PLAYBOOK GRATUIT · DÉCUPLER

Playbook Claude Code × SEO

Par Nathan Fenina, fondateur de Décupler · version du 05/10/2026 · plugin decupler-seo v3.7.1

Ce playbook est la version à emporter de https://decupler.com/claude-code-seo-os/. Il décrit le système avec lequel je fais du SEO et du GEO avec Claude Code : les skills, les agents, l'infra, les 17 procédures (SOP) que je suis, et la partie GEO. Tout est étape par étape, même sans être développeur.

Une règle traverse tout le document : Claude ne publie jamais un chiffre qu'il n'a pas trouvé dans une source. Quand une donnée manque, il écrit « non mesuré » ou « nécessite tel outil ».

Avant de commencer

Claude Code, c'est l'agent de Claude qui travaille dans un dossier : il lit et écrit des fichiers, lance des scripts, appelle tes outils par MCP, et peut tourner seul à heure fixe (les routines). Tu lui parles en français, comme à un collaborateur.

- Il faut un abonnement Claude **Pro, Max, Team ou Enterprise**. Le plan gratuit ne donne pas accès à Claude Code.
- Sans terminal : l'**application de bureau** Claude (macOS, Windows) fait tourner Claude Code.
- Avec un terminal : `curl -fsSL https://claude.ai/install.sh | bash` sur Mac et Linux, `irm https://claude.ai/install.ps1 | iex` sur Windows (PowerShell).
- Ensuite, installe le plugin et lance le diagnostic (partie 1).

La checklist complète, case par case, est dans `checklist-installation.md`.

Chat Claude ou Claude Code ?

	Chat Claude (claude.ai)	Claude Code
Données	ce que tu colles, ou un connecteur	Search Console, GA4, DataForSEO… par MCP ou par script, à chaque tâche
Fichiers	pièces jointes	tout un dossier : tes pages, ta mémoire, tes rapports, lus et écrits
Mémoire	projets et mémoire de conversation	`CLAUDE.md` et `memoire/`, relus à chaque session et versionnés
Méthode	un prompt, ou un skill importé	50 skills et 15 agents, lancés par une phrase ou une commande
Volume	une page à la fois	un lot : 459 URL inspectées en une semaine sur decupler.com
Publication	copier-coller	brouillon WordPress par l'API, sauvegarde avant
Régularité	quand tu y penses	une routine cloud chaque mercredi, ordinateur éteint

Par où commencer

- **Jour 1** : installer Claude Code et le plugin, `/seo doctor`, brancher Search Console, `/seo audit` sur ton site.
- **Semaine 1** : `/seo cartographie` (un mot-clé et un prompt par page), `/seo quickwins`, corriger trois pages, un premier brief, une publication en brouillon.
- **Mois 1** : la routine du mercredi, le journal des modifications mesurées à J+28, un premier relevé de part de voix dans les IA (20 prompts), le rapport du mois.

---

1. Les skills et le repo GitHub

Le plugin **decupler-seo** est public, sous licence MIT : https://github.com/NathanFenina/decupler-seo

- 50 skills, 15 agents, 15 commandes `/seo`, 30 scripts Python, 13 MCP préconfigurés, 310 tests (README, v3.7.1).
- Un **skill** est une méthode écrite une fois dans un fichier `SKILL.md`, avec une description qui dit quand s'en servir. Claude le charge tout seul quand ta demande s'y prête.

Installation, dans Claude Code :

```
/plugin marketplace add NathanFenina/decupler-seo
/plugin install decupler-seo@decupler
/seo doctor
```

`/seo doctor` dit ce qui est branché, ce qui manque, ce que ça coûte, et lance la première action utile. Ensuite, tu parles normalement : « Audite mon site », « Quelles pages rattraper ce mois-ci ? », « Pourquoi ChatGPT ne me cite pas ? ».

Les commandes du plugin (README, v3.7.1) :

Commande	Ce qu'elle fait
`/seo audit <url>`	audit complet, spécialistes en parallèle, score sur 100, plan priorisé
`/seo quickwins`	les pages en position 4 à 20 à rattraper, chiffrées, corrections écrites
`/seo onpage <url> <mot-clé>`	note sur 100 critère par critère, puis réécrit la page
`/seo fix <url>`	détecte et corrige la technique : robots, canonicals, redirections
`/seo brief <mot-clé>`	brief adossé à la SERP réelle, réponse directe et FAQ déjà rédigées
`/seo article`	rédaction : recherche obligatoire, thèse, zéro chiffre inventé
`/seo geo <url>`	visibilité dans ChatGPT, Gemini, AI Overviews, et comment y entrer
`/seo rapport`	le mois écoulé : ce qui a bougé, pourquoi, et quoi faire

Avant d'installer un skill, quel qu'il soit, ouvre son `SKILL.md` : un skill peut lancer des scripts.

2. Les agents : audit, monitoring, reporting, création de contenu

Un **agent** est un spécialiste avec ses propres instructions et sa liste d'outils (un fichier dans `agents/`). Les skills et les commandes l'appellent, souvent plusieurs en parallèle.

- **Audit** : `/seo audit https://ton-site.fr` lance le skill `seo-audit-360`, qui délègue en parallèle aux agents seo-technique, seo-performance, seo-contenu, seo-crawler, seo-schema, seo-geo et seo-data, puis agrège en un score sur 100 et cinq urgences au plus.
- **Monitoring** : une routine cloud tourne le mercredi matin, même ordinateur éteint. L'agent `seo-manager` lit la mémoire du site, contrôle les pages en lecture seule, relève Search Console avant de toucher à quoi que ce soit, exécute seulement ce que tu as validé, et écrit le bilan.
- **Reporting** : l'agent `seo-data` et le script `rapport.py` sortent les chiffres ; Claude n'écrit que la lecture et les décisions.
- **Création de contenu** : `seo-serp` et `seo-analyste-concurrence` pour le brief, `seo-redacteur` pour le texte (recherche obligatoire, aucun chiffre inventé), `seo-publisher` pour le brouillon.

