

La Gueznet IA n°10

Les prompts ! · Fin octobre 2026

Les **56 prompts** trouvés dans la veille du mois, sortis de 15 publications, traduits en français et prêts à coller. Chacun avec le lien de son post d'origine.

56 prompts · 15 publications · 20 annonces creuses

Traduction intégrale, crochets compris · Aucun prompt corrigé ni réécrit

Comment lire ce catalogue

Les crochets sont des trous. Tout ce qui est entre crochets est à remplacer par votre situation réelle. Un prompt collé avec ses crochets intacts produit une réponse sur un cas imaginaire.

Je n'ai rien amélioré au passage. Un prompt mal fichu reste mal fichu. C'est une information sur son auteur. Les fautes et la ponctuation sont d'origine.

Vérifiez ce qui sort. Plusieurs de ces prompts réclament au modèle des volumes de recherche, des salaires ou des marges qu'il ne connaît pas. Il produira des chiffres tout de même, et ils auront l'air sérieux.

Les garde-fous des auteurs font partie des prompts. La série n°9 précise « sur votre propre visage et celui de personne d'autre ». Lancée sur le visage d'un tiers, cette recherche est de la surveillance, et elle est illégale en France comme dans la plupart des pays européens.

Les 15 séries

#	Série	Compte	Prompts
1	Le studio motion design avec Opus 5.5 : règles, one-liners, brief de réalisateur, critique, skill	@0xMovez	16
2	Motion design avec Opus 5.5 : le fichier de règles, le skill de vérification et les boucles /goal et /loop	@N01ennn	5
3	Sonoriser une vidéo animée en mesurant au lieu d'écouter, et trouver quoi animer avec Claude Motion	@Voxyz_ai	2
4	Trois sous-agents relecteurs avant de publier une vidéo sur X	@Voxyz_ai	1
5	Opus en conseiller, Sonnet aux commandes, Haiku en essaim, plus un HARNESS.md et une relecture en deux passes	@mirku21	4
6	L'audit de prompts du skill claude-api dans Claude Code	@Bober_smart	1
7	La règle AGENTS.md qui met en tête ce que l'agent attend de toi	@Voxyz_ai	1
8	Régler la durée du cache par sous-agent dans Claude Code	@Voxyz_ai	2
9	Demander une vidéo explicative plutôt qu'un texte	@karpathy	1
10	Une couche de décision devant l'agent : les questions posées à Jev et l'escalade vers Kimi K3	@starmexxx	2
11	Brancher Claude sur Gemini : quatre images d'un coup et le visionnage de vidéos	@artificial.corner	2
12	LLM local : les neuf prompts du cours en dix étapes	@0xMovez	9
13	Chercher un logiciel espion sur un téléphone, en lecture seule	@0xGoldTooth	5
14	Avant la mise en ligne : 14 attaques sur son propre site, puis 20 causes de lenteur	@Voxyz_ai	2

#	Série	Compte	Prompts
15	Auditer une appli Android ou un contrat de jeton avec REA (la seule part défensive du fil)	@0xCodila	3

1 Le studio motion design avec Opus 5.5 : règles, one-liners, brief de réalisateur, critique, skill

Movez (@0xMovez), 1 167 j'aime, 01/10/2026 · X · 16 prompts

Ressource : <https://x.com/0xMovez/status/2104216919033192746> · <https://github.com/JohnHeibel/PDoomVideo> · <https://github.com/JohnHeibel/ClaudeAnimationBase> · <https://github.com/buildwithhanif/claude-animation-skill> · <https://github.com/heygen-com/hyperframes> · <https://www.remotion.dev/docs/ai/skills> · <https://github.com/WinterArc21/Battle-of-Austerlitz-Film> · <https://github.com/guanmo-ai/awesome-ai-motion> · <https://github.com/athemeroy/awesome-opus-5-5-videos>

Le post annonce « un seul prompt »; l'article cité (un cours en 12 étapes) en donne seize, du one-liner au brief de réalisateur, à lancer dans Claude Code sur Opus 5.5 en effort xhigh ou max. Le code du moteur (rendu seek(t), ressorts, son, commandes ffmpeg) n'est pas un prompt et reste dans l'archive.

Garder une session par marque: le moteur de rendu, la synthèse audio et l'export déjà construits servent à la vidéo suivante.

Modèle de l'auteur, bâti sur la structure des posts de @twoclipping et @verbove.

Suppose le fichier lib/motion.js (fonctions spring() et track()) écrit dans l'article; Opus refait tout le fichier en une passe selon l'auteur.

Les trois images se fabriquent avant, avec les commandes ffmpeg de l'article (planche contact, bande de 12 images, test à 360 px).

1.1 · Le fichier CLAUDE.md du studio (à poser à la racine du projet)

```
# Règles du studio motion

## Contrat de rendu
- Chaque film est une fonction pure du temps : `window.seek(t)` peint l'image t.
- Pas de transitions CSS, pas de setTimeout, pas de requestAnimationFrame en mode rendu,
  aucun état transmis d'une image à l'autre. Bruit à graine uniquement (mulberry32), jamais Math.random.
- Rendu avec `node render.mjs`, encodage H.264 yuv420p, CRF 16.

## Rendu visuel
- Défauts bannis : titre centré sur un dégradé, tout qui apparaît en fondu,
  étiquettes dans les coins et cadres autour de l'image, halo sur l'habillage de l'interface, explosions
  de particules génériques.
- Une police de titrage, une police d'interface. Une seule couleur d'accent sauf si le brief dit
  autrement.
- Toutes les 2 à 4 secondes, quelque chose de nouveau doit se passer à l'écran.

## Son
- La musique et les bruitages sont synthétisés dans le code, sauf si une piste est fournie.
- Place les impacts sur la grille de temps mesurée (beats.json). Volume -14 LUFS.

## La boucle avant de me montrer quoi que ce soit
1. Fais le rendu d'une image par temps sous forme de planche contact et REGARDE-la.
2. Note-la de 1 à 10 sur : accroche dans les 2 premières secondes, lisibilité sur écran de téléphone,
  qualité du mouvement, variété, fidélité à la marque, synchro du son.
3. Corrige les 3 pires problèmes. Recommence jusqu'à ce que chaque note soit à 8 ou plus.
4. Seulement alors, fais le rendu complet.
```

1.2 · Le one-liner de la bande démo (pour tester l'installation)

fais une vidéo motion design dynamique de 15 secondes qui montre quel incroyable motion designer tu es, comme si c'était ta bande démo pour un CV. donne tout.

1.3 · Variante : 60 secondes calées sur un piano (motif de @kloss_xyz)

fais une bande démo motion design dynamique en 16:9 de 60 secondes qui montre tes vraies limites créatives.
Sound design de niveau S, pas de nappes de synthé génériques. Compose une musique originale au piano et cale chaque coupe dessus. Exporte en MP4 1080p.

1.4 · Variante : le garde-fou anti-bouillie IA (motif de @1littlecoder)

fais une vidéo motion design dynamique de 10 secondes qui présente qui tu es en tant qu'Opus 5.5. Évite les cadres et le texte dans les coins, les signes habituels d'une vidéo faite par IA.

1.5 · Variante : une histoire plutôt que des techniques (motif de @sonnylazard)

reprends l'énergie de ta bande démo, mais raconte une histoire : l'histoire de [SUJET] de [DÉBUT] à aujourd'hui, surprends-moi avec le storyboard. 45 secondes, vertical 9:16.

1.6 · Variante : la persona d'agence

fais une bande démo de 30 secondes comme si tu étais un studio de branding de niche pour fondateurs de startups.
Crée chaque élément graphique de zéro. Une seule couleur d'accent. Chaque plan utilise une technique différente.

1.7 · La bande démo braquée sur ton produit

Fais une vidéo motion design dynamique de 20 secondes pour [PRODUIT] ([URL]), avec l'énergie de la bande démo d'un motion designer. Donne tout.

Ressources

- Va sur le site. Utilise de vraies captures d'écran (Playwright), le vrai logo, les vraies couleurs et polices.
Enregistre tout dans ./assets et liste ce que tu as trouvé avant d'animer.
- Ne redessine jamais l'interface du produit en l'imaginant. Recadre et anime la vraie.

Histoire (un temps fort chacun, 2 à 4 secondes)

1. Accroche : le problème en 5 mots de typographie cinétique énorme.
2. Le produit apparaît, l'interface s'assemble pièce par pièce.
3. Trois fonctionnalités, chacune comme un moment d'interface avec un curseur qui fait une vraie action.
4. Un chiffre qui prouve que ça marche : [INDICATEUR].
5. Bloc logo final + [APPEL À L'ACTION].

