `claude plugin disable nom@marketplace`, et `claude plugin enable` le remet.
- Les skills synchronisées depuis claude.ai se coupent dans /skills ou dans les réglages des skills sur claude.ai. Une copie effacée à la main revient à la synchro suivante. Certaines skills d'Anthropic, comme pdf et xlsx, se synchronisent toujours.
- Donne-lui la sortie de `claude plugin list` plutôt qu'une liste de dossiers. On avait donné les dossiers : 16 plugins chargeaient des skills sans y figurer, Claude a dû marquer une partie de ses conseils « si actif ».
- Renommer une skill casse les fichiers qui la citent (CLAUDE.md, autres skills). Cherche ces mentions avant de renommer.
- Ne recopie pas l'arborescence dans CLAUDE.md. Anthropic range « décrire l'architecture et l'arborescence dans CLAUDE.md » parmi les conseils à abandonner : Claude lit le dossier

lui-même.

---

2. Prompt 2, les modèles : un fichier par tâche qui revient

Quand le lancer : une fois, puis dès qu'une nouvelle tâche revient deux fois dans le mois.

Ce qu'il faut lui donner : la liste de tes tâches du mois et le nombre de fois où chacune est revenue (compte-les dans tes dossiers, ton agenda ou tes factures). Et pour chaque tâche, un vrai rendu que tu as validé. Sans exemple réel, il en invente un.

> **À COLLER**
>
> ```
> Passe en revue toutes les tâches que je te confie.
>
> Je te colle la liste de mes tâches du mois et le nombre de fois où chacune est
> revenue.
>
> 1. Garde celles qui reviennent au moins deux fois par mois.
> 2. Pour chacune, écris un modèle dans templates/<tâche>.md : le but en une
> ligne, les étapes, le format de sortie exact, un exemple réussi, ce qu'il ne
> faut jamais faire. Trente lignes au plus.
> 3. Ajoute à CLAUDE.md une seule ligne par modèle, qui dit quand le lire.
> ```

Ce que Claude nous a rendu

On lui a donné dix tâches, comptées du 14 octobre au 13 novembre : de 63 fois (la légende d'une vidéo) à une seule (nettoyer le disque du Mac). Claude Sonnet 5 a répondu en 61 secondes. Il a écarté la tâche faite une fois et écrit neuf modèles : traduire-script.md, legende-textes-video.md, fiche-reponse-automatique.md, guide-pdf.md, verification-faits-prix.md, relecture-voix-off-whisper.md, post-linkedin-repris.md, carrousel-instagram.md, compte-rendu-journal.md.

Dans CLAUDE.md, une ligne par modèle, comme celle-ci : « Lire templates/traduire-script.md quand on traduit un script de vidéo étrangère réplique par réplique. »

Dans ce modèle de traduction, le format de sortie tient en deux lignes par réplique, « [00:00:12] VO : texte original » puis « [00:00:12] FR : traduction », et la liste de ce qu'il ne faut jamais faire commence par « Fusionner ou couper des répliques ».

Ce qui n'allait pas : on ne lui avait pas donné d'exemple réel, et il en a inventé. Le modèle de légende vouvoie, met un emoji et propose « Ce prix va vous surprendre ». Le modèle de post LinkedIn dit de changer le fond du post repris, l'inverse de notre façon de faire. Corrige ces lignes à la main, ou relance avec tes vrais exemples.

Pourquoi une ligne par modèle dans CLAUDE.md, plutôt qu'une skill de plus : c'est l'approche qui a gagné chez Vercel. Un index court posé dans AGENTS.md, le fichier de consignes que lisent beaucoup d'agents, a fait mieux qu'une skill appelée à la demande. Leur test portait sur la documentation de Next.js : l'idée vaut pour des modèles de tâches, leurs chiffres non.

Next.js to APIs, we found something unexpected.

A compressed 8KB docs index embedded directly in `AGENTS.md` achieved a 100% pass rate, while skills maxed out at 79% even with explicit instructions telling the agent to use them. Without those instructions, skills performed no better than having no documentation at all.

Here's what we tried, what we learned, and how you can set this up for your own Next.js projects.

L'article de Vercel du 27 janvier 2026, capturé le 13 novembre : un index de 8 Ko posé dans AGENTS.md atteint 100 % de réussite, la skill plafonne à 79 % même avec une consigne qui demande de l'utiliser.