Le garde-fou : un hook optionnel (`hooks/garde-fou.sh`) intercepte toute commande qui écrirait sur le site (publication WordPress, robots.txt, redirections, email, post Reddit) et la fait autoriser, valider ou bloquer.

La routine du mercredi : sur https://claude.ai/code/routines, crée une routine Cloud sur le dépôt du site, chaque semaine, quelques minutes après l'heure pile (7 h 07 plutôt que 7 h 00). Son prompt tient en trois lignes et renvoie au dépôt (modèle dans `modeles/routine-hebdo.md`). Une routine ne charge pas les plugins : elle ne voit que le dépôt qu'elle clone. D'où un dépôt par site, avec la méthode copiée dedans (`projet.py init`, ou dis à Claude « nouveau projet »).

Un statut vert dans la liste des routines veut dire que la session a démarré et s'est terminée sans erreur technique, pas que le travail a été fait. Chaque passage écrit donc son journal dans `rapports/runs/`.

3. L'infra : MCP, connexions API, Search Console, Notion

Couche	Rôle
`CLAUDE.md` (mémoire)	Lu au début de chaque session et de chaque routine : ce qu'est le projet, ses règles, où est le reste
Skill	Une méthode, chargée quand la demande s'y prête
Agent	Un spécialiste avec ses outils, appelé en parallèle
Commande `/seo …`	Un raccourci qui lance un skill avec des arguments
MCP	Un branchement vers un outil externe (Search Console, DataForSEO, WordPress, Notion…)
Routine	Une session qui tourne seule à heure fixe, dans le cloud
Hook	Un script lancé automatiquement avant une action, qui peut la bloquer

Les 13 MCP du plugin, par niveau. Aucun n'est obligatoire : `/seo doctor` dit lesquels répondent.

- **Niveau 1, gratuit** : Search Console, GA4, Chrome DevTools. D'après la documentation du plugin, ils couvrent l'essentiel de la valeur.
- **Niveau 2, la donnée marché** : DataForSEO (paiement à la requête), Firecrawl, Ubersuggest.
- **Niveau 3, production et pilotage** : WordPress, Webflow, Notion, Semrush, Ahrefs.
- **Visibilité IA** : clés OpenAI et Gemini ; Claude et Perplexity en option ; Reddit.

Search Console, deux façons :

1. Le connecteur : Réglages de Claude → Connecteurs → autoriser Google → choisir la propriété. Cinq minutes, aucun code.
2. Le compte de service (recommandé pour les routines) : Google Cloud → activer la Search Console API → créer un compte de service → télécharger la clé JSON → dans Search Console, ajouter l'email du compte de service en lecture seule → vérifier avec `python3 scripts/gsc.py sites`.

Une propriété de domaine s'écrit `sc-domain:exemple.com`. L'API remonte 16 mois d'historique au plus.

Notion : le connecteur natif de claude.ai, sans clé. Dans les deux cas, partage explicitement la page avec l'intégration (« ••• » → Connexions) : c'est la cause de la plupart des « object not found ».

DataForSEO : les identifiants sont dans l'onglet API Access, pas ceux de connexion au site (cause n°1 des erreurs 401).

Les clés : jamais dans le dépôt, jamais dans la conversation.

- Les scripts du plugin lisent le fichier `.env` du projet (`/seo doctor` le crée).
- Les MCP lisent les variables de l'environnement où tourne Claude Code. Le plus sûr : les exporter dans ton terminal avant de lancer `claude`. Sinon, le bloc `env` de `.claude/settings.local.json`, un fichier personnel jamais commité : la documentation de Claude Code dit que ces variables valent pour chaque

session et les processus qu'elle lance. Vérifie avec `/mcp` que chaque serveur est connecté. Le `.mcp.json`, lui, ne contient que des références `${VARIABLE}`.

- Dans une routine cloud, les clés vont dans les variables de l'environnement de la routine.

Modèles : `modeles/exemple.mcp.json` et `modeles/settings.local.exemple.json`.

---

4. Les 17 SOP

Une SOP dit quoi faire et dans quel ordre. Le skill le fait. Chacune a son objectif, son moment, ses étapes, la commande à lancer, les MCP nécessaires, le livrable et les pièges. Elles sont aussi dans le dossier `sop/`, un fichier par SOP.

SOP 1 · Topical map

Skills : `seo-cartographie`, `seo-cocon-semantique` · **Commande** : `/seo cartographie, puis /seo cocon`

Objectif : Savoir quelle page doit gagner quel mot-clé sur Google et quelle question dans les IA, et comment les pages se tiennent en silos.

Quand : Au démarrage d'un site, puis une fois par mois pour le relevé.

Dis à Claude : « Fais la cartographie du site : une ligne par page avec son mot-clé principal et son prompt principal. Puis propose les silos et les pages piliers. »

Étapes

1. Claude croise Search Console (90 jours, page × requête), le sitemap et ton calendrier éditorial : une ligne par page, avec son mot-clé principal et trois secondaires (`cartographie.py initialiser`).
2. Il propose un prompt principal par page : la question qu'un acheteur poserait à ChatGPT, sans le nom de ta marque, en 20 mots au plus.
3. Tu valides ligne par ligne, en commençant par les pages P1. Le bon mot-clé est celui qui amène des clients, pas seulement des clics.
4. Contrôle de cannibalisation (`cartographie.py verifier`) : deux pages sur le même mot-clé, tu tranches avant d'écrire quoi que ce soit (fusion, redirection ou nouvelle cible).
5. Les silos : une page pilier par thème, des satellites qui la soutiennent, et le plan de liens entre les niveaux.
6. Chaque mois, `cartographie.py mensuel` relève la position de chaque page sur son mot-clé et la citation de son prompt par ChatGPT, Gemini et Claude.

MCP nécessaires : Search Console (ou un export CSV). DataForSEO ou Ubersuggest pour les volumes. Notion en option, pour l'export.

Livrable : `memoire/cartographie.csv` (une ligne par page : mot-clé, prompt, silo, priorité) et le plan de silos.