Son

- Musique originale, 120 BPM, synthétisée dans le code. Clics d'interface et « whoosh » sur le temps.

Format : 1080x1920 (9:16) d'abord, puis 1:1 et 16:9 depuis la même timeline.

Avant le rendu complet, montre-moi une planche contact avec une image par temps.

1.8 · Partir d'une vidéo de référence

Référence : ./refs/launch.mp4 (et ./refs/frames/*.png)

1. Extrais une image toutes les 0,5 s avec ffmpeg. Étudie-les.
2. Écris ./docs/style_guide.md : palette (hex), typographie (famille, graisse, approche), durée des plans, types de transitions, mouvements de caméra, texture et grain, façon dont le texte entre et sort.
3. Écris ./docs/shotlist.md pour une vidéo de [DURÉE] s sur [SUJET] dans CE style.
Prends la grammaire de la référence, jamais son contenu, ses logos ni ses personnages.
4. Montre-moi les deux fichiers. Attends mon accord avant d'écrire du code.

1.9 · La spec en XML : un seul élément qui se transforme d'état en état

```
<inputs>
Demande-moi : mon produit + URL, 8 à 12 états d'interface qui racontent son histoire, les vraies données
affichées dans
chaque état, couleurs + polices de la marque + un accent, une piste libre de droits autour de 120 BPM,
les formats.
</inputs>

<direction>
Motion d'interface façon film produit. Un seul conteneur, jamais coupé : chaque état est le même élément
qui change
de taille, d'arrondi et de remplissage pendant que son contenu change derrière un bref flou. Un curseur
déclenche chaque changement.
Fond neutre chaud, un seul accent. Ressorts avec au plus un tout petit dépassement.
Interdits : easing rebondissant, halos, dégradés sur l'habillage de l'interface, explosions de
particules, temps morts.
</direction>

<structure>
120 BPM, 8 mesures, il se passe quelque chose sur chaque temps.
logo → bouton d'appel à l'action → champ e-mail (saisi) → changement → coche de réussite → carte du
tableau de bord
→ le graphique se dessine tout seul → infobulle au survol → palette ☞K → notification → logo.
</structure>

<build>
1. Un seul fichier HTML, un seul canvas, window.seek(t). Pas de transitions CSS, pas de minuteries, pas
d'état transmis.
2. Ressorts en forme close. Une valeur avec plusieurs cibles = somme d'un ressort par changement.
3. Le texte dans un conteneur qui se transforme entre après le début de la transformation, et sort avant
la suivante.
4. Indicateurs d'onglet : bord avant et bord arrière sur des ressorts différents, pour qu'ils s'étirent.
5. Grille de temps tirée de la piste (numpy/librosa). Démarre sur un premier temps de mesure. Sons
d'interface sur les pics mesurés.
6. Rendu dans Chrome headless à 60 i/s, 4 sous-images par image, fusionnées pour le flou de mouvement.
</build>

<gotchas>
N'utilise jamais will-change sur un élément que la caméra met à l'échelle (texte flou).
La dernière image doit être identique à la première, position et vitesse du curseur comprises.
</gotchas>

<start>
Demande les éléments d'entrée, puis montre-moi la liste des états sur la grille de temps avant d'écrire
du code.
</start>
```

1.10 · Route B avec Remotion (à taper dans Claude Code après `npx skills add remotion-dev/skills`)

```
/remotion-create un film de lancement de 20 s en 9:16 pour [PRODUIT], uniquement des ressorts, une seule
couleur d'accent
```

1.11 · Route B avec HyperFrames (après `npx hyperframes init`)

Avec `/hyperframes`, transforme `./notes.md` en vidéo de présentation de 45 secondes avec sous-titres cinétiques

1.12 · Remplacer les courbes d'easing par des ressorts

Remplace chaque courbe d'easing par les ressorts en forme close de `lib/motion.js`. Tout petit dépassement sur l'interface, aucun sur le texte. Toute valeur qui a plus d'une cible utilise `track()`.

1.13 · Le brief de réalisateur pour un film tourné toute la nuit

Tu es le réalisateur, l'animateur, le sound designer et l'ingénieur de rendu d'un film de [DURÉE] fait en code. Traite-le comme une production en plusieurs sessions. Ne te précipite pas vers un rendu final.

Le film en une ligne

[PITCH EN UNE PHRASE. Ce que le spectateur doit ressentir à la fin.]

Références et entrées

- `./refs/` : [vidéo / images / bibliothèque d'images]. Prends la grammaire, jamais le contenu.
- `./audio/track.wav` : utilise-la telle quelle. Mesure d'abord les temps avec `beats.py`.
- Skills disponibles : [`/remotion-best-practices` | `/hyperframes` | `/claude-animation`].
- API dans `.env` : [`ELEVENLABS_API_KEY`, `FAL_KEY`]. Budget : [X \$]. Sois économe.

Rendu visuel

[3 à 5 lignes : palette, typographie, texture, langage de caméra. Rendus interdits.]

Découpage par temps forts

0:00-0:02 accroche : [l'image la plus frappante]

0:02-0:10 [acte 1]

... un nouveau temps fort visuel toutes les 3 à 5 secondes

[FIN] la dernière image prépare la première (boucle)

Déroulé, avec étapes de validation

1. Écris `docs/style_guide.md` et `docs/shotlist.md` (chaque plan : images, caméra, texte, bruitages). Montre-moi la liste des plans. Puis continue sans attendre si je ne réponds pas dans les 10 minutes.
2. Construis des images fixes pour chaque plan. Planche contact. Critique.
3. Animatique en 960x540 avec un son provisoire. Corrige le rythme avant la finition.
4. Animation complète, passe de finition, passe son, rendu final.
5. Répartis le travail entre des sous-agents, un par chapitre. Écris d'abord `docs/ANIMATION_GUIDE.md` pour que chaque sous-agent code dans le même style.

Boucle de critique (chaque plan, au moins 3 tours)

Fais le rendu de 3 à 5 images fixes, note de 1 à 10 : accroche, lisibilité à 360 px de large, mouvement, composition, profondeur, synchro du son, finition. Consigne les notes + les 3 plus gros problèmes dans `docs/review_log.md`. Corrige. Recommence jusqu'à ce que tout soit à 8 ou plus.

Livrables

`out/final.mp4` · `out/loop_check.mp4` · `out/poster.png` · `out/contact.png` · `README.md`

1.14 · La passe de critique sur ses propres images

Ouvre out/contact.png, out/strip.png et out/phone.png et regarde-les vraiment.
Sois un directeur motion exigeant, pas un auteur fier de lui.

Note de 1 à 10 : accroche dans les 2 premières secondes · lisibilité sur téléphone · qualité du mouvement (ressorts, pas d'images mortes) · variété (du nouveau toutes les 2 à 4 s) · composition · fidélité à la marque · synchro du son.

Liste les 3 plus gros problèmes avec leur minutage. Cherche en particulier : du texte qui se chevauche pendant

les changements, tout ce qui glisse au lieu de s'amortir, les étiquettes dans les coins et les cadres, les plans centrés sur un dégradé, le texte flou mis à l'échelle, un temps mort où rien ne se passe, un à-coup au raccord de la boucle.

Corrige-les, refais le rendu des seules secondes concernées, montre-moi la nouvelle planche contact et les nouvelles notes.

1.15 · Le skill motion-reel (tout le cours en un fichier SKILL.md)

```
---
name: motion-reel
description: Fait une vidéo motion design de produit ou une bande démo rendue à partir de code. À
utiliser quand
  l'utilisateur demande une vidéo de lancement, une bande démo, une vidéo produit, une vidéo explicative
animée ou une pub animée.
---

# Motion reel

## Éléments à réunir d'abord
Produit + URL, durée, formats (9:16 / 1:1 / 16:9), couleurs + polices de la marque,
une référence (image, vidéo ou dossier d'images), la musique (fichier ou « synthétiser »).

## Chaîne de production
1. Récupère les ressources depuis l'URL avec Playwright dans ./assets. Liste-les.
2. S'il y a une référence, écris docs/style_guide.md à partir d'elle.
3. Mesure ou synthétise la musique. beats.py → beats.json.
4. Écris docs/shotlist.md sur la grille de temps. Montre-la et attends mon accord.
5. Construis index.html avec window.seek(t) en utilisant les ressorts de lib/motion.js. Suis CLAUDE.md.
6. Planche contact → passe de critique (voir prompts/critique-pass.txt) → correction. 3 tours minimum.
7. node render.mjs → sfx.mjs → mixage à -14 LUFS → out/final.mp4, tous formats.
8. Livre final.mp4, contact.png, poster.png. Dis ce que tu améliorerais ensuite.

## Règles strictes
- Uniquement la vraie interface du produit. N'invente jamais d'écrans.
- Pas de Math.random, pas de minuteries, pas de transitions CSS en mode rendu.
- Interdits : étiquettes dans les coins, titre centré sur un dégradé, tout qui apparaît en fondu.
```

1.16 · L'appel du skill une fois installé

```
/motion-reel pour [URL], 20 s, vertical, référence ./refs/frame.png
```

2 Motion design avec Opus 5.5 : le fichier de règles, le skill de vérification et les boucles /goal et /loop

NO1ennn (@NO1ennn), 5 975 j'aime, 08/10/2026 · X · 5 prompts

Ressource : <https://x.com/NO1ennn/status/2107826459804840072>

Le post liste 20 dépôts GitHub de motion design; les prompts sont dans l'article cité (« Opus 5.5 en studio: motion design, harnais et boucles »). Les commandes /goal et /loop sont celles d'Addy Osmani, reprises par l'auteur.

