Eigenflow

GUIDE · CLAUDE CODE

# Arrête de brûler ta limite Claude

La skill qui trie chaque tâche entre Opus et Haiku, ses trois fichiers à coller et notre test chiffré.


PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb



ÉDITION

novembre 2026

Dans Claude Code, Opus 5.5 est le modèle par défaut des offres Pro et Max. Il fait donc tout, même renommer un fichier ou relire 5 000 lignes de logs, et chacune de ces tâches compte dans ta limite.

Ce guide te donne les trois fichiers qui confient ces corvées à Haiku 5.5. Il te donne aussi le résultat de notre test du 6 novembre 2026, avec ses bons et ses mauvais côtés.

---

## À savoir avant de lire

- Haiku 5.5 coûte 90 % de moins que Haiku 4.5, et 40 fois moins qu'Opus 5.5 par token, tant qu'une requête reste sous 100 000 tokens.

- La skill de ce guide a fait baisser la note de 23 % sur une journée pleine de corvées. Sur un bug qui demande surtout de réfléchir, elle l'a fait monter de 29 %.

- Dans notre test, le plus gros gain vient de Sonnet tout seul : 56 à 59 % de moins, sans une erreur.

- Les chiffres sont des prix d'API. Anthropic ne publie pas comment chaque token compte dans la limite d'un abonnement.

---

## 1. Ce que coûtent les modèles

Prix officiels de l'API pour un million de tokens, relus le 6 novembre 2026 sur la page de prix d'Anthropic (https://platform.claude.com/docs/en/about-claude/pricing) :

- **Claude Opus 5.5** : 4 $ en entrée, 20 $ en sortie.

- **Claude Sonnet 5.5** : 2 $ en entrée, 10 $ en sortie.

- **Claude Haiku 5.5**, requêtes jusqu'à 100 000 tokens : 0,10 $ en entrée, 0,50 $ en sortie.

- **Claude Haiku 5.5**, au-delà de 100 000 tokens : 0,50 $ en entrée, 2,50 $ en sortie.

- **Claude Haiku 4.5** : 1 $ en entrée, 5 $ en sortie.

Haiku 5.5 coûte donc 90 % de moins que Haiku 4.5, et 40 fois moins qu'Opus 5.5 sous 100 000 tokens. Au-delà, l'écart avec Opus tombe à 8 fois.

La page d'aide d'Anthropic sur les limites de Claude Code (https://support.claude.com/en/articles/14552983-models-usage-and-limits-in-claude-code) le dit

sans détour : *Spending Opus on routine work is the fastest way to drain a daily limit.* Dépenser Opus sur le travail courant est la façon la plus rapide de vider sa limite du jour.

---

## 2. La skill : trois fichiers à coller

### Le sous-agent des corvées

Crée le fichier `.claude/agents/petites-mains.md` à la racine de ton projet, avec ce contenu :

**À COLLER**

```
---
name: petites-mains
description: Tâches mécaniques : renommer un fichier et mettre à jour ses
imports, modifier une ligne précise, ajouter une ligne à un fichier, lancer les
tests et résumer le résultat.
tools: Read, Glob, Grep, Bash, Edit, Write
model: haiku
effort: low
---
Tu fais la tâche mécanique qu'on te confie, exactement, sans rien ajouter.
Exemples : renommer un fichier et mettre à jour ses imports, modifier une ligne
précise, ajouter une ligne à un fichier, lancer les tests. Rends un compte rendu
court : ce que tu as fait, les fichiers touchés, et pour les tests le nombre de
réussites et d'échecs avec le message d'erreur exact.
```

### L'éclaireur sur Haiku

Claude Code a un éclaireur intégré, Explore, qui cherche dans tes fichiers. Il tourne sur le modèle de la conversation, donc sur Opus. La doc des sous-agents (https://code.claude.com/docs/en/sub-agents) donne la parade : un sous-agent de projet qui s'appelle Explore remplace l'intégré et garde son propre modèle.

Crée `.claude/agents/Explore.md` :

**À COLLER**

```
---
name: Explore
description: Éclaireur en lecture seule : chercher dans les fichiers du projet,
lire et filtrer des logs, lister des dossiers.
tools: Read, Glob, Grep, Bash
model: haiku
effort: low
---
Tu es l'éclaireur du projet. Tu cherches dans les fichiers et les logs, en
lecture seule, et tu ne modifies rien. Commence par grep et glob plutôt que de
tout lire. Rends un compte rendu court et précis : les fichiers et numéros de
ligne utiles, les extraits exacts (quelques lignes), les chiffres (combien
d'occurrences, quelles dates). Ne donne pas d'avis sur la cause si on ne te le
demande pas.
```

### La note de tri

Ajoute ces lignes au fichier `CLAUDE.md` de ton projet :

**À COLLER**

```
## Tri des tâches
Avant chaque étape, demande-toi si elle demande de la réflexion.
- Tâche mécanique (chercher dans les fichiers, lire ou filtrer des logs, lister un dossier, renommer un fichier et ses imports, modifier une ligne, lancer les tests) : confie-la à un sous-agent sur Haiku, Explore pour chercher et lire, petites-mains pour agir.
- Réflexion (comprendre la cause d'un bug, choisir et écrire le correctif, décider de l'architecture) : fais-la toi-même.
Ne relis pas toi-même ce qu'un sous-agent a déjà lu : demande-lui les lignes exactes dont tu as besoin.
```

Relance Claude Code, puis tape la commande suivante : petites-mains et Explore doivent y apparaître, avec le modèle Haiku.