Pièges

- Un silo n'est pas un répertoire d'URL : c'est une intention commerciale.
- Un prompt qui contient ta marque mesure ta notoriété, pas ta visibilité.

- Le script propose, tu valides : la colonne `a_valider` doit être vide avant de produire.

> Chez nous : Sur decupler.com, la carte topique a croisé 143 URL du sitemap avec 12 mois de Search Console : 58 n'étaient pas indexées, et trois pages faisaient 42 % des clics (relevé du 22/09/2026).

SOP 2 · Ligne éditoriale

Skills : `seo-opportunites`, `seo-keyword-research` · **Commande** : `/seo opportunites`

Objectif : Décider quoi écrire, dans quel ordre et sur quel ton, à partir de ce qui rapporte.

Quand : Une fois par mois, avant de lancer la production.

Dis à Claude : « Classe mes opportunités par ce qu'elles rapportent et propose un calendrier de 8 semaines, 3 contenus par semaine. »

Étapes

1. Remplis le lexique de ton métier : tes services et ce qu'ils valent pour toi. C'est lui qui dit qu'un sujet compte plus qu'un autre.
2. Claude classe chaque opportunité : clics gagnables × valeur du thème × facilité, avec l'action à mener (title, enrichissement, consolidation, page à créer).
3. Il ajoute la demande où tu n'apparais pas encore (`demande.py idees --lexique`, avec DataForSEO). Indispensable sur un site jeune.
4. Il équilibre le funnel (par défaut 60 % découverte, 25 % comparaison, 15 % décision) et sort un calendrier (`opportunites.py --calendrier --semaines 8 --capacite 3`).
5. Il mesure ton style sur 5 à 10 de tes pages (`style_maison.py`) et l'écrit dans `memoire/style.md` : longueur des phrases, tutoiement ou vouvoiement, expressions qui reviennent.

MCP nécessaires : Search Console. DataForSEO ou Ubersuggest pour la demande.

Livrable : `rapports/opportunites-<date>.md`, `rapports/calendrier-<date>.md`, `memoire/style.md` et `memoire/marque.md`.

Pièges

- Prioriser au volume brut. On vise la valeur pour ton chiffre d'affaires.
- Écrire du neuf avant de rattraper les pages en position 4 à 20.
- Sans `locId`, les volumes Ubersuggest sont mondiaux : il faut le dire.

> Chez nous : En tête du classement de decupler.com le 01/10/2026 : « agence geo bordeaux », position 8,5, 78 impressions par mois, action « enrichir la page et son maillage ».

SOP 3 · Brief SEO/GEO

Skills : `seo-brief`, `seo-benchmark`, `seo-serp-analysis` · **Commande** : `/seo brief "<mot-clé>"`

Objectif : Un brief adossé à la page de résultats du jour, avec la réponse directe et la FAQ déjà rédigées.

Quand : Avant chaque page neuve ou chaque réécriture.

Dis à Claude : « Fais le brief SEO/GEO de « audit geo » : la SERP du jour, le top 5 lu, ce que personne ne couvre, la réponse directe et la FAQ. »

Étapes

1. La SERP en direct : top 10, « Autres questions posées », AI Overview et ses sources (`serp_concurrents.py`, avec DataForSEO). Sans DataForSEO, repli sur Firecrawl, puis sur Ubersuggest.
2. Le top 5 lu page par page : longueur, plan Hn, tableaux, FAQ, schémas, entités citées.
3. La matrice de gap : ce que le top traite, et ce que personne ne traite. C'est là qu'est ton angle.
4. Contrôle de cannibalisation : si une page du site vise déjà cette intention, on l'optimise au lieu d'en créer une autre.
5. Le brief en 18 sections : intention, plan chiffré, H1, réponse directe de 40 à 60 mots, FAQ, prompts IA et fan-out, données structurées, sources E-E-A-T, checklist GEO, liens internes tirés du sitemap.

MCP nécessaires : DataForSEO (ou Firecrawl, ou Ubersuggest). Search Console si la page existe déjà.

Livrable : Le brief en 18 sections et `recherche/serp-<mot-clé>-<date>.md`, avec l'origine de chaque chiffre.

Pièges

- Écrire un brief sans avoir regardé la SERP du jour.
- Ce qu'une source ne donne pas s'écrit « non mesuré », jamais « absent ».
- Un volume qu'aucun outil n'a renvoyé s'écrit « n.d. », pas une estimation.

> Chez nous : Brief « audit geo » du 05/10/2026 pour decupler.com : l'AI Overview citait 6 sources, dont aucune des 5 premières pages organiques. Coût des appels : 0,09 $.

SOP 4 · Rédaction

Skills : `seo-redaction`, `seo-eeat` · **Commande** : `/seo article`

Objectif : Un texte qui se lit comme le tien, sourcé, sans remplissage, prêt à être cité.

Quand : Après un brief validé.

Dis à Claude : « Rédige la page à partir de ce brief, dans mon style, puis note-la en E-E-A-T avant de me la rendre. »

Étapes

1. Vérifier qu'il faut écrire : aucune page du site ne vise déjà la même intention.
2. Recherche obligatoire : lecture réelle du top 3, et chaque chiffre vérifié à sa source.
3. Une thèse avant un plan : ce que ta page affirme et que les autres ne disent pas.
4. Rédaction par l'agent `seo-redacteur`, dans ton style mesuré (`memoire/style.md`), une donnée par paragraphe.

5. Test anti-remplissage sur chaque paragraphe, puis note E-E-A-T sur 40 et correctifs localisés.
6. `controle_contenu.py` bloque toute promesse ou tout chiffre sans source avant publication.

MCP nécessaires : DataForSEO ou Firecrawl pour la recherche. WordPress pour le brouillon.

Livrable : La page en HTML ou en Markdown, ses sources, son schema, et le verdict E-E-A-T (publier, réviser ou réécrire).