Les « contrôles ci-dessous » sont, dans l'article: un test de fumée (six compteurs à zéro: erreurs d'exécution, bandeaux d'erreur, couleurs invalides, texte hors scène, texte hors zone, chevauchements de texte), un contrôle de raccord (image 0 et image d'une boucle plus loin identiques au pixel près) et une relecture des images fixes.

`every 24h` est laissé tel quel: c'est l'intervalle lu par /loop. L'étiquette `bug` est celle du dépôt.

2.1 · Les règles maison, à mettre dans le fichier de règles avant toute animation

```
fonction pure de l'image : pas de minuteurs, pas de transitions CSS, pas d'aléatoire sans graine
la boucle se referme : chaque mouvement répété divise la durée de la boucle, pour que la dernière image
s'enchaîne sur la première sans saut
la couleur a un rôle : le noir et blanc fait la surface, la couleur ne va que là où elle a un sens. sur
mes pièces Grok Bot, ce sont les agents et rien d'autre
données réelles seulement : chaque étiquette remonte à une source. pas de chiffres inventés, pas
d'interface imaginée
rien ne part sans contrôle : aucun rendu ne sort tant que les contrôles ci-dessous ne reviennent pas
propres
```

2.2 · La consigne de vérification à écrire dans un skill (conseil de l'équipe Claude Code)

```
ne déclare jamais une modification d'interface terminée sur la seule foi de l'édition. ouvre la page,
utilise la modification, fais une capture avant et après, vérifie la console pour de nouvelles erreurs,
lance une trace de performance, et si une étape échoue, corrige et recommence.
```

2.3 · Un /goal avec règle d'arrêt mesurable (Lighthouse)

```
/goal Refactorise la couche de récupération des données jusqu'à ce que le score de performance Lighthouse
soit >= 92 et le LCP < 1,8 s dans la sortie de Lighthouse CLI.
Ne change l'API publique d'aucun hook. Chaque tour doit améliorer une métrique ;
abandonne après deux tours sans amélioration. Arrête-toi après 10 tours.
```

2.4 · Une /loop quotidienne qui lance un /goal sur les bugs

```
/loop every 24h "Vérifie sur GitHub les tickets étiquetés 'bug'. S'il y en a un,
utilise /goal pour implémenter un correctif jusqu'à ce que tous les tests locaux passent, et pousse la
branche."
```

2.5 · Le /goal du studio : six compteurs à zéro et boucle sans raccord

```
/goal Le test de fumée affiche zéro sur les six compteurs et l'image 0 correspond
à l'image 720 au pixel près. Corrige le pire échec à chaque passe.
Abandonne si une passe ne corrige rien. Arrête-toi après 5 passes.
```

3 Sonoriser une vidéo animée en mesurant au lieu d'écouter, et trouver quoi animer avec Claude Motion

Vox (@Voxyz_ai), 826 j'aime, 11/10/2026 · X · 2 prompts

Ressource : <https://support.claude.com/en/articles/17454997-get-started-with-claude-motion>

Deux prompts pour Opus 5.5 dans Claude Code. Le premier est dans le post; le second dans le post cité du même auteur. L'auteur a vu librosa se caler sur le contretemps, avec une grille décalée de 0,22 seconde.

3.1 · Ajouter le son à une vidéo

Ajoute du son à cette vidéo. Tu ne peux pas entendre, alors mesure chaque étape et ne te fie pas à ton intuition. Installe d'abord ffmpeg et librosa s'ils manquent.

1. Utilise de la musique sous licence : une piste que je te donne, ou une piste sous la licence gratuite de Mixkit. Note la source et la licence. Ne synthétise ni la musique ni les effets sonores en code.
2. Trouve les temps avec librosa. Il se cale parfois sur le contretemps, alors trace la forme d'onde sous 120 Hz face à la grille des temps et zoome sur quelques mesures. Si les grosses caisses ne tombent pas dessus, décale toute la grille.
3. Place les grands mouvements, comme les changements de scène et l'apparition des titres, sur le temps. Si tu peux modifier l'animation, modifie-la. Sinon, déplace le point de départ de la musique.
4. Donne à chaque coupe et à chaque grand mouvement un fichier d'effet sonore tout prêt : whoosh pour un mouvement, pop pour une apparition, clic pour les clics, bruits de frappe pour la saisie. Utilise d'abord la sonothèque existante du projet, comme celle fournie avec HyperFrames, et demande-moi s'il n'y en a pas.
5. S'il y a une voix off, c'est l'élément le plus fort, et la musique baisse quand quelqu'un parle. Les effets sonores passent au-dessus de la musique. Vise environ -16 LUFS au global avec des crêtes sous -4 dBFS, pour que rien ne sature après la compression AAC ou sur un haut-parleur de téléphone en mono. Mesure le MP4 exporté avec ebur128 de ffmpeg. Ne mesure pas le WAV.
6. Avec ffmpeg, extrais une image avant et après chaque repère du MP4 exporté, et trace une forme d'onde pour comparer. Puis choisis 5 instants que je vérifierai au casque. Ne me dis pas que ça sonne bien.

3.2 · Trouver dans son projet ce qui mérite une animation

Lis ce guide : <https://support.claude.com/en/articles/17454997-get-started-with-claude-motion>

Puis suis la démarche de Claude Motion et parcours le README de ce projet, sa page d'accueil, son parcours de prise en main, ses dernières notes de version et tous les graphiques qu'il contient. Trouve trois sortes de choses qui méritent d'être transformées en animation :

1. Un enchaînement qu'une seule capture d'écran ne peut pas expliquer, comme les étapes que suit un nouvel utilisateur pour démarrer.
 2. Un changement qui tombe à plat sous forme de simple chiffre, comme la vitesse de la nouvelle version comparée à l'ancienne.
 3. La nouvelle fonctionnalité de la dernière version qui mérite le plus d'être montrée.
- Si le projet n'a pas l'une de ces choses, dis-le simplement. Ne force pas.

Pour chaque idée, suis le guide : dis à qui elle s'adresse, où elle sera diffusée et combien de temps elle dure, puis écris un storyboard avec la durée en secondes de chaque scène et les mots, chiffres et captures d'écran qui apparaissent à l'écran. N'utilise que du contenu réel du projet. Classe les idées selon leur utilité pour moi.

Une fois que j'en ai choisi une, construis-la sous forme d'animation en code avec HyperFrames (installe-le d'abord s'il n'est pas installé), avec tous les textes, chiffres et durées regroupés à un seul endroit. Plus tard, quand je dirai « ralentis la deuxième scène » ou « change ce chiffre en 18 % », ne modifie que cette chose-là. Fais d'abord un rendu de brouillon, extrais une image par seconde et passe-les en revue, corrige les problèmes, puis exporte un MP4 en 1080p.

Montre-moi d'abord les storyboards. Ne construis rien tant que je n'ai pas choisi.

4 Trois sous-agents relecteurs avant de publier une vidéo sur X

Vox (@Voxyz_ai), 76 j'aime, 12/10/2026 · X · 1 prompt

Le prompt du son qui accompagnait ce post est au n°3 (même auteur, même texte).

L'auteur dit d'envoyer tout le post à Claude Code avec une ligne en plus à la fin: le premier bloc reprend le post (sans l'anecdote des scores passés de 5/6/6 à 8/8/8 ni le renvoi au post sur le son), suivi de cette ligne. Le second bloc est le prompt du post cité pour le son.

4.1 · Les 9 contrôles et la mise en place des relecteurs

Astuce Claude Code : avant de publier sur X une vidéo réalisée par Opus 5.5, fais-lui envoyer 3 sous-agents relecteurs qui signalent les problèmes, horodatage à l'appui, selon ces 9 contrôles, et qu'il ne te l'apporte qu'une fois que les 3 lui donnent 7 ou plus.

