JulienWeb.fr

STUDIO INTERNET · CODE · DESIGN · FORMATION

Synthèse

Les meilleurs prompts des Gueznet Prompts n°7 à n°10 (août à octobre 2026)

EXPÉDITEUR

Julien GUÉZENNEC

JulienWeb.fr — Entrepreneur Individuel
4 rue Lépine — 93500 PANTIN
+33 6 61 75 64 98 · contact@julienweb.fr

SIREN 510 600 661 · SIRET 51060066100035 · APE 6201Z — Prestataire référencé Cybermalveillance.gouv.fr

DESTINATAIRE

Lecteurs de julienweb.fr

Document libre, sans inscription
En téléchargement sur julienweb.fr

Pantin, le 5 novembre 2026

Référence : Dossier **Gueznet IA** · julienweb.fr/gueznet

Objet : Mieux faire travailler Claude en dépensant moins de tokens

Cette synthèse reprend les quatre derniers numéros « Prompts » de la Gueznet IA (n°7 à n°10, d'août à fin octobre 2026). Sur plusieurs centaines de prompts publiés, j'ai gardé les 14 qui servent vraiment au quotidien avec Claude, et surtout avec Claude Code. Chaque prompt se copie tel quel ; les crochets sont à remplir.

Trois idées reviennent d'un numéro à l'autre :

- **faire travailler un modèle moins cher**, dans un contexte court ;
- **donner une condition d'arrêt** à tout ce qui tourne en boucle ;
- **demander une preuve** (tests, mesures) avant d'accepter le mot « terminé ».

A. Dépenser moins de tokens

1. Envoyer le code vers un modèle moins cher

Une seule phrase, à poser dans le fichier d'instructions du projet (`CLAUDE.md`). D'après le n°7.

```
Pour toute tâche de code, utilise ton jugement pour choisir un modèle moins puissant adapté, et fais-le tourner dans un sous-agent.
```

Le modèle principal garde le jugement et la synthèse ; l'exécution part sur Sonnet ou Haiku.

2. Le modèle le plus bas qui suffit, et monter seulement sur preuve

D'après le n°9, adapté à la gamme Claude.

```
Classe la tâche avant d'agir : Haiku, Sonnet ou Opus, effort bas à élevé. Prends toujours la voie la plus basse qui suffit.
N'escalade que si le même test échoue deux fois, si le code touche la sécurité, ou si un doute d'architecture demeure. N'escalade pas parce qu'un modèle plus fort est disponible.
```

3. Haiku explore, Sonnet exécute, Opus conseille

D'après le n°10. Haiku parcourt les fichiers et la documentation et ne renvoie que des résumés ; Sonnet fait le travail ; Opus n'intervient que sur un échec répété. Le contexte de la conversation principale reste court, donc chaque tour coûte moins.

4. Un plan écrit par le modèle fort, exécuté par le modèle moins cher

D'après le n°7.

```
Construis un plan qu'un modèle moins cher peut exécuter seul. Pose-moi d'abord toute
question qui changerait la conception, puis écris un seul fichier de spécification :
OBJECTIF : ce qu'on construit et pourquoi
RÉUSSITE : une liste de contrôle qu'un modèle peut vérifier
JALONS : les phases, dans l'ordre
TÂCHES : petites étapes numérotées, avec les fichiers touchés et le critère de fin
PASSATION : conventions, pièges, ordre de travail
Chaque tâche assez petite pour que Sonnet la termine sans toi.
```

5. Faire le ménage dans ses instructions

D'après le n°8. C'est le prompt qui fait gagner le plus, parce que le fichier d'instructions et les skills sont relus à chaque tour. On demande un rapport, sans réécriture, sur six défauts :

- descriptions de skills trop longues ou trop vagues ;
- un gros fichier qui charge tout d'un coup, au lieu d'un fichier court qui renvoie vers des documents lus au besoin ;
- recettes trop rigides, étape par étape ;
- obligation de « tout lire avant de modifier », même pour une petite correction ;
- contrôles que le modèle fait déjà de lui-même ;
- permissions trop prudentes (« demande avant de faire X »).