Les pièges

- Il doit ajouter ses lignes à CLAUDE.md. Dans notre test, le dossier était vide et Claude a créé CLAUDE.md. Chez toi, vérifie qu'il ajoute ses lignes à la fin de ton fichier au lieu de le réécrire.
- Opus 5.5 a refusé notre premier essai au bout de 30 minutes (« Opus 5.5's safeguards flagged this message »), un faux positif sur une simple liste de tâches. Le même prompt, relancé sur Sonnet 5, est passé en une minute. Si ça t'arrive, change de modèle avec /model.
- Compte les lignes toi-même. Claude a conclu que chaque fichier tenait « en 30 lignes max » ; on a compté, lignes vides comprises : sept modèles sur neuf dépassent, le plus long fait 46 lignes.

---

3. Prompt 3, l'interview : tes préférences en quinze lignes

Quand le lancer : une fois, puis chaque fois que tu te surprends à corriger Claude deux fois sur la même chose.

Ce qu'il faut lui donner : rien. Il pose les questions, tu réponds franchement, avec des exemples de ce qui t'agace.

À COLLER

```
Interviewe-moi pour savoir comment je veux travailler avec toi.

Une question à la fois, avec des choix quand c'est possible. Couvre la langue et
le ton, la longueur des réponses, ce que tu fais sans me demander et ce que je
dois valider, mes outils, et ce qui m'agace dans tes réponses.

Arrête-toi quand tu peux écrire mes préférences en quinze lignes au plus. Écris-
les dans preferences.md, puis ajoute à CLAUDE.md une ligne qui dit de le lire.
```

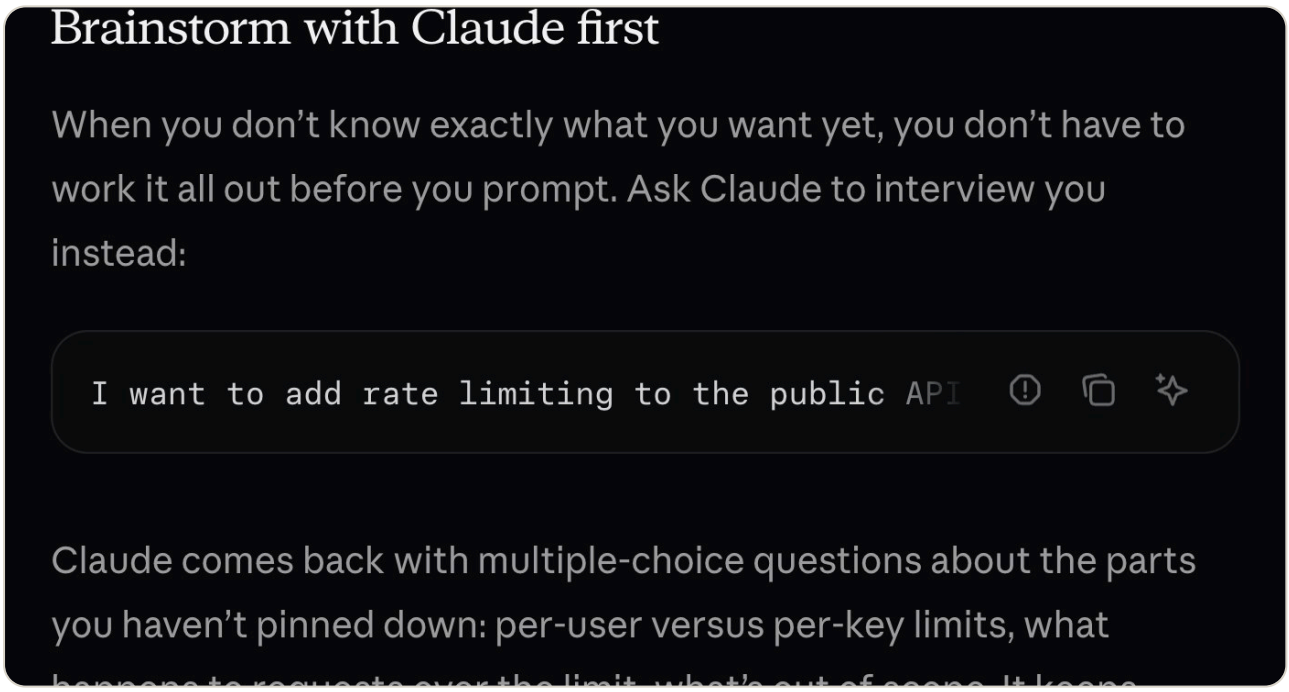
Ce que Claude nous a rendu

Avec Claude Opus 5.5 : six questions à choix, une par message, chacune en moins de 5 secondes. Dans l'ordre, la langue, le ton et le tutoiement, la longueur des réponses, ce qu'il peut faire sans demander, nos outils, ce qui nous agace. Il a ensuite écrit un preferences.md de 14 lignes, et dans CLAUDE.md : « Lis preferences.md avant de commencer et applique ces préférences. »