Ouverture

- Quand X lance une vidéo en lecture automatique, l'image 0 sert de vignette. Est-ce qu'elle s'ouvre sur un écran vide, une image noire ou quelque chose encore en fondu d'entrée ?
- Regarde les 3 premières secondes sans le son. Est-ce qu'on comprend de quoi il s'agit ?

Image

- Ne relis que des images extraites du MP4 exporté. Ce qui semble correct dans l'aperçu peut disparaître au rendu.
- Enregistre une démo qui fait des allers-retours sur une page, et la moitié de ce que tu captures se lit à l'envers. Chaque extrait que tu montes se lit-il vers l'avant ?
- Y a-t-il au milieu une image noire ou un fondu au noir qui fait croire que c'est fini ?
- La dernière image ramène-t-elle à la première ? X lit les vidéos en boucle.

Contenu

- Chaque sous-titre et chaque nombre correspondent-ils au matériau source ? Une affirmation comme « le premier » a-t-elle une source ?
- Des sous-titres ou des informations clés se trouvent-ils dans les 8 % du bas ? Le lecteur mobile de X couvre cette bande.
- Réduis à la largeur d'un téléphone. Peux-tu encore lire le texte ?

Lance les 3 sur Opus 5.5 en effort élevé. Après chaque série de corrections, fais relire la nouvelle version par le même sous-agent. Il se souvient de ce qu'il a signalé au tour précédent et vérifie ces points en premier.

Si la session principale met plus de 5 minutes à corriger et à refaire le rendu, le cache des sous-agents expire. Quand ils reprennent, chaque image des tours précédents est retraitée de zéro, et ça pèse davantage sur tes limites d'utilisation. Le prompt donne donc à chacun un cache d'une heure.

Crée 3 sous-agents relecteurs dans ~/.claude/agents, un pour chaque groupe ci-dessus : hook-reviewer, motion-reviewer et content-reviewer. Utilise model opus et effort high, et ne leur donne que Read, Glob et Grep. Ils relisent et ne modifient jamais rien. Dans le frontmatter de chacun, écris experimental: et mets cacheTtl: 1h sur la ligne suivante, en retrait. Si mon Claude Code est plus ancien que 2.1.248, ou si subagentPromptCacheTtl est défini dans mes réglages, dis-le-moi d'abord.

La première fois, demande-moi où se trouve le projet vidéo et quel MP4 est le dernier export. Installe d'abord ffmpeg ou Pillow si l'un des deux manque.

Ensuite, chaque fois que j'exporte une vidéo :

1. Extrais des images du MP4 exporté et inscris sur chacune son horodatage (si ffmpeg n'a pas drawtext, utilise Pillow). hook-reviewer reçoit l'image 0 et les images des 3 premières secondes. motion-reviewer reçoit des planches-contact à 4 images par seconde, plus la première et la dernière image. content-reviewer reçoit une image en taille réelle pour chaque sous-titre, une série à la largeur d'un téléphone, et les chemins du script des sous-titres et du matériau source.
2. Liste les problèmes sous la forme « temps · P0/P1/P2 · problème · correctif », et que chacun termine par une note de 1 à 10.
3. Corrige d'abord les P0, puis les P1, refais le rendu, et enregistre-le comme nouvelle version (v1, v2...). Enregistre les problèmes, les notes et les ID d'agent des 3 sous-agents de chaque tour dans un dossier review/.
4. Utilise SendMessage avec chaque ID d'agent pour faire relire la nouvelle version par le même sous-agent. N'en lance pas de nouveaux. Ils vérifient d'abord les problèmes du tour précédent. Dès que les 3 donnent 7 ou plus, ou après 3 tours de relecture, arrête et remets-la-moi. Lance de nouveaux sous-agents pour la vidéo suivante.

Montre-moi d'abord les fichiers que tu vas créer, et ne les écris pas tant que je n'ai pas confirmé.

5 Opus en conseiller, Sonnet aux commandes, Haiku en essaim, plus un HARNESS.md et une relecture en deux passes

mirku (@mirku21), 6 631 j'aime, 07/10/2026 · X · 4 prompts

Ressource : <http://code.claude.com/docs/en/advisor>

Le premier prompt est à coller dans Claude Code selon l'auteur; certains réglages qu'il cite (advisorModel, pool de sous-agents) sont les siens, le prompt demande de toute façon de montrer les diffs avant d'appliquer. Les trois suivants viennent de son X Article cité (« Harness Engineering »).

5.1 · Configurer l'orchestration hiérarchique avec conseiller

Configure mon espace de travail Claude Code pour une orchestration hiérarchique avec conseiller :

1. Audite ~/.claude/settings.json et la configuration du projet pour trouver les rôles de modèle adaptés au rôle principal, aux sous-agents et au conseiller :
 - > Fixe le modèle de la session principale : sonnet avec effortLevel: high pour l'exécution principale
 - > Fixe le modèle des sous-agents : haiku avec effortLevel: medium pour la découverte de fichiers en parallèle et la récupération de documentation
 - > Fixe le modèle du conseiller : opus en astreinte pour la revue stratégique
2. Active l'outil conseiller dans ~/.claude/settings.json :
 - > Règle advisorModel sur claude-opus-5-5
 - > Active le pool de répartition parallèle des sous-agents (3 workers)
3. Configure des points de consultation automatique du conseiller dans ~/.claude/CLAUDE.md :
 - > Consulte /advisor opus avant de finaliser un plan d'architecture qui touche plusieurs fichiers
 - > Appelle automatiquement /advisor opus quand le même test ou la même erreur de compilation échoue deux fois
 - > Impose un audit du diff par le conseiller, au regard du contrat, avant de déclarer une tâche terminée ou de préparer un commit git
4. Restreins le périmètre des sous-agents :
 - > Les sous-agents Haiku renvoient uniquement des résumés AST structurés et des extraits de documentation, sans modifier le contexte de la session principale

Montre d'abord chaque diff de configuration. N'applique aucune modification avant confirmation

5.2 · HARNESS.md, à enregistrer à la racine du projet

```
# Contrat d'exploitation du projet

## Limites d'intervention
- Périmètre autorisé : ne modifie que les fichiers explicitement assignés dans la tâche en cours
- Actions interdites : ne supprime jamais de tests existants, n'ajoute jamais de dépendances non vérifiées

## Carte du projet
- /src : code source de l'application
- /tests : suites de tests de validation
- /docs/decisions.md : journal durable des choix d'architecture acceptés

## Protocole de vérification
Avant de marquer une tâche comme terminée, lance ces contrôles dans l'ordre :
1. Lance le linter local : `npm run lint`
2. Lance la suite de tests : `npm test`
3. Vérifie que le format de sortie correspond au schéma demandé

## Escalade en cas d'échec
Si un test échoue deux fois sur la même erreur :
- Arrête les modifications autonomes
- Affiche le journal d'erreur exact
- Énonce le correctif proposé et attends la confirmation de l'utilisateur
```

5.3 · Passe 1 : l'exécution

Rédige la spécification technique de [Fonctionnalité X].
Respecte les contraintes listées dans HARNESS.md.
N'évalue pas encore ta propre production.
Livre le brouillon brut et énonce toutes les hypothèses que tu as faites.

5.4 · Passe 2 : la relecture adverse

Relis le brouillon ci-dessus en relecteur senior sceptique.
Vérifie ces points de défaillance précis :

1. Les cas limites sont-ils gérés quand les tableaux en entrée sont vides ?
2. La logique introduit-elle des dépendances supplémentaires ?
3. La solution viole-t-elle une contrainte de HARNESS.md ?

Liste chaque point de défaillance trouvé.
Puis livre la version révisée qui comble ces manques.

6 L'audit de prompts du skill claude-api dans Claude Code

Bober_smart (@Bober_smart), 609 j'aime, 01/10/2026 · X · 1 prompt

À taper dans Claude Code. Selon l'auteur, la commande audite tes skills, ton CLAUDE.md et tes prompts, retire ce qui ralentit le modèle et réécrit le tout selon le guide officiel d'Opus 5.5 (attribution à Lance Martin et « +300 % » non vérifiés). L'article cité (second cerveau en dossiers Markdown) annonce un exemple de CLAUDE.md, mais il est en image et absent de l'archive.