Pour chaque point signalé : pourquoi, et quoi faire. Phrase de fin : « Ne réécris rien pour l'instant, fais-moi un rapport. »

6. Une condition d'arrêt à chaque boucle

D'après le n°10.

```
/goal [métrique chiffrée à atteindre]. Chaque tour doit améliorer une métrique ; abandonne
après deux tours sans amélioration. Arrête-toi après 10 tours.
```

Sans plafond, un agent qui n'y arrive pas tourne et consomme jusqu'à épuisement.

7. Couper la prose

D'après le n°8, reprise du guide d'Anthropic.

```
Retire toute prose maniérée. Sois direct. Pas de préambule.
```

8. Brider le format de sortie

D'après le n°10. Pour une réponse destinée à une machine : JSON strict, champ « raison » limité à 12 mots, et « en cas d'hésitation entre deux voies, choisis la moins chère ».

B. Obtenir un meilleur travail

9. Pas de preuve, pas de « terminé »

D'après les n°9 et n°10.

```
Avant de déclarer terminé : reformule l'objectif, liste chaque livrable, montre où il se trouve, montre ce qui a réellement été vérifié (tests, compilateur, linter, git diff : préfère une preuve automatique au jugement d'une IA), liste chaque lacune connue. Ne cache jamais une vérification qui a échoué.
```

Pour une page web, le n°10 ajoute : ouvrir la page, faire une capture avant et après, lire la console.

10. Des relecteurs qui ne peuvent que lire

D'après le n°10.

- Les sous-agents relecteurs n'ont que les outils de lecture : ils signalent, ils ne modifient rien.
- L'effort élevé est réservé au groupe de contrôles où il compte.
- Pour la relecture suivante, on relance le même relecteur, qui revérifie d'abord ce qu'il avait signalé, au lieu d'en créer un nouveau.
- On s'arrête quand tous donnent au moins 7 sur 10, ou après 3 tours.

11. Diagnostiquer avant de corriger

D'après le n°8.

```
Ne le corrige pas immédiatement. Diagnostique-le d'abord : symptômes, causes profondes, impact, priorité. Crée ensuite une version améliorée et montre les 3 changements les plus importants, avec la raison de chacun.
```

12. Étiqueter le degré de certitude

D'après les n°8 et n°10. On demande d'étiqueter chaque affirmation `[Certain]`, `[Probable]` ou `[Supposition]`, ou plus simplement : « indique ce qui est confirmé et ce qui est déduit ». Pour une réponse tirée de documents : « cite la phrase exacte sur laquelle tu t'appuies ; si la réponse n'y est pas, réponds “Absent de ces documents.” »

13. Le prémortem

D'après le n°7. Demander « mon plan est-il bon ? » pousse le modèle à dire oui. On inverse la question :

```
On est dans 6 mois et ce plan a complètement échoué. Écris l'autopsie : les 7 causes les plus probables, classées, chacune reliée à un détail que je t'ai donné. Pas de conseil générique. Ne me rassure jamais.
```

14. Tenir un journal des erreurs

D'après le n°8. Une ligne dans le fichier d'instructions :

```
Consigne les erreurs dans MISTAKES.md : ce qui s'est passé, la cause racine, comment l'éviter la prochaine fois.
```

Une erreur qui revient devient une règle du fichier d'instructions. Le journal doit rester court, sinon il pèse lui aussi sur chaque tour.

Ce qui n'est pas vérifié

Le n°10 cite un réglage de cache d'une heure par sous-agent (`cacheTtl`) et une commande `/claude-api prompt-audit`. Ce sont des affirmations des auteurs repris dans l'article ; je ne les ai pas testées. Les chiffres de gain annoncés par certains auteurs (« +300 % », « de 60 % à 94 % ») ne sont pas vérifiés non plus.

Les articles complets sont en ligne sur julienweb.fr, rubrique Gueznet.

Bien à vous,

Julien GUÉZENNEC

Dirigeant — EI JulienWeb.fr

Document libre, tiré de la Gueznet IA. Article complet : julienweb.fr/intelligence-artificielle/economiser-tokens-claude-code/12688/

Julien Guézennec

Studio Internet & Conception digitale 360° depuis 1998 ツ

SERVEUR, E-COMMERCE, SEO, UX, AD, FORMATION, CYBERSÉCURITÉ, IA, ...

+33 6 61 75 64 98 · contact@julienweb.fr · Pantin (93) · Julienweb.fr

Référencé **Cyber.gouv.fr** | Activateur **France Num** | Laissez-moi **un avis** ★★★★★