Eigenflow

GUIDE DUO · CLAUDE CODE

L'advisor de Claude Code avec Fable

Le réglage en trois commandes, ce que coûte chaque duo, et notre test du 15 novembre 2026 : Fable en conseiller a coûté plus que Fable seul.

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté DUO, voici le guide. L'advisor de Claude Code fait travailler deux modèles ensemble : un exécutant, Sonnet 5.5 par exemple, fait la tâche, et un conseiller plus fort, Fable 5.2 ou Opus 5.5, intervient aux moments clés. Ici : ce que c'est, le réglage en trois commandes, les paires possibles, ce que ça coûte selon ton offre, et notre test du 15 novembre 2026, cinq essais sur la même tâche.

Documentation de Claude Code et de l'API, billet d'Anthropic et page des prix relus le 15 novembre 2026. Test lancé le même jour avec Claude Code 2.1.325. On n'a aucun lien avec Anthropic.

À savoir avant de lire

- L'advisor est expérimental : la doc prévient que son fonctionnement, ses prix et sa disponibilité peuvent changer.
- Claude décide seul quand il consulte le conseiller. Aucun réglage ne force ni ne plafonne les appels.
- Sur notre tâche, Sonnet avec Fable en conseiller a coûté 1,59 $ puis 1,55 $. Fable seul : 1,27 $. Même résultat partout, 19 cas cachés réussis sur 19.
- Le duo que recommande le menu de Claude Code, Sonnet avec Opus en conseiller, a coûté 0,85 $: un tiers de moins que Fable seul. Sonnet seul a fait 19 sur 19 pour 0,28 $.
- À chaque appel, le conseiller relit toute la conversation au plein prix de son modèle, sans cache.
- Sur l'offre Pro, Fable se paie en crédits d'usage, en conseiller comme en modèle principal. Sur Max et Ultra, il est inclus jusqu'à la moitié de ta limite hebdomadaire.
- Une seule tâche, courte, un ou deux essais par réglage : un indice, à refaire sur ton propre travail.

1. Ce qu'est l'advisor

Deux modèles dans la même session. L'exécutant fait tout le travail : il lit les fichiers, écrit le code, lance les tests. Quand il le juge utile, il appelle le conseiller. Le conseiller reçoit toute la conversation, chaque appel d'outil et chaque résultat compris, et renvoie un conseil : un plan, une correction ou un signal d'arrêt, d'après le billet d'Anthropic du 9 avril 2026. Il n'a aucun outil, n'écrit rien dans ton projet et tourne sur les serveurs d'Anthropic.

Quand Claude l'appelle

D'après la doc de Claude Code, Claude consulte plutôt avant de choisir une approche, quand une erreur revient, et avant de déclarer la tâche finie, « but the timing is model-driven rather than rule-

based ». Pour plus ou moins d'appels, écris-le dans ton message, par exemple « consulte le conseiller avant de continuer ».

Dans nos trois essais avec conseiller, Sonnet a fait deux appels à chaque fois. Le premier après avoir lu la spec et les tests, avant d'écrire une ligne de code. Le second juste avant de conclure, pour une relecture. Ce n'est pas garanti : dans une mesure d'Anthropic faite par l'API, Sonnet 5.5 à l'effort high n'a consulté son conseiller sur aucun des 20 problèmes de code testés.

Ce que tu vois

Pendant l'appel, une ligne Advising avec le nom du conseiller, puis un compte rendu : conseil reçu, refus ou panne. Ctrl+O affiche le conseil quand il est lisible. Fable 5.2 et Opus 5.5 le renvoient chiffré : dans nos journaux, c'est un bloc `advisor_redacted_result` illisible, on n'a donc pas pu lire ce que Fable a conseillé. Ce que montre Ctrl+O avec ces deux modèles : à tester.

---

2. Le régler en trois commandes

Le conseiller : /advisor

Dans Claude Code, au terminal :

À COLLER

```
/advisor
```

Le menu propose quatre choix : Fable 5.2, Opus 5.5, Sonnet 5.5 et No advisor. Sous la liste, Claude Code donne son propre conseil : « Recommended setup: Sonnet as the main model with Opus as the advisor. For certain workloads this gives near-Opus performance with reduced token usage. » On a relevé ce menu le 15 novembre, en version 2.1.325, puis on l'a fermé par Échap sans rien choisir.