7 La règle AGENTS.md qui met en tête ce que l'agent attend de toi

Vox (@Voxyz_ai), 476 j'aime, 05/10/2026 · X · 1 prompt

Section à ajouter à un AGENTS.md (lu par Codex, et par Claude Code en l'absence de CLAUDE.md). Le fichier AGENTS.md en six sections du post cité est en image, absent de l'archive.

```
## 7. Commence par ce dont tu as besoin de ma part
Quand une longue tâche est terminée, ou que tu es bloqué et as besoin que je fasse quelque chose, liste ce dont tu as besoin de ma part tout en haut de ta réponse. Un seul point par élément, numérotés. Précise ce que je dois faire, où, et ce que je dois te renvoyer quand j'ai fini, par exemple « Connecte-toi à Stripe, exporte le CSV des remboursements de septembre, mets-le dans inbox/ ». Ce qui te bloque le plus passe en premier. Si tu n'as besoin de rien de ma part, dis-le simplement.
```

8 Régler la durée du cache par sous-agent dans Claude Code

Vox (@Voxyz_ai), 126 j'aime, 12/10/2026 · X · 2 prompts

Ressource : <https://code.claude.com/docs/en/prompt-caching>

À envoyer à Opus 5.5 dans Claude Code avec le lien de la doc. Le post commence par quatre extensions officielles à installer, une commande à la fois (`/plugin install receipts@claude-plugins-official`, `session-report@claude-plugins-official`, `claude-code-setup@claude-plugins-official`, `/plugin enable cc-plugin-you-should-know@builtin`), puis `/receipts` et les deux phrases du 3.2.

8.1 · Le cache d'une heure seulement pour les sous-agents qui attendent

```
Lis cette doc, ainsi que les docs sur les sous-agents vers lesquelles elle renvoie, et règle mes durées de cache selon la tâche :
1. Lance d'abord claude --version. Régler cacheTtl sur un seul sous-agent demande la 2.1.248 ou une version plus récente. Si la mienne est plus ancienne, dis-le-moi. Ne la mets pas à jour toi-même.
2. Liste chaque sous-agent dans ~/ .claude/agents et .claude/agents. Pour ceux qui peuvent rester inactifs plus de 5 minutes en cours de route (tests qui tournent, builds, attente de la CI), ou que je reprendrai plus tard, ajoute ces deux lignes en haut du fichier :
experimental:
  cacheTtl: 1h
S'il y a déjà un bloc experimental, ajoute cacheTtl dessous. N'en écris pas un second. Laisse sur la valeur par défaut ceux qui vont d'une traite et finissent en quelques minutes. Donne une ligne de justification pour chacun.
3. Vérifie ~/ .claude/settings.json, .claude/settings.json et .claude/settings.local.json (y compris leurs blocs env) ainsi que mes variables d'environnement actuelles, à la recherche de subagentPromptCacheTtl, CLAUDE_CODE_SUBAGENT_PROMPT_CACHE_TTL ou FORCE_PROMPT_CACHING_5M. Si l'une d'elles est définie, dis-moi qu'elle prend le pas sur les réglages par sous-agent et demande-moi s'il faut la supprimer.
4. Demande-moi si je suis sur un abonnement, une clé API ou un fournisseur cloud. Ne lis aucune clé toi-même. Sauf si je suis sur un abonnement dans les limites de son forfait, demande-moi si je m'absente souvent plus de 5 minutes en pleine session, et si c'est le cas, règle promptCacheTtl sur 1h.
Montre-moi d'abord ce que tu vas changer, et n'écris rien tant que je n'ai pas confirmé.
```

8.2 · Les deux phrases après installation des extensions

```
génère un rapport de session  
recommande des automatisations pour ce projet
```

9 Demander une vidéo explicative plutôt qu'un texte

Andrej Karpathy (@karpathy), 56 062 j'aime, 02/10/2026 · X · 1 prompt

Karpathy classe les formats de sortie du moins au plus parlant: un texte en ASD-STE100 (l'anglais technique contrôlé de la maintenance aéronautique, qu'il adoucit parfois en demandant « à 80 % du chemin vers ASD-STE100 »), un schéma, une page « en HTML », puis la vidéo explicative. Seul ce dernier est donné comme prompt complet. Sans clé ElevenLabs, il propose de demander au modèle une alternative gratuite qui tourne sur ta machine.

```
Crée une vidéo explicative dans le style de 3b1b sur X. Utilise ma clé API ElevenLabs pour la narration audio.
```

10 Une couche de décision devant l'agent : les questions posées à Jev et l'escalade vers Kimi K3

starmex (@starmexxx), 743 j'aime, 12/10/2026 · X · 2 prompts

Ressource : <https://x.com/starmexxx/status/2108159970017488953>

Le post parle d'un moteur local (colibri) sans donner de prompt; les prompts sont dans l'article cité, un journal de montage où Jev (API payante de TypeSafe, sur liste d'attente) tranche les décisions et Kimi K3 rédige. Les deux blocs sont des chaînes du code Python de l'auteur.

Les accolades sont des variables Python remplies par le code (confiance, probabilités, état).

Réunies depuis six blocs de code de l'article (aiguillage, urgence, sécurité, porte avant publication, compactage, routage entre modèles). Les clés `research`, `write`, `review`, `fast`, `powerful` sont des identifiants du code.

10.1 · Le prompt d'escalade quand la confiance de Jev est sous le seuil (0,85)

```
Un classifieur était sûr à {a.confidence:.0%}.  
Ses probabilités : {a.probabilities}
```

```
État :  
{state}
```

```
Choisis la prochaine étape et dis ce qui rendait ce cas ambigu.
```

10.2 · Les questions typées posées à Jev, regroupées

```
[Choix] Choisis la prochaine étape pour ce brief de recherche.  
  research : Collecte les preuves encore nécessaires pour l'objectif  
  write : Rédige le brief à partir de preuves suffisantes  
  review : Objectif flou, hors périmètre, ou travail terminé  
  
[Score] Évalue l'urgence de cette demande.  
  niveaux : faible, moyenne, haute, critique  
  
[Vrai ou faux] L'action demandée peut être exécutée sans approbation humaine.  
  
[Vrai ou faux, seuil 0,95] Ce brouillon est complet, sourcé, et sûr à publier.  
  
[Score] Quelle est la pertinence de cette entrée par rapport à l'objectif en cours ?  
  niveaux : hors sujet, contexte, utile, critique  
  
[Routage] Choisis le modèle le moins coûteux capable d'accomplir la tâche.  
  fast : Recherches directes, extraction, modifications localisées.  
  powerful : Architecture et décisions à fort enjeu.
```

11 Brancher Claude sur Gemini : quatre images d'un coup et le visionnage de vidéos

artificial.corner (@artificial.corner), 2 061 j'aime, 08/10/2026 · [Instagram](#) · 2 prompts

Annoncés: 3, servis: 2.

Deux prompts sont dans la légende; le troisième usage (brancher Claude sur ses carnets NotebookLM) est réservé au guide envoyé en DM.

11.1 · Quatre images Nano Banana d'un coup, dans ton style

Dans ce dossier, lis ma clé API Gemini dans gemini-key.txt et ne l'affiche jamais à l'écran. Demande au modèle Gemini 4 images d'un coup, en 16:9, à partir de ce prompt : [ton prompt d'image]. Enregistre-les dans un dossier nommé output

11.2 · Faire regarder ses vidéos par Gemini et en tirer des notes horodatées

Envoie chaque vidéo du dossier videos au modèle Gemini et demande : « Regarde cette vidéo. [Ce que tu veux en tirer]. Donne l'horodatage de chaque point. » Enregistre chaque réponse dans un fichier texte à côté de sa vidéo.

12 LLM local : les neuf prompts du cours en dix étapes

Movez (@0xMovez), 2 278 j'aime, 04/10/2026 · [X](#) · 9 prompts

Ressource : <https://drive.google.com/file/d/1bPfAgQOAUb9NzTjOn1BkboqllByjTTT0/view?usp=sharing>

Les prompts viennent de l'article cité « How to set up your first Local LLM in 10 Steps » (<https://x.com/0xMovez/status/2106761689123139973>), qui installe Ollama et sert les modèles qwen3:4b et qwen3.8:27b

en local. Dans les prompts lus par un script, les clés et valeurs JSON restent en anglais: le code les lit telles quelles.

À donner à n'importe quel modèle de conversation, avant d'acheter du matériel.

Prompt système du petit modèle (qwen3:4b, température 0, sortie JSON) dans le script Python de l'étape 9.

À garder dans un fichier et à relancer sur chaque modèle et chaque quantification envisagés.

Texte passé à `ollama run qwen3:4b --think=false` par un hook git; `%s` reçoit le diff indexé (12 000 caractères au plus).

Prompt système du script de tri par lots (qwen3:4b, 8 requêtes en parallèle). L'auteur conseille de vérifier 50 étiquettes à la main avant de faire confiance aux 950 autres.