Pièges

- Un chiffre sans source : le contrôle bloque, et c'est voulu.
- Deux pages d'un même gabarit qui se lisent pareil.
- Un avis, une note, une anecdote ou une référence client inventés. Jamais.

SOP 5 · Optimisation on-page

Skills : `seo-optimisation-onpage`, `seo-meta-serp`, `seo-quick-wins` · **Commande** : `/seo onpage <url> <mot-clé>`

Objectif : Noter une page existante sur 100 face à son mot-clé, puis la réécrire.

Quand : Pour toute page en position 4 à 20 : c'est là que l'effort rapporte le plus vite.

Dis à Claude : « Note cette page sur 100 pour « claude code seo », critère par critère, puis réécris-la. »

Étapes

1. Donne le mot-clé cible. Sans cible, il n'y a pas d'optimisation possible.
2. `onpage_score.py` mesure les balises, la structure, le contenu, la sémantique et la lisibilité : un feu vert, jaune ou rouge par critère, et un malus de sur-optimisation.
3. Claude compare au top 3 : entités, sous-sujets et questions qui manquent.
4. Il réécrit la version complète, prête à coller, et propose 3 titles et 2 metas avec l'aperçu dans Google.
5. Les requêtes Search Console de la page qui ont des impressions sans clic deviennent des sections ou des questions de FAQ.

MCP nécessaires : Search Console. DataForSEO ou Firecrawl pour le top 3.

Livrable : Le rapport noté sur 100 et la version réécrite, avec title et meta.

Pièges

- Une meta hors de 120 à 156 caractères.
- Répéter le mot-clé partout : la note baisse, et la page aussi.
- Juger un changement au lendemain. On mesure à J+28, contre les pages qui n'ont pas bougé.

SOP 6 · Maillage interne

Skills : `seo-maillage-interne` · **Commande** : `/seo maillage`

Objectif : Faire circuler l'autorité vers les pages qui rapportent, avec un plan applicable ligne par ligne.

Quand : Après chaque publication, et une fois par trimestre sur tout le site.

Dis à Claude : « Trouve mes pages orphelines et propose 20 liens internes : la page de départ, l'ancre et l'endroit exact. »

Étapes

1. Crawl du site et autorité interne (`crawl_site.py`, `internal_pagerank.py`). Sur WordPress, la carte des liens du corps de texte (`wp.py carte`).
2. Cinq diagnostics : pages orphelines, pages qui convertissent mais reçoivent peu de liens, pages à plus de 3 clics de l'accueil, culs-de-sac, ancres vides ou sur-optimisées.
3. Le plan : depuis, vers, ancre, emplacement, raison. Une ligne doit pouvoir s'appliquer sans réfléchir.
4. Sur WordPress, `wp.py maillage --cible <url> --appliquer` pose le lien en révision quand l'ancre existe déjà dans le texte.
5. Positions de départ relevées, puis remesurées à J+30.

MCP nécessaires : Search Console, GA4 pour les conversions, WordPress pour appliquer.

Livrable : Le plan de liens en CSV et les révisions prêtes à valider.

Pièges

- Compter sur le menu et le pied de page : un lien dans le texte pèse bien plus.
- « Cliquez ici » ou « en savoir plus » : ces ancres ne disent rien à Google.
- Poser 400 liens en un jour. Le rythme conseillé : 20 à 30 par semaine.

SOP 7 · Entités

Skills : `seo-entites-triplets`, `geo-llms-txt` · **Commande** : `/seo entités`

Objectif : Que Google et les IA comprennent qui tu es et ce que tu fais, et retrouvent les mêmes faits sur toutes tes pages.

Quand : Une fois pour le site (l'entité), puis sur chaque page qui énonce des faits : prix, délais, lieux.

Dis à Claude : « Vérifie que Google comprend qui est derrière le site, puis liste les faits clés de cette page en triplets. »

Étapes

1. L'entité : une seule Organisation et une seule Personne, chacune avec un identifiant `@id` unique et ses profils en `sameAs`.
2. Le registre des faits (`memoire/triplets.csv`) : une seule valeur par sujet et par prédicat, par exemple « [ton offre] — coûte — [prix HT] ».
3. Sur chaque page, les 3 à 5 faits clés dans les 200 premiers mots, chacun avec son sujet nommé, jamais un pronom.
4. `triplets.py verifier` signale les faits absents, implicites ou contredits ; `triplets.py coherence` trouve les pages qui disent autre chose.

5. Les entités reliées à Wikidata dans le JSON-LD (`about`, `mentions`), un fichier `llms.txt`, et les robots des IA autorisés dans robots.txt.

MCP nécessaires : Aucun obligatoire. Firecrawl ou le navigateur pour lire les pages.

Livrable : Le registre des faits, le JSON-LD de l'entité et `llms.txt`.

Pièges

- « Il coûte… » n'est pas extractible : nomme le sujet dans la phrase.
- Un prix différent sur deux pages, et l'IA ne sait plus lequel citer.
- Baliser une date de création qui contredit ce que dit ton site.

> Chez nous : Sur decupler.com, Nathan Fenina apparaissait sous trois identifiants différents et deux adresses LinkedIn circulaient : tout a été relié à un identifiant par entité, sur 48 pages.

SOP 8 · Données structurées

Skills : `seo-schema-jsonld` · **Commande** : `/seo schema`

Objectif : Un balisage JSON-LD valide, fidèle à ce qu'on voit sur la page, relié à ton entité.

Quand : À chaque page publiée ou refondue.

Dis à Claude : « Génère le JSON-LD de cette page, valide-le, et vérifie qu'il correspond au contenu visible. »

Étapes

1. Un seul `@graph` par page : un socle identique partout (Organization et WebSite), puis les nœuds du type de page (WebPage, Service, BlogPosting, FAQPage, Person…).
2. Les `@id` : une entité, un identifiant, réutilisé sur tout le site.
3. Génération selon le type de page : service, article, page locale, FAQ, page auteur.
4. Validation par `schema_validate.py --strict`, sur le HTML initial : ce que voient les robots qui n'exécutent pas le JavaScript.
5. Contrôle à l'œil : chaque prix, note ou avis balisé apparaît bien sur la page.