Ou en une ligne, sans menu :

À COLLER

```
/advisor fable
```

D'après la doc, Claude Code confirme par « Advisor set to » suivi du modèle, et garde ce choix pour les sessions suivantes. Noms acceptés : `fable`, `opus`, `sonnet`, ou un nom complet comme `claude-opus-5-5`.

Dans l'appli de bureau, avec `claude -p` et dans l'Agent SDK, il n'y a pas de menu : tape directement `/advisor fable`. Seul, `/advisor` affiche le réglage en cours. Chez nous, `claude -p "/advisor"` a répondu « Advisor: off » puis « Usage: /advisor <fable|opus|sonnet|off> ».

Le modèle principal : /model

```
À COLLER
/model
```

Notre menu, sur une offre Ultra : Default (recommended) Fable 5.2, puis Opus 5.5, Fable 5.2, Sonnet 5.5 et Haiku 5.5. Si tu es sur Default, ton modèle principal est déjà Fable, et Fable 5.2 n'accepte que Fable 5.2 comme conseiller. Pour le duo, passe sur Sonnet :

```
À COLLER
/model sonnet
```

Dans le menu, Entrée garde ton choix pour les sessions suivantes, la touche s le limite à la session en cours.

Pour toutes tes sessions, ou pour une seule

Le réglage `advisorModel`, dans ton fichier `~/.claude/settings.json` :

```
À COLLER
{
  "advisorModel": "opus"
}
```

Ou au lancement, pour cette session seulement, sans toucher au réglage enregistré :

```
À COLLER
claude --model sonnet --advisor opus
```

L'option `--advisor` n'apparaît pas dans `claude --help`. Claude Code refuse de démarrer si ton modèle principal ne prend pas de conseiller, si le modèle demandé ne peut pas conseiller (Haiku 4.5), ou si tu demandes Fable sans ton accord pour les crédits d'usage (partie 8). Un conseiller moins fort que le modèle principal laisse démarrer la session, sans conseiller.

Le couper

```
À COLLER
/advisor off
```

Ou No advisor dans le menu. La variable `CLAUDE_CODE_DISABLE_ADVISOR_TOOL=1` coupe l'outil complètement.

3. Les paires possibles

La règle de Claude Code : le conseiller doit être classé au moins au niveau du modèle principal. Le tableau de la doc, du modèle principal le moins fort au plus fort :

- Haiku 4.5 : Fable, Opus, Sonnet, Haiku 5.5.
- Sonnet 4.6 : Fable, Opus, Sonnet, Haiku 5.5.
- Opus 4.6 : Fable, Opus, Sonnet 5 ou plus récent, Haiku 5.5.
- Sonnet 5 ou Haiku 5.5 : Fable, Opus 4.7 ou plus récent, Sonnet 5 ou plus récent, Haiku 5.5.
- Opus 4.7 ou 4.8 : Fable, Opus 4.7 ou plus récent, Sonnet 5.5.
- Sonnet 5.5 : Fable, Opus 5 ou plus récent, Sonnet 5.5.
- Opus 5 ou 5.5 : Fable, Opus 5 ou plus récent.
- Fable 5 : Fable 5.2, 5.1 ou 5.
- Fable 5.1 : Fable 5.2 ou 5.1.
- Fable 5.2 : Fable 5.2 seulement.

Haiku 4.5 peut appeler un conseiller mais ne peut pas en être un.

Les duos que cite la doc :

- Sonnet et Opus : Sonnet fait la routine, Opus prend le plan, les échecs ambigus et la vérification finale.
- Sonnet et Fable : l'avis de Fable aux moments de décision, sans faire tourner Fable tout le temps.
- Haiku et Opus : plus cher que Haiku seul, moins cher que de passer à Sonnet ou Opus.
- Opus et Opus : un second Opus relit le premier, quand le contrôle compte plus que le coût.
- Fable et Fable : le duo le plus fort.
- Sonnet et Sonnet : un second avis peu cher, pour les oublis de routine.