Prompt système du script de questions-réponses sur un dossier `notes/` (qwen3.8:27b). Le script envoie les extraits sous l'étiquette `CONTEXT:` : si tu traduis ce prompt, traduis aussi cette étiquette dans le code.

Message envoyé à qwen3.8:27b avec l'image et un schéma JSON qui contient un champ `unreadable`.

Fichier de skill pour Claude Code branché sur le modèle local (étape 8).

Consigne passée à `ollama run qwen3.8:27b --think=false` par un script shell qui lui envoie les commits depuis la veille et les TODO ouverts du dépôt (cron à 6 h 30).

12.1 · Dimensionner son installation à partir de sa semaine

Tu dimensionnes pour moi une installation de LLM local. Ne me pose aucune question. Travaille à partir de la liste ci-dessous.

MES TÂCHES IA DE LA SEMAINE DERNIÈRE

[colle 10 à 20 tâches réelles : ce que c'était, la longueur de l'entrée, la fréquence]

Pour chaque tâche, produis une ligne de tableau :

tâche	données privées ? (oui/non)	taille de l'entrée (tokens, approximative)	exécutions par jour	nécessite un raisonnement de pointe ? (oui/non, raison en une ligne)	plus petite classe de modèle qui suffirait (4B / 12B / 27B / 100B+ / API de pointe)
-------	-----------------------------	--	---------------------	--	---

Puis donne-moi trois totaux :

1. le nombre d'heures par jour pendant lesquelles un modèle local serait occupé
2. le plus grand contexte dont une seule tâche a besoin
3. la part des tâches qui doivent rester sur une API de pointe

Ne recommande pas de matériel. N'arrondis pas à la hausse. Si une tâche n'est pas claire, marque-la « pas clair » au lieu de deviner.

12.2 · Le routeur qui envoie chaque tâche au bon modèle

Tu es un routeur. Lis la tâche et réponds uniquement en JSON :

```
{"lane": "small" | "big" | "cloud", "reason": "<12 mots max>"}
```

small = classer, extraire, étiqueter, reformuler, décisions oui/non, un seul appel d'outil

big = écrire ou modifier du code, tâches en plusieurs étapes, tout ce qui touche à des fichiers privés

cloud = décisions d'architecture inédites, tâches qui ont échoué deux fois en local

En cas d'hésitation entre deux voies, choisis la moins chère.

12.3 · La batterie de test à passer à chaque modèle

Exécute chacune des tâches ci-dessous. Pour chaque tâche, affiche : la réponse, puis une ligne « AUTO-VÉRIFICATION : » qui indique une façon dont la réponse pourrait être fausse.

1. [un vrai bug de mon dépôt + la sortie du test qui échoue]
2. [une clause de contrat] -> extrais les parties, les dates, les montants en JSON
3. [20 messages de support] -> étiquette chacun : facturation / bug / fonctionnalité / autre
4. [une capture d'écran d'un tableau de bord] -> liste chaque nombre que tu peux lire
5. [un document de 30 pages] -> réponds à trois questions dont les réponses sont en page 24
- ...
10. Une tâche où la bonne réponse est « Je ne peux pas le dire à partir de cette entrée. »

12.4 · Le message de commit rédigé par un modèle de 3 Go

Rédige un message de commit git pour ce diff.
Ligne 1 : résumé à l'impératif, 60 caractères max.
Puis une ligne vide, puis jusqu'à 3 puces : ce qui a changé et pourquoi.
Pas de préambule. Pas de blocs de code.

%s

12.5 · Le tri de mille tickets de support pendant la nuit

Étiquette ce ticket de support. Réponds uniquement en JSON :

```
{"category": "billing" | "bug" | "feature" | "account" | "other",  
  "urgency": 1 | 2 | 3,  
  "needs_human": true | false,  
  "summary": "<15 mots max>"}
```

urgence 3 = argent perdu, données perdues, ou impossible de se connecter.
Si le ticket n'est pas clair, category vaut "other" et needs_human vaut true.

12.6 · Interroger ses propres documents sans base vectorielle

Réponds uniquement à partir du CONTEXTE. Après la réponse, cite la phrase exacte sur laquelle tu t'es appuyé. Si le contexte ne contient pas la réponse, réponds « Absent de ces documents. » N'utilise aucune connaissance extérieure.

12.7 · Lire la photo d'une facture en JSON

Extrais les champs de la facture. Liste sous 'unreadable' tout champ que tu n'as pas pu lire clairement. Ne devine jamais un nombre.

12.8 · Le skill de relecture avant commit pour un agent local

```
---
name: local-review
description: Relit le diff indexé avant un commit. À utiliser quand l'utilisateur dit
  « relis », « vérifie mes modifications », ou demande si quelque chose peut être commité sans risque.
---
# Relecture locale

1. Lance `git diff --staged`. S'il est vide, dis-le et arrête-toi.
2. Lis en entier chaque fichier modifié, pas seulement les morceaux du diff.
3. Rends compte dans cet ordre : bugs, tests manquants, modifications risquées. 10 lignes max.
4. Pour chaque bug, donne fichier:ligne et un correctif en une ligne. Ne modifie aucun fichier.
5. Si tu n'es pas sûr que quelque chose soit un bug, étiquette-le « pas sûr ».
6. Termine par un seul mot sur sa propre ligne : COMMIT ou ATTENDRE.
```

12.9 · Le digest du matin écrit pendant la nuit

Rédige un digest du matin en markdown.
Sections : Livré hier / Semble inachevé / À faire en premier aujourd'hui (3 éléments).
N'utilise que des faits tirés de l'entrée. Si l'entrée est vide, écris « Rien de nouveau. »

13 Chercher un logiciel espion sur un téléphone, en lecture seule

goldtooth.exe (@0xGoldTooth), 10 230 j'aime, 20/10/2026 · X · 5 prompts

L'auteur a examiné le téléphone de sa sœur, à sa demande. Conditions de l'auteur, à garder avec les prompts: ton propre téléphone, ou celui d'un proche qui te le demande; Claude Code en lecture seule, rien n'est modifié. Branchement: iPhone sur un Mac, déverrouillé, toucher « Faire confiance »; Android avec le débogage USB activé, sur n'importe quel ordinateur. S'il trouve quelque chose (étape 5 de l'auteur): ne supprime rien d'abord, fais des captures du rapport complet tant que l'appli est encore sur le téléphone, change le mot de passe Apple ID ou Google depuis un autre appareil et active la double authentification; si c'est une question de sécurité et pas une simple appli indiscreète, montre le rapport à quelqu'un dont c'est le métier avant de retirer quoi que ce soit.

13.1 · Étape 1 : les applis installées

Liste toutes les applis installées sur ce téléphone avec leur identifiant et leur éditeur. Signale celles qui n'ont pas d'icône, celles installées en dehors de la boutique officielle (App Store / Play Store), et tout ce qui porte un nom de service système. Lecture seule, ne change rien.

13.2 · Étape 2 : ce qui a un contrôle profond

Liste chaque profil de configuration/MDM (iPhone) ou chaque appli administrateur de l'appareil et chaque appli ayant l'accessibilité activée (Android). Pour chacun : qui l'a installé, ce qu'il peut contrôler, quand il a été ajouté. Signale tout ce que je n'ai pas configuré moi-même.

13.3 · Étape 3 : suivre les données qui sortent

Montre chaque connexion sortante de ce téléphone. Signale toute destination qui n'est pas un service officiel connu : adresses IP brutes, ports inhabituels, ou tout ce qui envoie des données à intervalle fixe, par exemple toutes les deux ou trois minutes.

13.4 · Étape 4 : les autorisations et ce qui refuse de mourir

Liste les applis qui détiennent en permanence la localisation, le micro ou la caméra sans en avoir besoin, et tout ce qui se relance au démarrage ou tourne en continu en arrière-plan que je n'ai pas configuré.

13.5 · L'audit complet en un seul prompt

RÔLE : Tu es un auditeur de confidentialité et de sécurité mobile.

PÉRIMÈTRE : C'est mon propre téléphone (iPhone ou Android) et tu as un accès en lecture seule. Ne modifie, n'installe et ne supprime rien. Si une étape devait écrire sur le téléphone, saute-la et dis-le.

OBJECTIF : Dis-moi en langage simple si quelque chose sur ce téléphone me surveille (localisation, messages, micro, caméra) ou fait des rapports à un tiers.

VÉRIFIE :

Applis : chaque appli installée avec son identifiant et son éditeur. Signale celles sans icône, installées en dehors de l'App Store / du Play Store, ou déguisées en service système.

Contrôle profond : tous les profils de configuration/MDM (iPhone) et les applis administrateur de l'appareil / d'accessibilité (Android) : qui les a installés, ce qu'ils contrôlent, quand.