MCP nécessaires : Aucun. Chrome DevTools pour vérifier le rendu.

Livrable : Le JSON-LD de la page, validé, et le socle commun à tout le site.

Pièges

- Un outil cité se balise `Thing` avec `sameAs`, pas `SoftwareApplication`.
- Pas d'`aggregateRating` sur ta propre Organisation : avis auto-attribués, ignorés et sanctionnables.
- `SearchAction` ne sert plus : le champ de recherche des sitelinks a disparu en novembre 2024.
- `headline` reprend le H1, en 110 caractères au plus.

SOP 9 · Design de page

Skills : `seo-design-pages`, `seo-page-builder-html` · **Commande** : `/seo design, puis /seo page`

Objectif : Une page qui se lit, se cite et convertit, construite selon son type.

Quand : Avant d'intégrer une page neuve ou une refonte.

Dis à Claude : « Quel gabarit pour cette page ? Fais le plan des boutons et des images, puis génère le HTML prêt à coller. »

Étapes

1. Typer la page : service, article, page locale, landing, comparatif, outil, lead magnet. En cas de doute, le format des trois premiers résultats tranche.
2. Le gabarit section par section, le plan de CTA (une action, ses emplacements, ses libellés) et les bannières de preuve.
3. Le plan d'images : un rôle par image, son poids, son texte `alt`.
4. Le HTML autonome, avec un CSS limité à la page, prêt pour WordPress, Elementor ou Webflow.
5. Contrôle bloquant : rendu réel à 390 et à 1 440 px, image principale jamais en `lazy`, chaque bouton mène quelque part, chaque preuve a sa source.

MCP nécessaires : Chrome DevTools pour le rendu réel. Un générateur d'images en option.

Livrable : `design-<slug>.md`, le fragment HTML et les captures à 390 et 1 440 px.

Pièges

- Un guide construit comme une landing : la réponse se noie dans l'argumentaire.
- Vérifier le rendu en ouvrant le fichier en local (`file://`) : les animations ne se déclenchent pas.
- Un bouton « Télécharger » qui ne télécharge rien.

> Chez nous : Cette page a suivi la même méthode : HTML assemblé par script, rendu vérifié dans Chromium à 390 et 1 440 px avant publication.

SOP 10 · Intégration CMS

Skills : `seo-publication-cms` · **Commande** : `/seo publish`

Objectif : Publier sans copier-coller, sans jamais casser la page en ligne.

Quand : Chaque fois qu'un contenu est prêt.

Dis à Claude : « Publie cette page en brouillon sur WordPress, avec son title, sa meta et son image, et donne-moi le lien d'aperçu. »

Étapes

1. Crée un mot de passe d'application WordPress (Utilisateurs → Profil). Jamais ton mot de passe de connexion.
2. `wp.py verifier` contrôle les droits, l'extension SEO et les champs ouverts à l'API.

3. Le garde-fou (`guard.py`) répond : autorisé, à valider, ou bloqué.
4. `wp.py publier` : brouillon par défaut, sauvegarde avant écriture, contrôle du contenu, puis lien d'aperçu. Une page déjà en ligne est modifiée en révision : la version publique ne bouge pas.
5. Vérification sur la page réelle après publication, puis la modification entre au journal, pour être mesurée à J+28.

MCP nécessaires : WordPress ou Webflow, ou directement l'API REST par `wp.py`.

Livrable : Le brouillon, son lien d'aperçu, la sauvegarde et la ligne du journal.

Pièges

- Yoast n'ouvre pas title et meta à l'API par défaut : sans un petit plugin qui les expose, WordPress répond 200 et jette les valeurs.
- WordPress transforme les lignes vides en paragraphes, y compris dans un `<style>` : la page casse en ligne alors qu'elle était parfaite en local.
- Mettre un `<h1>` dans le contenu quand le thème affiche déjà le titre.

SOP 11 · Indexation

Skills : `seo-indexation` · **Commande** : `/seo index`

Objectif : Que Google explore et indexe tes pages neuves ou corrigées, sans tricher.

Quand : Après chaque publication, et chaque semaine pour les pages importantes.

Dis à Claude : « Inspecte les pages publiées ce mois-ci, resoumets les sitemaps et prépare-moi les 10 URL à demander aujourd'hui. »

Étapes

1. Inspection par l'API de Search Console (`gsc.py inspect --fichier urls.txt`) : indexée ou non, date du dernier passage, canonique retenue par Google.
2. Diagnostic des pages non indexées : découverte mais pas explorée, explorée mais pas indexée, canonique, noindex, doublon.
3. Des sitemaps propres (seulement des URL indexables), resoumis par l'API (`gsc.py sitemap --resoumettre`).
4. Un lien vers chaque page neuve depuis une page que Google visite souvent : l'accueil ou une page hub.
5. La liste du jour, une dizaine d'URL, à passer dans Search Console : Inspection de l'URL, puis « Demander l'indexation ».

MCP nécessaires : Search Console, avec un compte de service. L'écriture complète n'est nécessaire que pour resoumettre les sitemaps.

Livrable : L'état d'indexation page par page et la liste priorisée des demandes du jour.

Pièges

- Croire qu'une API permet de « Demander l'indexation » : il n'en existe pas de publique. L'Indexing API ne couvre que les offres d'emploi et les vidéos en direct.

- Payer un service d'indexation : ça ne sert à rien.
- Se fier au rapport d'indexation, qui a plusieurs jours de retard. La vérité page par page vient de l'inspection.

Astuce, à tester : Claude Code peut piloter ton Chrome (`claude --chrome`, extension Claude in Chrome) dans ta session déjà connectée. Tu peux lui demander d'ouvrir l'inspection de chaque URL de la liste du jour et de cliquer sur « Demander l'indexation ». Je ne l'ai pas encore éprouvé à grande échelle : regarde-le faire la première fois, reste à une dizaine d'URL par jour, et il s'arrête de lui-même sur un CAPTCHA.