---

4. Ce que ça coûte, et qui paie

À l'API

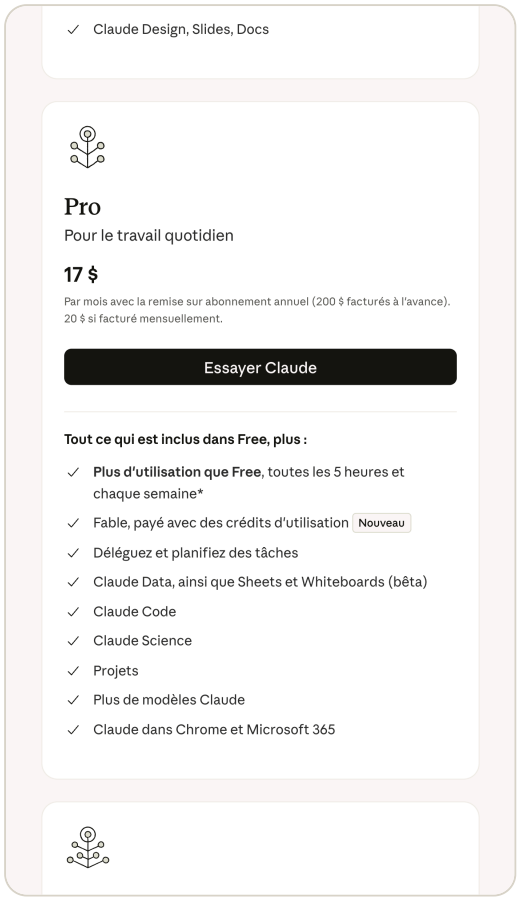
Prix par million de tokens, entrée puis sortie, relevés le 15 novembre : Fable 5.2 à 8 $ et 40 $, Opus 5.5 à 4 $ et 20 $, Sonnet 5.5 à 2 $ et 10 $. Relire depuis le cache coûte bien moins : 0,20 $ le million pour Fable 5.2 et Opus 5.5, 0,10 $ pour Sonnet 5.5.

Les tokens du conseiller se paient au tarif du conseiller, en plus de ceux de l'exécutant. Et dans Claude Code, sa lecture de la conversation n'est jamais mise en cache : chaque appel relit tout. Un token relu par Fable en conseiller coûte 8 $ le million. Le même token relu par Fable dans sa propre session, depuis le cache, coûte 0,20 $: quarante fois moins.

Sur un abonnement

La page des prix en français affiche des dollars, hors taxes : Pro à 17 $ par mois en annuel (200 $ facturés d'avance) ou 20 $ au mois, Max à 100 $ par mois, Ultra à 200 $ par mois.

- Pro : un conseiller Opus ou Sonnet compte dans les limites de ton offre. Un conseiller Fable se paie en crédits d'usage, au tarif de l'API, après un accord donné une fois (partie 8).
- Max et Ultra : Fable est inclus jusqu'à la moitié de ta limite hebdomadaire. Le centre d'aide d'Anthropic parle d'une limite hebdomadaire à part pour Fable. Au-delà, ce sont des crédits d'usage, si tu les as activés.
- Les crédits d'usage s'activent dans Settings > Usage, avec un plafond mensuel. Ils sont facturés à part de l'abonnement. Les crédits API mensuels de Max et d'Ultra (100 $ et 200 $) ne servent pas à les payer.



La carte Pro de la page des prix de Claude en français, le 15 novembre 2026 : 17 $ par mois en annuel, 20 $ au mois, et la ligne « Fable, payé avec des crédits d'utilisation »

Ce que valent nos chiffres

Notre test a tourné sur une offre Ultra. Les montants de la partie 5 sont ceux que Claude Code calcule au prix de liste de l'API (champ `costUSD`) : ce qu'aurait payé un compte API, ou ce qu'auraient pris des crédits d'usage. Pas une facture.

---

5. Notre test du 15 novembre 2026

La tâche

Une fonction Python à écrire, `creneaux_libres` : trouver les créneaux où tout le monde est libre dans plusieurs agendas. La spec tient en une page, avec ses pièges : rendez-vous qui se chevauchent, marge autour des rendez-vous mais pas de la pause, rendez-vous qui passent minuit, heures mal formées à refuser. Le projet contient deux tests. Après chaque essai, on a passé 19 cas cachés que Claude n'a jamais vus.

Le message, identique pour les cinq essais : « Implémente creneaux_libres dans agenda.py en respectant toute la spécification de SPEC.md. Les tests de tests/ doivent passer (python3 -m pytest -q). Gère tous les cas de la spec, pas seulement ceux des tests. »

La commande

Chaque essai part d'une copie vierge du projet, dans un dossier temporaire. L'option `--safe-mode` coupe CLAUDE.md, skills, plugins, hooks et MCP : Claude ne voit que le projet. La commande, ici pour le duo Sonnet et Fable :

```
À COLLER

claude -p "$PROMPT" --safe-mode --model sonnet \
  --permission-mode acceptEdits \
  --allowedTools "Bash(python3 -m pytest*)" "Bash(python3*)" Read Edit Write
Glob Grep \
  --output-format stream-json --verbose \
  --max-budget-usd 4 --no-session-persistence \
  --advisor fable
```

Pour Fable seul : `--model fable`, sans `--advisor`. Pour le duo avec Opus : `--advisor opus`.

