Chapitre H — Outils & Méta

4 skills regroupés sous ce thème.

- make-pdf
- learn
- skillify
- gstack-upgrade

make-pdf

make-pdf

> *Catalogue généré le 2026-05-11*

En une phrase

Transforme n'importe quel fichier markdown en PDF de qualité publication, avec marges propres, numéros de page, table des matières cliquable et même un filigrane DRAFT si besoin.

Quand l'utiliser

- Tu veux convertir une note, un essai, une lettre ou un rapport en PDF présentable.
- Tu veux partager un document avec quelqu'un qui ne lit pas de markdown.
- Tu prépares un brouillon et tu veux un filigrane "DRAFT" en diagonale pour bien signaler que ce n'est pas final.
- Tu produis une vraie publication (essai, mémoire) avec page de garde, table des matières et coupures de chapitre automatiques.
- Tu veux itérer rapidement sur le rendu visuel d'un document long.

Comment l'invoquer

- **Slash command** : `/make-pdf`
- **Voice triggers** : « make this a pdf » · « make it a pdf » · « export to pdf » · « turn this into a pdf » · « turn this markdown into a pdf » · « generate a pdf » · « make a pdf from » · « pdf this markdown »
- **Phrases déclencheurs (texte)** : "make a PDF", "export to PDF", "turn this markdown into a PDF", "generate a document"

- **Auto-invocation** : □ Oui — dès que tu as un fichier `.md` ouvert et que tu demandes un PDF, Claude propose ce skill.

Description détaillée

Le skill `make-pdf` repose sur un petit binaire compilé qui produit des PDF avec une mise en page soignée : marges de 1 pouce, Helvetica partout, guillemets typographiques, tirets cadratins corrects, et un texte qui se copie-colle proprement (pas de "S a i l i n g" à la lettre près comme avec certains outils). C'est l'équivalent d'avoir un mini-Faber & Faber pour tes notes markdown.

Trois modes principaux. Le mode rapide (`generate fichier.md`) sort un PDF propre avec en-tête, numéro de page et mention CONFIDENTIAL en pied. Le mode publication (`--cover --toc`) ajoute une page de garde avec ton nom et la date, une table des matières cliquable, et coupe automatiquement chaque chapitre H1 sur une nouvelle page. Le mode brouillon (`--watermark DRAFT`) ajoute un filigrane diagonal sur chaque page pour bien signaler que ce n'est pas final.

Il y a aussi un mode prévisualisation (`preview fichier.md`) qui ouvre le rendu HTML dans ton navigateur, parfait pour itérer sans repasser par le PDF à chaque modification. Les options principales couvrent la taille de page (Letter, A4, Legal), les marges (en pouces, points, centimètres ou millimètres), le titre, l'auteur, et la possibilité de cacher le pied "CONFIDENTIAL". Sur Linux, il faut installer `fonts-liberation` pour que les polices s'affichent correctement.

Source

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

learn

learn

> *Catalogue généré le 2026-05-11*

En une phrase

Consulte, recherche et nettoie les "apprentissages" que gstack a accumulés sur ton projet au fil des sessions (patterns, pièges, préférences, choix d'archi).

Quand l'utiliser

- Tu te demandes "tiens, on n'avait pas déjà résolu ce bug ?" et tu veux fouiller la mémoire du projet.
- Tu veux voir ce que gstack a appris sur ton projet jusqu'ici.
- Tu veux nettoyer des notes obsolètes (fichiers supprimés, informations contradictoires).
- Tu veux exporter les apprentissages dans `CLAUDE.md` pour que ce soit toujours pris en compte.
- Tu veux ajouter manuellement une règle ou une préférence à mémoriser.

Comment l'invoquer

- **Slash command** : `/learn` (sous-commandes : `search`, `prune`, `export`, `stats`, `add`)
- **Phrases déclencheurs (texte)** : "show learnings", "what have we learned", "manage project learnings", "didn't we fix this before?"
- **Auto-invocation** : □ Oui — quand tu demandes "est-ce qu'on n'a pas déjà vu ça ?" ou que tu cherches un pattern passé.

Description détaillée

Le skill `learn` est ton wiki personnel pour chaque projet. Pendant que tu utilises gstack (avec `/review`, `/ship`, `/investigate`, etc.), Claude enregistre automatiquement les choses qu'il observe : un pattern qui revient, un piège à éviter, une préférence que tu exprimes, un choix d'architecture. Tout ça s'accumule dans un fichier `learnings.jsonl` propre à ton projet.

Le skill propose six commandes pour gérer cette mémoire. `/learn` tout court te montre les 20 dernières entrées groupées par type. `/learn search <terme>` cherche dans la mémoire. `/learn prune` détecte les entrées obsolètes (fichiers référencés qui n'existent plus, contradictions entre deux notes) et te propose de les supprimer ou les corriger. `/learn export` formate les apprentissages en markdown propre que tu peux coller dans `CLAUDE.md` ou un fichier de doc. `/learn stats` te donne un récap par type et par source. `/learn add` te permet d'ajouter manuellement une règle.

Chaque entrée a un type (`pattern`, `pitfall`, `preference`, `architecture`, `tool`), une clé courte, une description en une phrase, et un score de confiance de 1 à 10. C'est la couche de mémoire long terme de gstack : ce qui survit aux sessions et aux compactions de conversation. Important : ce skill ne modifie jamais ton code, il gère uniquement les apprentissages.