> Chez nous : Semaine du 22 au 25/09/2026 sur decupler.com : 459 URL inspectées une par une par l'API, sitemaps resoumis (154 pages lues, zéro erreur).

SOP 12 · Analyse GSC / GA4

Skills : `seo-gsc-analyses`, `seo-quick-wins` · **Commande** : `/seo quickwins, ou /seo gsc`

Objectif : Répondre à une question précise avec tes vraies données, et en tirer une action.

Quand : Chaque semaine pour les quick wins, et dès qu'une courbe bouge.

Dis à Claude : « Quelles pages sont en position 4 à 20 avec des impressions ? Classe-les et écris les corrections. »

Étapes

1. Brancher les données : `gsc.py` avec un compte de service (le plus fiable), le MCP Search Console, un connecteur, ou un export CSV.
2. Choisir la question parmi 21 analyses prêtes : gagnants et perdants, pages qui déclinent, cannibalisation, CTR anormal, requêtes sans page, saisonnalité, Google face aux IA…
3. Les quick wins : pages en position 4 à 20, triées en quatre paquets (snippet, contenu insuffisant, cannibalisation, presque là).
4. GA4 relie chaque page à ses conversions, pour chiffrer le potentiel en euros.
5. Chaque analyse se termine par une action et le skill qui l'exécute.

MCP nécessaires : Search Console et GA4.

Livrable : Le tableau de l'analyse et la liste d'actions, 10 pages au plus pour les quick wins.

Pièges

- Une propriété de domaine s'écrit `sc-domain:exemple.com`, sinon l'API renvoie zéro ligne.
- Chercher plus de 16 mois d'historique : l'API s'arrête là.
- Donner le rôle propriétaire au compte de service. La lecture seule suffit.

> Chez nous : Quick wins de decupler.com du 30/08 au 26/09/2026 : 13 requêtes entre la position 8 et 20, dont « agence geo bordeaux », 219 impressions, position 8,1, aucun clic.

SOP 13 · Visibilité IA

Skills : `geo-share-of-model`, `geo-visibilite-ia`, `geo-citation-tracker` · **Commande** : `/seo geo <url>, ou /seo share-of-model`

Objectif : Savoir si ChatGPT, Gemini, Claude et Perplexity te citent, qui ils citent à ta place, et pourquoi.

Quand : Une fois par mois, le même jour, avec la même liste de prompts.

Dis à Claude : « Mesure ma part de voix dans ChatGPT et Gemini sur mes 30 prompts, et compare à mes 3 concurrents. »

Étapes

1. La batterie : 20 à 40 questions d'acheteur, en quatre familles (découverte, comparaison, problème, marque). Les prompts principaux de ta cartographie en font partie.
2. L'interrogation (`share_of_model.py`) : ChatGPT et Gemini par API, avec la recherche web ; Claude et Perplexity si tu as leurs clés ; les AI Overviews par DataForSEO ou à la main.
3. Le relevé, prompt par prompt et moteur par moteur : cité ou non, à quel rang, avec ou sans lien, la phrase exacte, les sources, l'exactitude.
4. Le calcul : taux de citation, part de voix pondérée par le rang, par moteur, et le même calcul pour tes 3 concurrents.
5. Pour une page précise qui n'est jamais citée, `geo-citation-tracker` réécrit sa réponse directe (voir la partie GEO).

MCP nécessaires : Clés OpenAI et Gemini (les deux suffisent pour commencer). Anthropic et Perplexity en option. DataForSEO pour les AI Overviews.

Livrable : `donnees/ia-<mois>.csv` et le tableau de part de voix, par moteur et face aux concurrents.

Pièges

- Mélanger les prompts qui nomment ta marque (ta réputation) et les autres (ta visibilité).
- Changer la liste d'un mois à l'autre : on ajoute, on ne retire jamais.
- Fêter une mention inexacte. C'est un problème à corriger.

SOP 14 · Reporting

Skills : `seo-reporting`, `seo-dashboard`, `seo-pilotage` · **Commande** : `/seo rapport, puis /seo dashboard`

Objectif : Un bilan mensuel où chaque chiffre vient d'un script et chaque courbe a sa cause.

Quand : Le premier mercredi du mois.

Dis à Claude : « Fais le rapport du mois : ce qui a bougé, pourquoi, ce qui n'a pas marché, et les 5 actions du mois prochain. »

Étapes

1. Le relevé de la cartographie (`cartographie.py mensuel`), puis les chiffres (`rapport.py`) : le mois contre le mois précédent et contre le même mois l'an passé.

2. Les modifications du mois, et le verdict de celles arrivées à J+28, mesurées contre un groupe témoin de pages qui n'ont pas bougé.
3. Claude n'écrit que la section « Lecture et décisions » : les causes, ce qui n'a pas marché, les 5 actions.
4. Le tableau de bord HTML (`/seo dashboard`) ou les bases Notion (`/seo notion`) pour le client.
5. La page de suivi : ce qui attend une décision, ce qui est fait, les wins.

MCP nécessaires : Search Console et GA4. Notion en option.

Livrable : `rapports/<mois>.md`, le tableau de bord et la page de suivi à jour.

Pièges

- Un chiffre recalculé à la main dans un texte finit faux.
- Comparer seulement au mois précédent. Compare aussi au même mois de l'an passé.
- Appeler « win » une intention. Un win est un fait mesuré.

> Chez nous : Septembre 2026 sur decupler.com : 71 clics sur 4 semaines contre 33 les 4 semaines d'avant, et 1 impression sur les pages de spam du piratage contre 3 308 en mai.

SOP 15 · La semaine type d'un projet

Skills : `seo-cycle`, `seo-pilotage`, `seo-journal-mesure` · **Commande** : `la routine hebdo, puis /seo-maj le lundi`

Objectif : Que chaque projet avance chaque semaine, au même rythme, quel que soit celui qui le tient : toi, un prestataire ou la routine.

Quand : Toutes les semaines, sur chaque projet.