Les cinq essais

Coût au prix de liste de l'API, relevé dans le bloc `modelUsage` du message final de chaque journal :

- Sonnet 5.5 seul : 19 sur 19, 0,28 $, 106 secondes.
- Fable 5.2 seul : 19 sur 19, 1,27 $, 193 secondes.
- Sonnet 5.5 avec Fable 5.2 en conseiller, premier essai : 19 sur 19, 1,59 $ (Sonnet 0,35 $, Fable 1,24 $), 302 secondes, 2 appels au conseiller.
- Sonnet 5.5 avec Fable 5.2 en conseiller, second essai : 19 sur 19, 1,55 $ (Sonnet 0,32 $, Fable 1,23 $), 305 secondes, 2 appels.
- Sonnet 5.5 avec Opus 5.5 en conseiller : 19 sur 19, 0,85 $ (Sonnet 0,31 $, Opus 0,55 $), 243 secondes, 2 appels.

Les essais ont tourné en partie en même temps : les durées sont indicatives.

Pourquoi le duo Fable a coûté plus que Fable seul

Fable seul a relu sa conversation depuis le cache : 327 170 tokens pour 0,07 $. Le gros de sa note, c'est ce qu'il a écrit, 19 044 tokens pour 0,76 $. Le reste, 0,44 $, a servi à mettre la conversation en cache.

Fable en conseiller, premier essai, en deux appels : 67 941 tokens lus sans cache pour 0,54 $, et 17 372 tokens écrits, réflexion comprise, pour 0,69 $. Il a donc écrit presque autant que Fable seul pour faire toute la tâche. La doc de l'API annonce d'habitude 1 400 à 1 800 tokens par appel, réflexion comprise. Chez nous : environ 8 700 par appel avec Fable, 8 177 puis 5 758 avec Opus.

Au total, le conseiller a coûté presque autant que Fable qui fait tout, et le travail de Sonnet s'est ajouté par-dessus. La relecture sans cache pèse 0,54 $ sur ses 1,24 $, le reste vient de ce qu'il a écrit.

Ce qu'on en conclut

- Sur cette tâche, Fable en conseiller n'a rien économisé : 25 % puis 22 % de plus que Fable seul, pour le même score.
- Sonnet avec Opus en conseiller a coûté 33 % de moins que Fable seul, même score.
- Sonnet seul a fait 19 sur 19 pour 0,28 $, trois fois moins que le duo Sonnet et Opus. Cette tâche n'avait pas besoin de conseiller.

Ce qu'on n'en conclut pas

- Une seule tâche, de 2 à 5 minutes. La doc de Claude Code le dit : l'advisor apporte moins sur une tâche courte, où il y a peu à planifier. Notre test tombe dans ce cas.
- Un essai par réglage, deux pour le duo Fable. Et aucun écart de qualité ne pouvait apparaître, puisque tout le monde a eu 19 sur 19.
- On n'a pas lancé Opus seul. La promesse du menu, « near-Opus performance with reduced token usage », reste à vérifier.
- Sur une longue tâche où Sonnet fait beaucoup de routine entre deux appels, la part du conseiller pèserait moins dans la note. C'est une hypothèse, à tester.
- pytest manquait sur la machine : les sessions Sonnet ont écrit leur propre lanceur de tests, Fable seul est passé par uv. Du travail en plus pour chaque essai.
- Anthropic a mesuré de son côté. Sur son banc d'essai de code, Opus 5.5 avec Fable 5.1 en conseiller gagne 1,7 point, « at the edge of run-to-run noise », pour environ 2,1 fois le coût d'Opus seul. Son billet du 9 avril 2026 donnait, pour Sonnet 4.6 avec Opus 4.6, 2,7 points de plus sur SWE-bench Multilingual et 11,9 % de coût en moins par tâche.

---

6. Quand l'utiliser, quand t'en passer

Le bon terrain, d'après la doc :

- les longues tâches faites surtout de routine, où la qualité du plan décide du résultat : un gros refactor, un bug qui revient, une vérification avant de déclarer fini ;
- Sonnet en modèle principal, Opus en conseiller : le réglage que recommande Claude Code, et le seul duo moins cher que Fable seul chez nous.