Source

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

skillify

skillify

> *Catalogue généré le 2026-05-11*

En une phrase

Transforme le dernier scraping réussi (via `/scrape`) en un skill permanent et automatisé, pour que la même extraction reparte en 200 millisecondes au lieu de tout réapprendre.

Quand l'utiliser

- Tu viens d'utiliser `/scrape` pour extraire des données d'un site, ça marche, et tu veux le réutiliser plus tard.
- Tu veux automatiser une extraction récurrente (front page d'un site, listing, stats).
- Tu veux passer d'un scraping piloté pas-à-pas par un agent à un script déterministe rapide.
- Tu veux partager ton extraction avec d'autres projets (mode global) ou la garder pour un projet précis.

Comment l'invoquer

- **Slash command** : `/skillify`
- **Phrases déclencheurs (texte)** : "skillify", "codify this scrape", "save this scrape", "make this permanent", "codify"
- **Auto-invocation** : Sur demande explicite (typiquement après un `/scrape` réussi).

Description détaillée

Le skill `skillify` est le multiplicateur de productivité du système de scraping de gstack. La logique : `/scrape` est lent parce que Claude pilote le navigateur en direct, page par page, sélecteur par sélecteur, en essayant des options jusqu'à trouver. Une fois que ça marche, c'est dommage de tout refaire à chaque fois. `/skillify` prend le dernier scraping réussi de la conversation et le transforme en un script Playwright propre, testé, et réutilisable.

Le workflow est strict pour ne jamais livrer un skill cassé. Il commence par retrouver le dernier `/scrape` valide dans la conversation (max 10 tours en arrière). Il propose un nom de skill court, des phrases déclencheurs (3 à 5), et te demande où le ranger : niveau global (`~/.gstack/browser-skills/`, accessible à tous tes projets) ou niveau projet (uniquement ce dépôt). Il synthétise ensuite un `script.ts` (la logique d'extraction sous forme de fonction pure), un `script.test.ts` (un test qui rejoue contre une capture HTML enregistrée comme fixture), et capture cette fixture (HTML figé du site cible). Tout est écrit dans un dossier temporaire d'abord.

Le test tourne dans ce dossier temporaire. S'il passe ET que tu approuves explicitement, le skill est promu dans son emplacement définitif. Sinon, le dossier temporaire est supprimé entièrement : pas d'état "à moitié livré". Une fois en place, les prochaines fois que tu lances `/scrape` avec une intention similaire, ce script déterministe tourne en environ 200 millisecondes — sans agent, sans navigation manuelle. Limitations à savoir : un seul URL par skill, les fixtures peuvent devenir obsolètes si le site change, et ce skill ne gère pas les flows qui modifient le site (formulaires, clics destructifs) — c'est le job de `/automate`.

Source

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

gstack-upgrade

gstack-upgrade

> *Catalogue généré le 2026-05-11*

En une phrase

Met à jour gstack vers sa dernière version, détecte automatiquement comment il est installé (global, local, vendored) et te montre ce qui change.

Quand l'utiliser

- gstack t'a affiché un message `UPGRADE_AVAILABLE` au début d'une session et tu veux mettre à jour.
- Tu veux passer manuellement à la dernière version de gstack sans attendre un prompt.
- Tu as activé l'auto-upgrade et tu veux comprendre ce qui s'est passé après une mise à jour automatique.
- Tu veux configurer le comportement des mises à jour (auto ou pas, snoozer une version, désactiver les checks).
- Tu veux voir le CHANGELOG des versions sorties depuis ta dernière mise à jour.

Comment l'invoquer

- **Slash command** : `/gstack-upgrade`
- **Voice triggers** : « upgrade the tools » · « update the tools » · « gee stack upgrade » · « g stack upgrade »
- **Phrases déclencheurs (texte)** : "upgrade gstack", "update gstack", "get latest version", "update gstack version", "get latest gstack"

- **Auto-invocation** : □ Oui — quand le préambule d'un skill détecte `UPGRADE_AVAILABLE`, Claude propose ce skill.

Description détaillée

Le skill `gstack-upgrade` gère le cycle de mise à jour de gstack. Au début de chaque session, gstack vérifie en arrière-plan s'il existe une version plus récente. Quand c'est le cas, il te le signale dans le préambule d'un skill. À ce moment, tu as quatre choix : oui mettre à jour maintenant, oui et active l'auto-upgrade pour ne plus jamais y penser, pas maintenant (avec un snooze qui s'allonge à chaque report : 24h, puis 48h, puis 1 semaine), ou ne plus me demander.

Le skill détecte automatiquement comment gstack est installé sur ta machine. Cinq cas possibles : install global git (le plus courant, dans `~/.claude/skills/gstack/`), install local git (cloné dans le projet courant), vendored local (copié sans git dans `.claude/skills/gstack/`), vendored global, ou pas trouvé du tout. Pour les installs git, il fait un `git fetch + reset --hard origin/main + ./setup`. Pour les vendored, il clone une nouvelle copie, l'échange avec l'ancienne (gardée en `.bak` au cas où), puis relance le setup.

Si tu as des modifications locales non commitées dans le dossier gstack (rare mais possible pour les contributeurs), le skill les stash avant l'upgrade et te prévient pour que tu puisses les récupérer après. Auto-upgrade s'active via la config (`gstack-config set auto_upgrade true`) ou la variable d'environnement `GSTACK_AUTO_UPGRADE=1`. Pour désactiver complètement les vérifications de mises à jour, tu réponds "Never ask again" — ça met `update_check: false` dans la config. À la fin de l'upgrade, le skill te montre ce qui a changé depuis ta version précédente.

Source

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