Dis à Claude : « Où en est le projet cette semaine : ce qui est fait, ce qui attend une décision, ce qui est bloqué ? »

Étapes

1. Lundi, 10 minutes : tu fusionnes la PR « Méthode decupler-seo » si elle est là (`/seo-maj` explique ce qui change), et tu déposes dans `notes/` ce que le client a dit, avec `#a-traiter`.
2. Mercredi 7 h, la routine `hebdo` : contrôle du site, chiffres, mesure à J+28, suivi de l'indexation, actions validées, notes `#a-traiter`, bilan sur la page de suivi, PR.
3. Mercredi, 20 minutes : tu ouvres la page de suivi, tu valides ou refuses ce qui est « À décider » (SOP 16), tu débloques ce qui attend quelqu'un, tu fusionnes ce qui attend un humain.
4. Du mercredi au mardi suivant : les actions validées, une SOP par action (brief, rédaction, optimisation, maillage, technique). Après chaque mise en ligne : SOP 11 (indexation) et une ligne au journal (SOP 14).
5. Premier mercredi du mois : la routine fait aussi le rapport (SOP 14) et propose les actions du mois. Tu en valides 3 à 5, pas plus.

MCP nécessaires : Search Console (compte de service), le CMS. Notion si le client suit dans Notion.

Livrable : La page de suivi à jour (fait, à décider, bloqué, wins chiffrés), le journal de run `rapports/runs/<date>-hebdo.md` et le journal des modifications.

Pièges

- Lancer une page neuve qui n'a pas été validée sur la page de suivi : la routine refuse, et elle a raison.
- Juger une modification avant J+28 : Search Console bouge trop lentement, on mesure contre un groupe témoin.
- Croire qu'un statut vert dans la liste des routines veut dire « travail fait » : seul le journal de run le dit.

SOP 16 · Arbitrer : quoi faire ce mois-ci

Skills : `seo-opportunites`, `seo-strategiste`, `seo-pilotage` · **Commande** : `/seo opportunites`

Objectif : Choisir les 3 à 5 actions du mois par ce qu'elles rapportent, avec les mêmes critères pour le contenu, la technique et le GEO, et dire ce qu'on ne fait pas.

Quand : Le premier mercredi du mois, avec le rapport, et chaque fois qu'une idée nouvelle arrive.

Dis à Claude : « Propose les actions du mois : contenu, technique et GEO, chiffrées, et dis-moi ce qu'on laisse de côté et pourquoi. »

Étapes

1. Une seule échelle : chaque action reçoit un gain estimé en euros par an (`Trafic = Volume × CTR de la position visée`, puis × taux de conversion × valeur d'un lead), un effort et une probabilité. Ordre = gain × probabilité ÷ effort.
2. D'abord ce qui bloque : page clé en 404, noindex accidentel, spam, canonique fausse. Pas de calcul, on corrige.
3. Ensuite ce qui rank déjà : les pages en position 4 à 20 (`/seo quickwins`) rapportent plus vite qu'une page neuve.
4. Puis le contenu neuf, par la ligne éditoriale (SOP 2) : un mot-clé non couvert, un service vendu derrière, au plus 3 pages neuves par semaine.
5. Le GEO en parallèle, pas à la place : les pages qui portent un prompt principal et que les IA ne citent pas reçoivent la réponse directe, les entités et les données structurées.
6. Écrire ce qu'on ne fait pas : chaque action écartée va dans `memoire/decisions.md` avec sa raison.

MCP nécessaires : Search Console. DataForSEO, Ahrefs ou Ubersuggest pour la demande. GA4 pour la valeur réelle d'une page.

Livrable : 3 à 5 actions chiffrées sur la page de suivi, et les actions écartées dans `memoire/decisions.md`.

Pièges

- Classer par volume de recherche : un mot-clé à 200 recherches qui vend peut valoir dix fois un mot-clé à 2 000 qui informe.
- Tout faire en même temps : 3 à 5 actions finies valent mieux que 15 commencées.
- Reproposer ce que le client a refusé : ça coûte sa confiance.

SOP 17 · Travailler avec un prestataire

Skills : `tous, selon la tâche` · **Commande** : `/seo doctor, puis la SOP de la tâche`

Objectif : Qu'un rédacteur, un intégrateur ou un consultant produise au même niveau que toi, sur le même projet, sans rien casser et sans tout réexpliquer.

Quand : À l'arrivée d'un prestataire sur un projet, puis à chaque tâche confiée.

Dis à Claude : « Je suis prestataire sur ce projet : lis la mémoire, dis-moi les règles, puis prépare la tâche « <tâche> » en suivant sa SOP. »

Étapes

1. Accès, le minimum : le dépôt du projet (jamais sur la branche principale), le CMS en rôle éditeur, Search Console en lecture. Les clés passent hors du dépôt, dans son propre `.env`.
2. Installation : Claude Code, puis `git clone` du dépôt. La méthode est déjà dedans (`.claude/`), et `/seo doctor` dit ce qui manque sur son poste.
3. Lecture avant la première tâche : `Accueil.md`, `CLAUDE.md`, `memoire/marque.md`, `memoire/style.md`, `memoire/decisions.md`, et la SOP de sa tâche.
4. Une tâche = une action validée = une branche = une PR. Il suit la SOP, passe les contrôles (`controle_contenu.py`, `humanisation.py`) et ouvre la PR. Tu relis et tu fusionnes.
5. Une amélioration de méthode se propose dans la PR, et tu la remontes dans decupler-seo (`projet.py remonter`) si elle vaut pour tous.

MCP nécessaires : Ceux de la tâche. Aucun accès en écriture à la production sans validation.

Livrable : Des PR qui suivent les SOP, journalisées, relues.

Pièges

- Donner un compte administrateur WordPress : personne n'en a besoin pour rédiger ou optimiser.
- Laisser un prestataire publier sans PR : la modification ne sera ni journalisée ni mesurée.
- Le laisser inventer un chiffre « pour illustrer » : la règle `memoire/faits.md` vaut pour tout le monde.