Six des quatorze lignes :

- Langue : français. Code, identifiants et commentaires de code en anglais.
- Ton : direct et sobre, tutoiement. Pas de formules du type « Great question ».
- Conclusion d'abord, explication ensuite.
- Jamais de commit ni de push sans demande explicite.
- Demander avant toute action irréversible ou externe : supprimer des données en production, publier, écrire à un client.
- Ne pas dire « c'est fait » sans avoir vérifié (tests, build).

L'écriture du fichier a pris 10 minutes 35 chez nous, parce qu'on avait coupé les outils pour le test : Claude essayait en boucle de lister un dossier qu'il ne pouvait pas lire. En session normale, il a ses outils et écrit le fichier directement.



La section « Brainstorm with Claude first » de la page Best practices, capturée le 13 novembre 2026 : Anthropic conseille de demander à Claude de t'interviewer, avec des questions à choix, pour écrire une spec.

Les pièges

- Une préférence écrite reste un conseil que Claude suit la plupart du temps. La page mémoire est claire : Claude traite CLAUDE.md « as context, not enforced configuration ». Ce qui ne doit jamais arriver (un push, une suppression) passe par une règle de permission ou un hook.
- Si ton projet est partagé avec une équipe, mets tes préférences dans CLAUDE.local.md, ajouté au .gitignore, comme le conseille Anthropic, ou dans ton ~/.claude/CLAUDE.md personnel.
- Quinze lignes au plus. Si tu en as trente, garde celles que Claude ne devinerait pas seul.

Ce qu'Anthropic conseille de laisser tomber

La page Best practices de la doc Claude Code a une table, « What you can stop doing ». Elle s'ouvre ainsi : « Every one of these was good advice when it was written. Most were workarounds for something the model couldn't reliably do yet, and the models have since caught up. » Chacun de ces conseils était bon quand il a été écrit ; la plupart contournaient ce que le modèle ne savait pas encore faire de façon fiable, et les modèles ont rattrapé.

Les sept lignes, traduites. D'abord le vieux conseil, puis pourquoi il ne sert plus.

1. « La fenêtre de contexte est ta contrainte principale, gère-la de près. » La plupart des modèles actuels ont une fenêtre d'un million de tokens et la compaction se fait toute seule : en manquer est rare. /clear entre deux tâches reste utile, parce qu'un contexte centré donne de meilleures réponses et coûte moins.

2. « Ajoute IMPORTANT ou des MAJUSCULES aux consignes que Claude rate. » Les anciens modèles en avaient parfois besoin. Les actuels suivent une ligne écrite simplement, et l'insistance les pousse surtout à trop appliquer la règle. Une règle qui ne doit pas plier devient un hook ou une règle d'interdiction. Si la ligne se perd dans un long fichier, c'est le fichier qu'il faut raccourcir.

3. « Utilise des sous-agents pour protéger ta fenêtre de contexte. » Manquer de contexte est rarement le problème aujourd'hui. Les sous-agents restent très bien pour travailler en parallèle, tenir à l'écart une sortie bavarde ou relire de façon indépendante, mais chacun coûte des tokens en plus.

