Chapitre G — Performance & Benchmark

2 skills regroupés sous ce thème.

- benchmark-models
- benchmark

benchmark-models

benchmark-models

> *Catalogue généré le 2026-05-11*

En une phrase

Compare plusieurs modèles d'IA (Claude, GPT, Gemini) sur la même tâche pour voir lequel est le plus rapide, le moins cher et le plus performant, avec des chiffres au lieu d'impressions.

Quand l'utiliser

- Tu veux savoir quel modèle utiliser pour un type de tâche précis (Claude vs GPT vs Gemini).
- Tu suspectes qu'un de tes skills gstack tournerait mieux avec un autre modèle.
- Tu veux mesurer l'impact d'une nouvelle version de modèle sur tes coûts et tes temps de réponse.
- Tu hésites entre plusieurs modèles pour une nouvelle automatisation et tu veux des données.
- Tu veux établir une référence pour détecter les régressions de qualité quand les modèles évoluent.

Comment l'invoquer

- **Slash command** : `/benchmark-models`
- **Voice triggers** : « compare models » · « model shootout » · « which model is best »
- **Phrases déclencheurs (texte)** : "benchmark models", "compare models", "which model is best for X", "cross-model comparison", "model shootout"
- **Auto-invocation** : Sur demande explicite.

Description détaillée

Le skill `benchmark-models` est un comparatif côte à côte. Tu lui donnes un prompt (ou un skill gstack), il l'envoie au même moment à Claude (via Claude Code), GPT (via le CLI Codex d'OpenAI), et Gemini, puis te ressort un tableau comparatif : temps de réponse, nombre de tokens consommés, coût, et optionnellement une note de qualité décidée par un juge LLM (un autre modèle qui note les sorties sur 10).

À ne pas confondre avec `/benchmark` qui mesure la performance d'une page web (Core Web Vitals). Celui-ci compare des modèles d'IA. Le workflow est interactif : Claude te demande d'abord quel prompt utiliser (un skill gstack, un prompt en ligne, ou un fichier sur disque), puis quels providers inclure. Une commande "dry-run" préalable te montre lesquels sont authentifiés sur ta machine — pas de surprise au moment de payer les appels API.

Le juge qualité ajoute environ 0,05 $ par run. Tu peux le désactiver si tu veux juste comparer vitesse et coût. À la fin du test, Claude résume : le plus rapide, le moins cher, le plus qualitatif (si le juge a tourné), et te propose de sauvegarder les résultats en JSON dans `~/.gstack/benchmarks/` pour pouvoir comparer plus tard. Chaque ligne de tableau montre le coût réel — tu sais ce que tu dépenses avant le prochain run.

Source

- **Plugin** : `gstack`
- **Nom interne** : `benchmark-models`
- **Fichier** : `/home/thymon/.claude/skills/gstack/benchmark-models/SKILL.md`

benchmark

benchmark

> *Catalogue généré le 2026-05-11*

En une phrase

Mesure les performances d'une page web (temps de chargement, Core Web Vitals, taille des fichiers) pour détecter quand ta nouvelle PR fait ramer le site sans que personne s'en aperçoive.

Quand l'utiliser

- Tu veux établir une mesure de référence (baseline) avant de modifier quelque chose qui peut impacter la perf.
- Tu veux comparer la perf actuelle d'une page à sa baseline et voir les régressions.
- Tu veux savoir quelles sont les ressources les plus lentes à charger sur une page.
- Tu veux vérifier que ton bundle JavaScript ne dépasse pas un budget perf (par exemple 500 KB).
- Tu veux suivre les tendances perf sur plusieurs semaines pour repérer les dérives lentes.

Comment l'invoquer

- **Slash command** : `/benchmark <url>` (options : `--baseline`, `--quick`, `--pages <liste>`, `--diff`, `--trend`)
- **Voice triggers** : « speed test » · « check performance »
- **Phrases déclencheurs (texte)** : "performance", "benchmark", "page speed", "lighthouse", "web vitals", "bundle size", "load time"
- **Auto-invocation** : Sur demande explicite.

Description détaillée

Le skill `benchmark` joue le rôle d'un ingénieur perf qui sait que les sites ne deviennent pas lents d'un seul coup : ils meurent à petit feu, 50 ms ici, 20 KB là, et un jour la page met 8 secondes à charger et personne ne sait quand ça a commencé. Sa mission : mesurer, enregistrer une baseline, comparer, alerter dès qu'une régression apparaît.

Il s'appuie sur le démon de navigateur de gstack (browse) et utilise les APIs Performance du navigateur pour collecter des mesures précises : TTFB (temps avant le premier octet), FCP (First Contentful Paint), LCP (Largest Contentful Paint), DOM Interactive, DOM Complete, Full Load, plus l'analyse de chaque ressource (taille, durée, type). Il distingue les bundles JS et CSS, compte les requêtes, identifie les ressources tierces lentes (analytics, fonts), et applique des seuils standards : plus de 50 % d'augmentation ou plus de 500 ms en absolu = REGRESSION, plus de 20 % = WARNING.

Quatre modes principaux. `--baseline` capture la mesure de référence (à lancer avant tes changements). Mode comparaison (par défaut) compare la perf actuelle à la baseline. `--quick` fait juste un check rapide sans baseline. `--trend` affiche l'historique des mesures pour repérer les tendances (par exemple : "LCP a doublé en 8 jours"). `--diff` ne benchmark que les pages affectées par ta branche actuelle. À la fin, il te donne aussi un check de budget perf (FCP < 1,8 s, LCP < 2,5 s, total JS < 500 KB, total CSS < 100 KB, etc.) avec une note A/B/C, et recommande des optimisations concrètes (code splitting, lazy loading, etc.).

Source

- **Plugin** : `gstack`
- **Nom interne** : `benchmark`
- **Fichier** : `/home/thymon/.claude/skills/gstack/benchmark/SKILL.md`