Autorisations : toute appli qui détient en permanence la localisation, le micro ou la caméra sans en avoir besoin.

Persistance : tout ce qui se relance au démarrage ou tourne en continu en arrière-plan que je n'ai pas configuré.

Réseau : les connexions sortantes. Signale les destinations qui ne sont pas des services connus : IP brutes, ports inhabituels, rapports à intervalle fixe.

Pour chaque signalement : ce que c'est, ce à quoi il a accès, depuis combien de temps il est là, où il envoie des données, et une gravité faible/moyenne/élevée.

TERMINE PAR : une ligne de verdict (sain ou compromis), puis trois listes (Sûr / À supprimer / Demande un professionnel), puis la chose la plus importante à faire ensuite. Lecture seule.

14 Avant la mise en ligne : 14 attaques sur son propre site, puis 20 causes de lenteur

Vox (@Voxyz_ai), 3 396 j'aime, 16/10/2026 · X · 2 prompts

L'auteur dit d'envoyer le post entier à Claude Code, puis d'ajouter au besoin une ligne finale; cette ligne est recollée à la fin de chaque bloc. Pour le premier, elle porte les garde-fous (ton site seulement, comptes et données de test, jamais la base de production): elle reste collée au prompt. Le second bloc vient du post cité du même auteur.

14.1 · Jouer le méchant : 14 attaques sur ton propre site, preuves à l'appui

Astuce Claude Code : avant de mettre en ligne un site fait en vibe coding, fais jouer le méchant à Opus 5.5. Envoie 3 sous-agents avec trois comptes de test (A, B et un admin) exécuter réellement ces 14 attaques, collecter des preuves, puis tout corriger en une passe :

Se faire passer pour d'autres utilisateurs

- Sans te connecter, prends la clé publique dans ton frontend et interroge directement chaque table de la base de données. Regarde ce qui revient
- Connecté en tant que A, remplace l'id dans une requête par celui de B et essaie de lire, modifier et supprimer les données de B
- Quand A modifie son profil, ajoute role: admin à la requête, puis essaie de modifier le rôle directement dans les données de son propre compte
- Utilise un lien de réinitialisation de mot de passe une fois, puis clique dessus à nouveau

Contourner les limites de l'interface

- Pour les boutons désactivés et les options masquées, saute la page et envoie la requête directement
- Une fois les utilisations gratuites épuisées, vérifie si l'appel direct à l'API fonctionne encore
- Avec seulement un peu de crédit restant, envoie 20 requêtes d'un coup et regarde si plus de requêtes réussissent que le crédit ne le permet
- Remplis les formulaires avec du texte très long, des caractères spéciaux, des nombres négatifs et des valeurs vides
- Change la redirection après connexion (?next=) vers un autre site, et essaie aussi des valeurs qui commencent par // et @

Entrer par l'extérieur

- Mets `<img src=x onerror=alert(1)>` dans un nom d'utilisateur, un commentaire ou du Markdown, puis ouvre la page dans un navigateur et regarde si une alerte s'affiche
- Si l'app récupère des URL (import de liens, génération d'aperçus), donne-lui `http://localhost` et `http://169.254.169.254` et vérifie dans les journaux du serveur qu'une requête sortante a réellement eu lieu. Sauter ce point s'il n'y a pas de fonctionnalité de ce genre
- Pour les pages et les réponses d'API que A a vues, demande la même adresse depuis un autre navigateur en tant que B ou déconnecté, et regarde si le contenu de A apparaît
- Ouvre les liens de prévisualisation de ta plateforme d'hébergement sans être connecté à la plateforme
- Cache dans un document téléversé ou une page web une instruction qui demande à ta fonctionnalité d'IA de lire les données d'un autre utilisateur ou d'appeler un outil qu'elle ne devrait pas appeler, et regarde si elle le fait

Collecter les preuves

- Chaque point doit réellement être exécuté. S'il a seulement été lu dans le code sans être exécuté, il compte comme « non prouvé »
- Après un correctif, désactive temporairement la protection dans le code local et relance. L'attaque doit réussir. Réactive-la : elle doit être refusée
- Mets les résultats dans un tableau : comment chaque attaque a été menée, la requête et la réponse, et si elle a été refusée quand elle devait l'être
- Enregistre les tests sous forme de scripts dans le dépôt et relance-les avant ta prochaine mise en ligne
- Quand tu as fini, installe le plugin officiel security-guidance d'Anthropic (/plugin install security-guidance@claude-plugins-official). Après chaque tour de modifications, et après chaque commit ou push qu'il fait, Opus relit à nouveau les changements. Ces relectures consomment de l'usage supplémentaire

L'effort moyen par défaut suffit. Le groupe usurpation est le plus coûteux à rater, alors monte celui-là en high.

D'abord, vérifie que l'environnement de prévisualisation n'est pas connecté à la base de données de production. S'il l'est, exécute tout uniquement en local. Utilise des comptes et des données de test, avec un jeu de comptes distinct pour chaque sous-agent. Ne touche pas aux vrais utilisateurs, n'attaque pas de services qui ne t'appartiennent pas, ne martèle aucun point d'accès, et ne teste les paiements qu'en mode test. Ne désactive les protections que temporairement dans le code local, et ne modifie jamais

les réglages de la base de données ni du tableau de bord. Garde les mots de passe des comptes de test dans .env, hors de git, et masque les jetons dans le tableau de résultats. Envoie un sous-agent par groupe pour attaquer, en consignait sans corriger. Lance le sous-agent usurpation en effort high. Ensuite, corrige tout toi-même à partir de leurs listes, relance chaque test et donne-moi un tableau avant/après. Supprime les comptes et les données de test à la fin. Pour installer le plugin ou modifier les réglages du tableau de bord, donne-moi les étapes et je le ferai.

14.2 · Les 20 causes de lenteur, vérifiées par 4 sous-agents

Astuce Claude Code : avant de mettre en ligne un site fait en vibe coding, fais envoyer par Opus 5.5 4 sous-agents vérifier ces 20 choses qui le ralentissent, puis tout corriger en une passe :

Images et polices

- Convertis chaque image en WebP ou AVIF, dimensionnée à sa taille réelle d'affichage sur la page
- Sers les grandes images depuis Supabase Storage via next/image à la bonne taille. Avec l'offre Pro, les transformations d'images propres à Supabase fonctionnent aussi
- Charge en différé les images et vidéos sous la ligne de flottaison, et charge en premier la grande image du haut de page
- Charge les polices comme le prévoit ton framework (next/font dans Next.js), et limite-toi à une ou deux polices
- Retire les scripts tiers que tu n'utilises pas, et lance l'analytics et les widgets de chat après le chargement de la page

Code frontend

- Lance un analyseur de bundle pour trouver les plus gros paquets, et là où une bibliothèque entière est importée, n'importe que les parties que tu utilises, comme des fonctions lodash isolées
- Charge les composants lourds dont le premier écran n'a pas besoin, comme les graphiques, les éditeurs et les cartes, avec un import dynamique au moment où ils sont nécessaires
- Récupère les données de la page d'accueil côté serveur, pour que la page s'ouvre avec son contenu déjà présent
- Diffuse les réponses de l'IA au fil de leur génération
- Affiche des squelettes pendant le chargement des données

Base de données et API

- Indexe les colonnes sur lesquelles tu filtres et tries, y compris celles qu'utilisent tes politiques RLS, comme user_id
- Dans les politiques RLS, écris auth.uid() sous la forme (select auth.uid()) pour qu'il ne s'exécute qu'une fois par requête
- Trouve les requêtes N+1 et transforme-les en une seule requête
- Pagine les longues listes
- Mets en cache les réponses d'API identiques pour tout le monde, et ne mets jamais en cache ce qui est lié à un utilisateur connecté
- Déplace les travaux lourds comme l'envoi de mails, le traitement d'images et la génération de rapports vers des tâches d'arrière-plan

Serveurs et réseau

- Place tes fonctions serveur dans la même région que ta base de données. Les nouveaux projets Vercel exécutent les fonctions dans la région US East par défaut
- Transforme les pages dont le contenu change rarement en pages statiques ou en ISR
- Quand une page appelle des API l'une après l'autre, exécute-les en parallèle ou fusionne-les en une seule
- Dans Chrome, émule un téléphone Android d'entrée de gamme en 4G lente et parcours les principaux enchaînements

L'effort moyen par défaut suffit. Le groupe base de données doit lire les requêtes et décider quels index ajouter, alors monte celui-là en high.