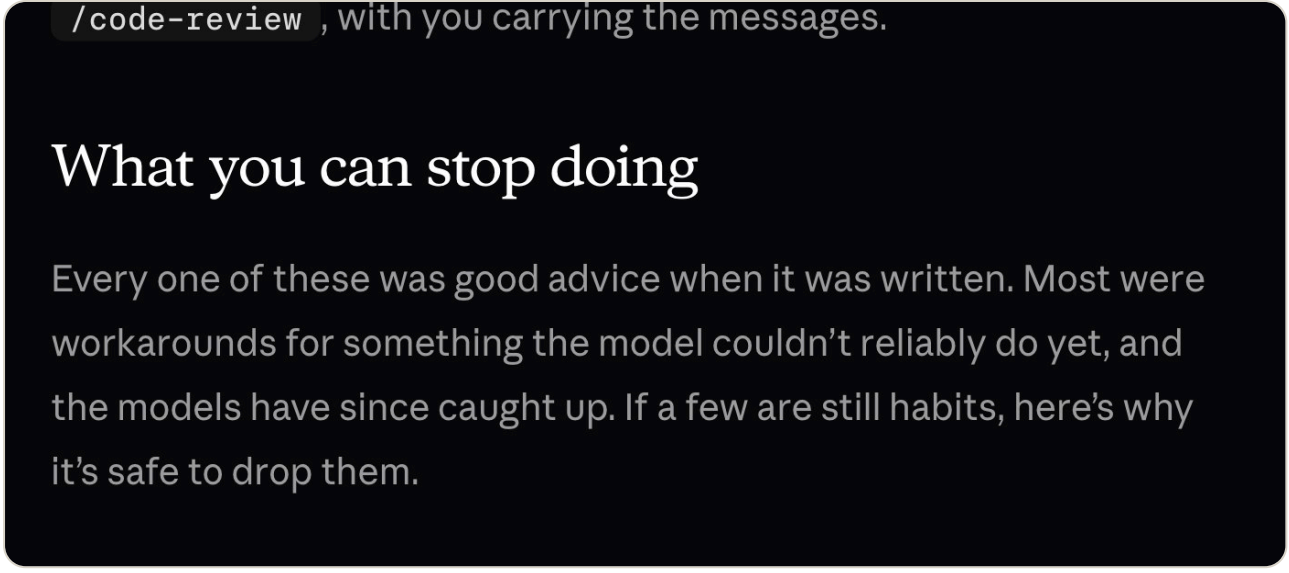
4. « Délègue à un sous-agent pour avoir une meilleure réponse. » Confier la même tâche à un sous-agent ne la rend pas meilleure. Un relecteur neuf aide, parce qu'il n'a pas vu ton raisonnement, et ça n'a rien à voir avec déléguer.

5. « think hard », « think harder », « be thorough » dans le prompt. Le réglage pour ça, c'est l'effort. Ces mots sont du texte ordinaire : ils poussent un peu Claude à l'intérieur de son réglage d'effort, sans le relever. « ultrathink » marche encore : il demande un raisonnement plus poussé sur ce seul message, sans toucher à ton réglage.

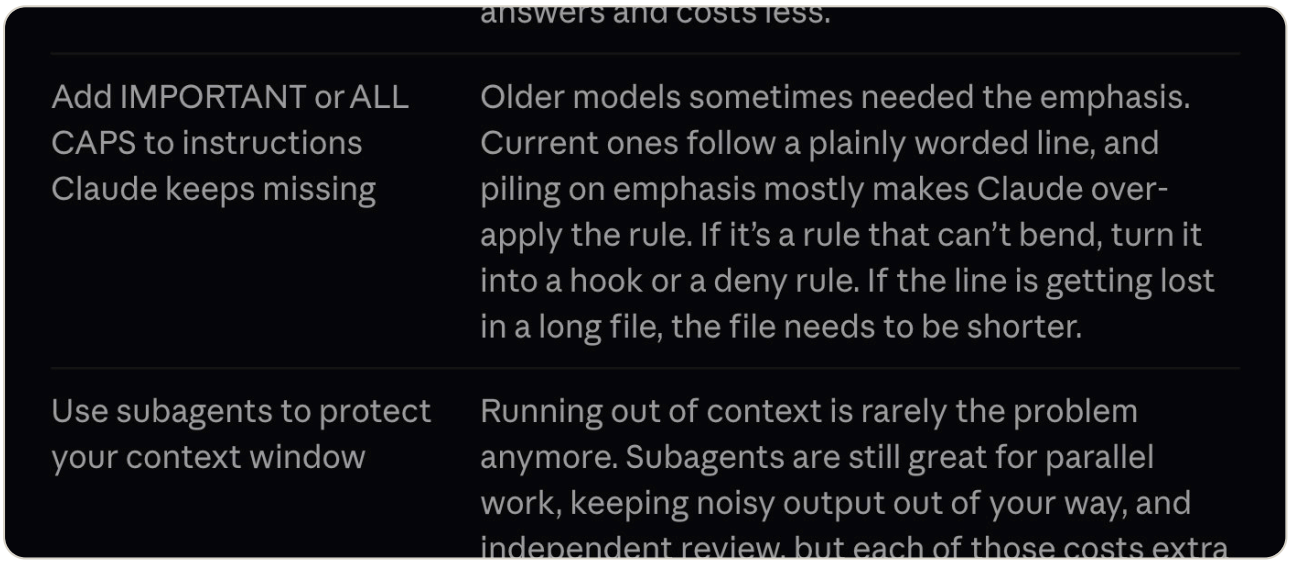
6. « Dis à Claude de continuer, de boucler jusqu'au bout, de ne pas être paresseux. » Les modèles actuels vont en général au bout d'une tâche convenue sans qu'on le leur dise, et ce genre de ligne cause plus souvent du travail en trop qu'il ne débloque un arrêt. Donner à Claude une vérification à passer reste utile. Si un travail s'arrête trop tôt, dis « continue » ; pour un travail sans surveillance, /goal ou un hook Stop s'en chargent.