T'en passer :

- sur les tâches courtes : rien à planifier, et un appel au conseiller peut coûter autant que tout le reste ;
- quand chaque tour demande le modèle le plus fort : change plutôt de modèle principal avec /model ;
- avec Fable en conseiller sur Pro, si tu ne surveilles pas tes crédits d'usage.

---

7. Mesurer chez toi

- Dans une session, `/usage` donne le total de la session, conseiller compris.

- Pour comparer, prends une vraie tâche longue, dans une copie de ton projet. Lance-la avec notre commande de la partie 5 en changeant seulement le modèle et le conseiller : Sonnet seul, Sonnet avec `--advisor opus`, puis ton réglage habituel.
- Dans chaque journal, le message final (type result) porte un bloc `modelUsage` : un coût par modèle, conseiller compris. Les appels au conseiller sont les blocs `server_tool_use` nommés advisor. Vérifie le résultat avec tes propres tests, et refais au moins deux fois le réglage qui gagne.
- L'option `--max-budget-usd` limite la dépense d'un essai (on a mis 4 $). Qu'elle compte les tokens du conseiller : à tester.

---

8. Les pièges

- **La relecture sans cache.** La doc de Claude Code le dit : le conseiller relit toute la conversation à chaque appel, sans rien réutiliser. Plus la session est longue, plus chaque appel coûte. L'API propose un cache pour le conseiller (option caching), rentable à partir d'environ trois appels d'après sa doc.
- **L'accord pour les crédits d'usage.** Sur les offres où Fable se paie en crédits, comme Pro, Claude Code demande ton accord une fois. Tant que tu ne l'as pas donné, `/advisor fable` n'enregistre rien et te renvoie vers `/model fable`, `claude --advisor fable` refuse de démarrer, et un conseiller Fable déjà enregistré est ignoré sans bruit. Pour donner l'accord : `/model fable`, choisis de continuer sur Fable, repasse sur Sonnet avec `/model sonnet`, puis règle `/advisor fable`.
- **claude -p ne demande rien.** En mode -p, et dans une appli Agent SDK qui n'affiche pas la question, Claude Code ne demande jamais : une requête Fable qui doit partir en crédits d'usage y part directement. Mets un plafond mensuel dans Settings > Usage avant de lancer des essais en -p.
- **Les versions.** Fable 5.2 demande Claude Code 2.1.321 ou plus, en conseiller comme en modèle principal. `/advisor` sans menu (appli, -p, SDK) : 2.1.260. Pour mettre à jour : `claude update`.
- **Les fournisseurs.** L'advisor marche avec l'API d'Anthropic, Claude Platform on AWS et Claude Models on Google Cloud (Anthropic managed), ces deux derniers à partir de 2.1.300. Pas sur Amazon Bedrock, Google Cloud's Agent Platform ni Microsoft Foundry.
- **Le réglage qui manque.** Hors AWS et Google Cloud, l'advisor dépend d'un réglage que Claude Code récupère chez Anthropic. Une variable comme `DISABLE_TELEMETRY` bloque cette récupération, et l'advisor reste éteint.
- **Les refus silencieux.** Si l'API refuse un duo, Claude Code renvoie la requête sans conseiller, sans message d'erreur. Un nouveau conseiller ne prend effet qu'après `/clear` ou `/compact`.
- **Fable en modèle principal.** Un conseiller Opus ou Sonnet n'est pas appliqué à Fable. Vérifie `/model` avant `/advisor`.

---

9. La méthode

Guide écrit le 15 novembre 2026. Relus le même jour : la doc de Claude Code (advisor, configuration des modèles), la page de l'API sur l'outil advisor et sa page sur le coût (section Advisor strategy), la fiche de Fable 5.2, le billet The advisor strategy du 9 avril 2026, la page des prix et l'article du centre d'aide sur les crédits d'usage.

Menus /advisor et /model relevés dans Claude Code 2.1.325, fermés par Échap sans rien choisir. Cinq essais lancés le 15 novembre entre 16 h 24 et 16 h 35, heure de Paris, journaux complets gardés, chaque coût recompté. On n'a rien installé. Pendant son essai, Fable seul a de lui-même téléchargé pytest dans un environnement temporaire, avec uv.

Abyane · @abyane_nb · eigenflow.fr

Eigenflow