AI TOOLS WIKI · MÉMO

Skill Creator *mémo*

L'essentiel pour utiliser **Skill Creator**, la skill officielle d'Anthropic qui aide à créer, tester et améliorer d'autres skills pour Claude Code. Elle se pilote en langage naturel ; les commandes ci-dessous sont les scripts qu'elle utilise.

Famille Revue, sécurité et création de skills · **Éditeur** Anthropic · **Licence** Apache-2.0
Note du wiki 11/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/revue-et-securite/skill-creator

Installation

```
/plugin install skill-creator@claude-plugins-official
```

Comment la déclencher

Il suffit de décrire le besoin :

- « Crée une skill qui … » / « Transforme ce workflow en skill »
- « Améliore la skill X » / « Lance des évals sur ma skill »
- « Optimise la description de ma skill pour qu'elle se déclenche mieux »

Tu peux entrer à n'importe quelle étape : si tu as déjà un brouillon, elle passe directement aux tests.

Le cycle en 5 étapes

Étape	Ce qui se passe
1. Cadrage	Questions sur l'objectif, les phrases qui doivent déclencher la skill, le format de sortie, les cas limites.
2. Rédaction	Écriture du **`SKILL.md`** : **`name`**, **`description`**, instructions, ressources (**`scripts/`**, **`references/`**, **`assets/`**).
3. Tests	2 ou 3 prompts réalistes dans **`evals/evals.json`**, exécutés en parallèle **avec** et **sans** la skill (ou contre l'ancienne version).
4. Revue	Un viewer HTML montre les sorties (onglet *Outputs*) et les chiffres (onglet *Benchmark* : réussite, temps, tokens). Tu laisses tes commentaires, sauvegardés dans **`feedback.json`**.
5. Itération	La skill est réécrite selon tes retours, puis on relance les tests.

Scripts utiles (depuis le dossier de skill-creator)

`python -m scripts.aggregate_benchmark <workspace>/iteration-N --skill-name <nom>`	Agrège les résultats en **`benchmark.json`** et **`benchmark.md`**.
`python eval-viewer/generate_review.py <workspace>/iteration-N --skill-name <nom> --benchmark …/benchmark.json`	Ouvre le viewer de revue. Ajouter `--previous-workspace` dès la 2e itération, ou `--static <fichier>` sans navigateur.
`python -m scripts.run_loop --eval-set <evals.json> --skill-path <skill> --model <modèle> --max-iterations 5 --verbose`	Optimise automatiquement la description pour mieux déclencher la skill.
`python -m scripts.package_skill <dossier-skill>`	Empaquette la skill en fichier **`.skill`** installable.

Optimiser le déclenchement

1. Générer **20 requêtes de test** : 8 à 10 qui doivent déclencher la skill, 8 à 10 qui ne doivent pas (de préférence des « presque » : même vocabulaire, besoin différent).
2. Les relire et les corriger dans le template HTML fourni.
3. Lancer **`run_loop`** : il sépare 60 % d'entraînement et 40 % de test, lance chaque requête 3 fois, et propose de meilleures descriptions sur 5 itérations maximum.
4. Reprendre la **`best_description`** (choisie sur le jeu de test, pas d'entraînement) dans le **`SKILL.md`**.

Conseils d'écriture de skill

- **Description un peu « insistante »** : Claude a tendance à ne pas assez utiliser les skills. Y lister clairement les situations où l'utiliser, même quand l'utilisateur ne les nomme pas.
- **`SKILL.md` sous 500 lignes** ; mettre le reste dans **`references/`** en indiquant quand le lire.
- **Expliquer le pourquoi** plutôt qu'empiler des « MUST » en majuscules : le modèle suit mieux une règle qu'il comprend.
- **Tester sur des prompts réalistes** (avec contexte, fautes de frappe, détails concrets), pas sur « Crée un graphique ».
- **Pas de tests chiffrés pour les skills subjectives** (style d'écriture, design) : la revue humaine suffit.
- **Repérer le travail répété** : si chaque test réécrit le même script, l'inclure dans **`scripts/`**.
- **Utiliser le même modèle que ta session** pour **`run_loop`**, sinon le déclenchement mesuré ne correspond pas à l'usage réel.

Sources (documentation officielle)

- SKILL.md de Skill Creator
- Dépôt claude-plugins-official
- Skill Creator – Smithery