7. « Décris ton architecture et l'arborescence de tes fichiers dans CLAUDE.md. » Claude lit le dépôt lui-même. N'écris que ce qui n'y est pas.

La page finit sur cette phrase : si un autre conseil commence à ressembler à une corvée, il a sans doute sa place dans la table.



Le début de la table « What you can stop doing », page Best practices de la doc Claude Code, capturée le 13 novembre 2026.



La ligne sur IMPORTANT et les majuscules, même page, même jour : « If the line is getting lost in a long file, the file needs to be shorter. »

Garder, fusionner, couper : la grille et nos chiffres

Pour décider seul, skill par skill ou plugin par plugin, avec ton rapport /skill-doctor sous les yeux. Pose les questions dans l'ordre et arrête-toi à la première réponse nette.

1. Elle a servi dans les 30 derniers jours (colonne last used) ? Garde-la. Chez nous, 27 sur 260.

2. Elle fait le même geste qu'une autre que tu gardes ? Fusionne : ce qu'elle a d'utile va dans un dossier references/ de l'autre, et elle sort de la liste.

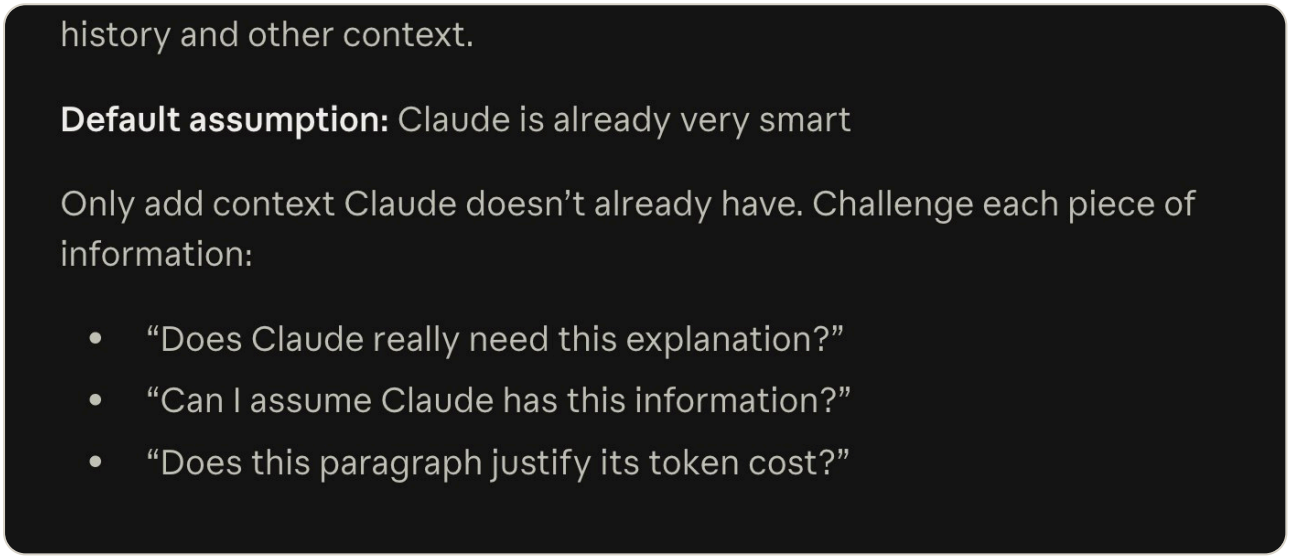
3. Elle vient d'un plugin dont une seule skill te sert ? Désactive le plugin et fais réécrire la seule utile en skill à toi. Chez nous, un plugin SEO de 17 skills pour un seul appel.