> /agents

---

## 3. Notre test

### Le projet

Une petite appli de facturation, écrite pour le test, en Python standard :

- 15 modules et 14 tests ;

- 6 823 lignes de logs de production sur cinq jours ;

- un vrai bug d'arrondi. Un retour en arrière a remis l'ancien arrondi sur des nombres à virgule, et 238 factures sont parties avec 1 à 3 centimes d'écart.

### Ce qu'on a mesuré

On a lancé Claude Code (version 2.1.312) sur notre abonnement, en mode restreint et sans réglages personnels. Chaque réglage a tourné sur deux consignes.

- **Le bug** : trouver l'écart dans les logs, le corriger, lancer les tests, renommer un fichier, ajouter une ligne au journal des versions.

- **La journée** : le même bug, puis une série de corvées : compter les erreurs jour par jour dans les logs, passer la version dans deux fichiers, mettre à jour le journal, lancer les tests.

Les coûts sont ceux que Claude Code calcule au prix de l'API pour chaque modèle. Le correctif a été vérifié sur 40 factures que l'agent ne voyait pas.

### Les résultats

Moyenne de deux essais, sauf pour Sonnet et Haiku sur la journée (un essai) :

- **Opus seul (le réglage par défaut)** : le bug, 0,26 $ en 1 min, 39 cas justes sur 40 en moyenne ; la journée, 0,48 $ en 1 min 50.

- **Opus avec la skill** : le bug, **0,34 $ (+29 %)** en 3 min, 40 sur 40 ; la journée, **0,37 $ (–23 %)** en 2 min 40.

- **Sonnet seul** : le bug, 0,12 $ (–56 %) en 1 min, 40 sur 40 ; la journée, 0,20 $ (–59 %) en 2 min.

- **Haiku seul** : le bug, 0,02 $ (–93 %) en 1 min 30, 38 sur 40 ; la journée, 0,33 $ (–31 %) en 3 min, avec un bilan des logs faux.

### Comment lire ces chiffres

- **La skill rapporte sur une journée pleine de corvées.** Opus lit 3,3 fois moins de texte et la note baisse de 23 % (de 13 à 33 % selon l'essai). C'est environ 30 % de travail en plus pour le même prix.

- **Elle coûte plus cher sur un bug à réfléchir.** Opus lit moins, mais il écrit plus : des consignes pour Haiku, puis il relit ses rapports. Or pour Opus, écrire coûte 5 fois plus que lire (20 $ contre 4 $ le million). Quand il n'y a presque rien à déléguer, ce surcoût l'emporte, et la session est 3 fois plus lente.

- **Le levier le plus fort, dans notre test, c'est Sonnet tout seul.** Il coûte 56 à 59 % de moins, sans une erreur. C'est aussi le conseil de la page d'aide d'Anthropic : *consider switching to Sonnet for routine work*.

- **Haiku tout seul est imbattable sur une petite tâche.** Sur la journée, il a lu 2,2 millions de tokens et s'est trompé dans les comptes.

---

## 4. Notre réglage conseillé

1. Le travail courant se fait avec Sonnet.

2. Les problèmes vraiment durs (bug coriace, refonte, architecture) passent à Opus, ou à opusplan, qui planifie avec Opus et exécute avec Sonnet.

3. Les journées de corvées (renommages en série, logs à éplucher, tests à relancer) gardent Opus avec la skill de ce guide.

4. La commande usage montre la part de ta limite qui part dans les sous-agents.

Les commandes, à taper dans Claude Code :

> /model sonnet

> /model opusplan

> /usage

## 5. Limites du test

- Ce sont des prix d'API. Anthropic ne publie pas comment chaque token compte dans la limite d'un abonnement ; sa page d'aide dit seulement qu'Opus en consomme nettement plus.

- Deux essais par réglage sur un seul projet ; un essai à l'autre peut varier de 15 %.

- Une autre note de tri, ou d'autres sous-agents, donneraient d'autres chiffres.

---

## Sources


- Prix de l'API Claude (https://platform.claude.com/docs/en/about-claude/pricing), relus le 6 novembre 2026.

- Models, usage, and limits in Claude Code (https://support.claude.com/en/articles/14552983-models-usage-and-limits-in-claude-code), page d'aide d'Anthropic.

- Les sous-agents de Claude Code (https://code.claude.com/docs/en/sub-agents).

- Réduire les coûts de Claude Code (https://code.claude.com/docs/en/costs).

- Configurer les modèles dans Claude Code (https://code.claude.com/docs/en/model-config).