Envoie un sous-agent par groupe vérifier sans rien modifier, et note dans quel fichier se trouve chaque problème et quelle page il touche. Lance le sous-agent base de données en effort high. Avant de modifier quoi que ce soit, lance un build de production (next build && next start dans Next.js) ou utilise un lien de prévisualisation, puis lance Lighthouse trois fois en mode mobile sur la page d'accueil et sur les deux ou trois pages les plus lentes, et note la médiane. Ensuite, corrige tout toi-même à partir de leurs listes, mesure de nouveau de la même façon, et donne-moi les chiffres avant/après. Ne supprime pas de fonctionnalités pour gagner en vitesse. Demande-moi avant de supprimer des dépendances, de changer la région de déploiement ou de modifier la base de données.

15 Auditer une appli Android ou un contrat de jeton avec REA (la seule part défensive du fil)

codila (@0xCodila), 5 263 j'aime, 17/10/2026 · X · 3 prompts

Ressource : <https://x.com/0xCodila/status/2109623431763292649>

Le reste du post et de l'article (rétro-ingénierie « sans ban ni refus », reprise du code extrait présenté à Opus comme son propre projet, clonage de produits payants) relève du contournement et n'est pas servi: voir la table des annonces. Garde-fou cité par l'auteur: REA sert à la recherche légale, l'enseignement, l'interopérabilité, la préservation et aux logiciels qu'on possède ou qu'on a le droit d'examiner.

Selon l'auteur: les noms de fonctions sont devinés depuis le bytecode, un signal d'alerte veut dire « creuser », et la commande ne tourne pour l'instant que sous Linux x64.

15.1 · Installer REA dans l'agent de code

Installe REA et connecte-le à cet agent de code avec `npx rea-agents@latest setup`. Montre-moi le plan d'installation pour validation, puis vérifie l'installation.

15.2 · Ce que fait vraiment une appli de ton téléphone (APK)

Utilise REA sur cet APK. Liste chaque permission, chaque SDK d'analyse ou de pistage, et chaque domaine avec lequel l'appli communique. Indique la classe et la méthode pour chaque constat.

15.3 · Ce jeton est-il une arnaque ? (bytecode du contrat)

Utilise REA sur ce bytecode de contrat. Liste chaque fonction capable de modifier l'état. Signale tout ce qui ressemble à mint, blacklist, pause, changement de frais ou transfert réservé au propriétaire. Indique ce qui est confirmé et ce qui est déduit.

Les annonces de prompt sans le prompt

Ces 20 publications promettaient un prompt absent de tout ce qu'elles servent. Je les ai ouvertes et j'ai cherché. C'est noté ici pour que personne n'y retourne.

Compte	Ce qui manque
Akshay (@akshay_pachaar) le post	Présentation de Harness ² (Google) et, dans l'article cité sponsorisé par Comet ML, d'Opik: du code Python et des commandes d'installation, aucun prompt.
Sammy (@sammy_capital) le post	Série REA: liste de 100 cibles de rétro-ingénierie (.exe, APK, IPA, firmware, jeux) et outils associés. Contournement, non servi.
Gyome1_ (@Gyome1_), article cité par @sammy_capital le post	Série REA: quatre fiches d'agents (analyste, rédacteur de spec, reconstructeur, vérificateur) et un script pour reconstruire en « salle blanche » un logiciel à partir de son binaire. La chaîne sert à cloner un programme fermé; les garde-fous de l'auteur (logiciel à soi ou autorisé, licence à lire) ne détachent pas une méthode défensive servable seule. Contournement, non servi.

Compte	Ce qui manque
thewizeai le post	Onze skills « /replica-* » pour cloner une appli de bout en bout, la renommer et la revendre; kit réservé au DM (« CLONE »). Contournement, non servi.
codiebro le post	Reel sur Open Higgsfield (vidéo « illimitée » avec une clé OpenRouter): aucun prompt dans la légende.
硅基人老王 (@AI_oldwang) le post	Post en chinois: feuille de route « 30 jours pour se lancer », attribuée à un ancien PDG de Google, en quatre phases (architecture d'agent, prompts et contexte, MCP et skills, guides communautaires). 13 liens vers des cours et docs gratuits (Skilljar, docs Claude Code mémoire, MCP, skills, tutoriel interactif d'Anthropic, guides GitHub), aucun prompt.
codila (@0xCodila) le post	Reste du cours REA (« obtenir n'importe quoi gratuitement »: installer REA, s'exercer sur l'app Notes, reconstruire la recherche de Notes ou l'indicateur de saisie de Discord, recréer un jeu depuis son exécutable, cloner un site, reconstruire une fonction payante, comparer deux versions d'une app): contournement, non servi. Les deux audits défensifs sont servis au n° 2.
Gyomei (@Gyome1_) le post	Index de 12 877 projets de rétro-ingénierie (« 12 000 produits fermés à démonter ») et X Article « équipe de rétro-ingénierie en 7 étapes »: 4 sous-agents (analyst, spec-writer, rebuild, verifier) et un script run-team.sh pour reconstruire en salle blanche un programme fermé. Contournement (clonage de produits fermés), non servi.
simplifyinai le post	« Qwen 3.8 uncensored »: poids modifiés pour réduire les refus et pack de prompts système, lien envoyé en DM. Contournement des restrictions du modèle, non servi (et absent de la légende).
simplifyinai le post	Dépôt MuseAI-Skills (68 skills, listes de permissions, scénarios d'évaluation), lien envoyé en DM; aucun prompt dans la légende.
nedz_reclassified le post	Modèle texte-vers-image local présenté « sans refus », envoyé en DM; aucun prompt dans la légende.
godofprompt (@godofprompt) le post	Le « mégaprompt » qui configure un agent ChatGPT Dots en t'interviewant une question à la fois (goulots, automatisations, dépendances à toi, mémoire, interdiction d'inventer) est dans les diapos (diapo 3 visée), non lisibles ici: pas d'OCR. Guide complet contre le commentaire « megaprompt ». Source citée: @EXM7777.
bencodezero (@bencodezero) le post	Mods de Claude Code: « les prompts à copier » sont dans le guide envoyé en DM (commentaire « MODS »); la légende n'en donne aucun.
codila (@0xCodila) le post	Prompt principal (reprendre dans Opus 5.5 le code d'une appli extrait par REA en le présentant comme « mon projet que j'avais arrêté ») et pack de l'article (cloner un site, reconstruire un jeu depuis son exécutable, recréer une fonction payante, reproduire la fonction d'une appli tierce): rétro-ingénierie pour « tout avoir gratuitement » et sans refus du modèle, contournement, non servi. Les trois prompts défensifs sont servis au n°3.
artificial.corner (@artificial.corner) le post	Le branchement de Claude sur NotebookLM (rapports, tableaux de bord, petites applis depuis ses carnets): « la mise en place est dans le guide », envoyé en DM (commentaire « GEMINI »).
Ruben Hassid (@rubenhassid) le post	Le prompt est pré-rempli par l'outil aicertified.free et n'est publié ni dans le post ni dans l'article cité. Son objet (faire passer les QCM de certification par Codex à la place de la personne) le rendrait de toute façon non servi. Aucun prompt d'apprentissage.

Compte	Ce qui manque
Sammy (@sammy_capital) le post	Série REA: feuille de route de liens, plus l'article cité de @Gyome1_ avec quatre fichiers d'agents (analyste, rédacteur de spécification, reconstruteur, vérificateur) et un script qui reconstruisent un logiciel fermé à partir de son binaire. Contournement (clonage de produits fermés, licences), non servi. Aucune méthode défensive séparable: tout le dispositif vise la reconstruction.
bestapps.ai (@bestapps.ai) le post	Kit d'installation de l'agent DeepSeek sur Ollama réservé au DM (commenter « DEEP »); la légende ne contient aucun prompt.
bencodezero (@bencodezero) le post	Guide d'installation de Ponytail v5 réservé au DM (commenter « PONYTAIL »); la légende ne contient aucun prompt.
rvaniaaa (@rvaniaaaa) le post	Les quatre prompts du second cerveau qui compile sont déjà au catalogue n°9 (#11985), non répétés ici.

Trois outils qui remplacent un paquet de ces prompts

Sortis de la même veille. Une fois installés, ils font le travail que plusieurs prompts ci-dessus refont à la main à chaque conversation.

- **TasteSkill v2** ([Leonxlnx/taste-skill](#), MIT): direction artistique, vraies images créditées et vérification dans le navigateur, là où les prompts de « bon goût » s'en remettent au modèle.
- **motion-graphics-skills** et **claude-motion** (cités par @N01ennn): les règles de studio des prompts 1 et 2 déjà empaquetées en skills.
- **Cloudflare security-audit-skill** et **NVIDIA SkillSpector** (cités par @marklaunches): l'audit de son propre code et de ses skills, sans réécrire la liste d'attaques à la main.