4. Elle corrige un défaut que le modèle n'a plus (insister en majuscules, « réfléchis étape par étape », « ne sois pas paresseux ») ? Coupe.

5. Jamais appelée, et tu ne sais plus pourquoi tu l'as installée ? Archive-la, ou désactive son plugin. Chez nous, 216.

6. Rare mais précieuse le jour où elle sert, comme une mise en production ? Garde-la. Si c'est une skill à toi et que tu préfères la lancer avec /nom, passe-la en « user-only » dans /skills : elle sort alors de la liste envoyée à Claude.

Pour les fichiers de consignes, la même logique, ligne par ligne. Anthropic propose deux questions avant d'écrire quoi que ce soit dans CLAUDE.md : Claude pourrait-il le trouver en lisant le code ? Irait-il vraiment voir ? Si les deux réponses sont oui, la ligne est en trop. Garde les commandes de test, les pièges du projet, les décisions qui ne sont écrites nulle part, et ce que tu as déjà dû corriger deux fois.



La page des bonnes pratiques des Agent Skills, documentation de la plateforme Claude, capturée le 13 novembre 2026 : « Default assumption: Claude is already very smart », et trois questions à poser à chaque paragraphe, dont « Does this paragraph justify its token cost? »

Nos chiffres, relevés le 13 novembre 2026 :

- 260 skills chargées, 27 servies dans le mois, 216 jamais appelées, 224 réduites à leur nom.
- 455 lignes de consignes en trois fichiers (150, 147 et 158 lignes), 20 k tokens à chaque message. Chaque fichier est sous la limite de 200 lignes de la doc : ce sont leurs lignes longues qui pèsent.
- 39,8 k tokens avant le premier message, contre 11,2 k pour Claude Code sans personnalisation.
- Environ 18 k après le tri, selon le chiffrage de Claude. On relancera /context une fois le tri appliqué pour avoir le vrai chiffre.

Garder un setup propre dans la durée

- Relance /skill-doctor une fois par mois et compare avec le mois d'avant.
- Une nouvelle installation remplace un élément de ta colonne garder. Sinon, elle n'entre pas. C'est la règle que Claude nous a laissée à la fin de l'audit.
- Quand Claude se trompe deux fois sur la même chose, ajoute une ligne à tes préférences ou à ton CLAUDE.md plutôt qu'une skill de plus. L'équipe de Claude Code fait pareil : «

Anytime we see Claude do something incorrectly we add it to the CLAUDE.md » (Boris Cherny, 2 janvier 2026).

- À chaque nouveau modèle, ouvre une session sans tes ajouts (`claude --safe-mode`) et regarde ce qui manque vraiment. C'est la version prudente du conseil de Boris Cherny, qui dit de tout supprimer puis de remettre ligne par ligne. « Only when you see it repeatedly stumble on the same thing, that's when you add it back » (Y Combinator, 27 juillet 2026).
- Après chaque grand ménage, relance /context pour lire le vrai chiffre plutôt qu'une estimation.

Sources, relues le 13 novembre 2026

- Best practices for Claude Code, doc Claude Code : « What you can stop doing », « Keep CLAUDE.md short », « Put team procedures in skills », « Brainstorm with Claude first ».
- Extend Claude with skills, doc Claude Code : « Find unused skills » (/skill-doctor, version 2.1.252 ou plus récente), « Skill descriptions are cut short », skillOverrides, skills synchronisées depuis claude.ai, /doctor.
- How Claude remembers your project, doc Claude Code : la limite de 200 lignes, /doctor prompt-audit (version 2.1.283 ou plus récente).
- Discover plugins et CLI reference, doc Claude Code : `claude plugin disable`, `--safe-mode`.
- Manage costs effectively, doc Claude Code : cache des requêtes, outils MCP différés.
- Agent Skills, best practices, documentation de la plateforme Claude.
- AGENTS.md outperforms skills in our agent evals, Vercel, Jude Gao, 27 janvier 2026.
- Le post de Boris Cherny sur son setup, X, 2 janvier 2026.
- Boris Cherny: We Cut 80% of Claude Code's Prompt, Y Combinator, Startup School 2026, 27 juillet 2026, et sa transcription publiée par YC.