Un cas complet : le brief « audit geo » (5 octobre 2026)

La phrase dite à Claude : `/seo brief audit geo https://decupler.com/audit-geo/`

1. **Les chiffres du mot-clé** : Ubersuggest (France, `locId` 2250) donne 210 recherches par mois, difficulté 11 ; DataForSEO en donne 260. Coût : 0,09 $.
2. **La SERP du jour** : Firecrawl refuse (402, crédits épuisés), le skill passe à DataForSEO (0,00215 $). L'AI Overview cite 6 sources, dont aucune des 5 premières pages organiques.
3. **Le top 5 lu et mesuré** : de 583 à 4 263 mots, fourchette cible de 2 250 à 2 450. Notre page : 1 106 mots, hors top 10, deux H1 identiques.
4. **Le gap** : 18 sujets × 6 pages. Personne ne traite la remesure à J+30, llms.txt, Le Chat, Google AI Mode ni le trafic IA dans GA4 : c'est l'angle.
5. **Le plan** : H1 de 52 caractères, 7 H2, fan-out couvert à 10 sur 10.
6. **Le livrable** : une réponse directe de 53 mots, une FAQ de 6 questions, 7 prompts IA proposés (non mesurés), title et meta.

7. **Le contrôle « zéro chiffre inventé »** : 3 corrections avant de rendre le brief.

Coût total des API : 0,09215 $. Ce qui n'a pas été mesuré, et le brief le dit : les réponses des IA aux 7 prompts, la Search Console de la page.

---

5. Les techniques SEO

- **Briefs** : SOP 3. Jamais sans la SERP du jour ; la réponse directe et la FAQ sont rédigées dans le brief, pas laissées au rédacteur.
- **Données structurées** : SOP 8. Un `@graph` par page, un `@id` par entité, rien de balisé qui ne soit visible.
- **Entités** : SOP 7. Une valeur par fait sur tout le site, des sujets nommés, Wikidata dans le JSON-LD.
- **Maillage interne** : SOP 6. Un plan applicable ligne par ligne, 20 à 30 liens par semaine.
- **Backlinks** (skill `seo-netlinking`, commande `/seo backlink`) : on compte les domaines référents, pas les liens. Un profil sain garde les ancres exactes sous 15 % ; au-delà de 15 à 20 %, le profil a l'air construit. Préfère la régularité à un pic d'acquisition. Les emails de prise de contact sont rédigés, mais l'envoi reste une décision humaine.
- **Prioriser en euros** : la formule que j'utilise (skills `audit-seo-geo` et `cartographie-client`). Trafic par mois = volume × CTR de la position visée (position 1 ≈ 0,28 · top 3 ≈ 0,15 · top 5 ≈ 0,08 · top 10 ≈ 0,03). Potentiel par an = trafic × taux de conversion de l'intention (transactionnelle 3 % · commerciale 2 % · informationnelle 0,5 %) × valeur d'un lead × 12. Valeur d'un lead = valeur d'un client signé × taux de closing (par défaut dans le modèle : 15 000 € × 12 % = 1 800 €, à remplacer par tes chiffres). Exemple avec « audit geo » (210 recherches par mois), top 3, intention commerciale : 210 × 0,15 = 31,5 visites ; × 2 % = 0,63 lead ; × 1 800 € × 12 = 13 608 € par an.
- **Suivi de position** : la cartographie mensuelle (position de chaque page sur son mot-clé principal), les instantanés hebdomadaires de Search Console (`gsc.py instantane`), et le journal des modifications mesurées à J+28 contre un groupe témoin. Exemple de la documentation : si tout le site a pris +12 % et la page +17 %, la modification n'explique que 5 points, et le verdict est « neutre ».

---

6. Le GEO : rendre une page citable par ChatGPT, Perplexity et les AI Overviews

Une IA cite un passage, pas une page. Le travail se fait dans cet ordre :

1. **L'accès** : les robots des IA autorisés dans robots.txt (GPTBot, OAI-SearchBot, ChatGPT-User, ClaudeBot, Claude-SearchBot, PerplexityBot, Perplexity-User, Google-Extended…), le contenu présent dans le HTML sans JavaScript, la page indexable, un `llms.txt`.
2. **La réponse directe** sous le H1 : 40 à 60 mots qui répondent dès la première phrase, se comprennent seuls, nomment le sujet et contiennent un fait précis quand c'est possible (skill `geo-citation-tracker`).
3. **Des H2 formulés comme des questions**, avec sous chacun un passage citable de 40 à 120 mots.
4. **Le bon format pour chaque information** : comparaison en tableau, étapes en liste numérotée, critères en liste.

5. **Des faits sourcés et datés**, une entité claire (auteur, organisation), des données propres quand tu en as.
6. **Les données structurées** du type de page, FAQPage compris.
7. **La mesure** : la part de voix chaque mois (SOP 13), et pour une page transformée, des prompts neutres relevés avant, à J+30 et à J+60.

Le rang ne garantit pas la citation. Le 05/10/2026, l'AI Overview de Google sur « audit geo » citait 6 sources, dont aucune des 5 premières pages organiques.

Notre référentiel « GEO Ready » formalise ces règles : 26 règles, 6 blocs notés sur 100 (extractibilité 30, faits et fraîcheur 20, entité et autorité 20, données structurées 10, accès IA 10, couverture 10), et 5 bloquants. Une seule règle bloquante non levée, et la page est « non citable », quel que soit son score : pas de réponse directe sous le H1, contenu daté de plus de 24 mois, robots des IA bloqués, corps absent sans JavaScript, ou page en noindex. Le simulateur gratuit : https://decupler.com/geo-ready/

---

7. Bonus : des systèmes d'acquisition avec Claude Code

Les mêmes briques servent à trouver des clients pour tes services SEO.

1. **Le pré-audit en masse** : pour chaque prospect, 10 prompts neutres de son secteur (sa marque n'y est jamais nommée) posés à ChatGPT, Claude et Gemini, soit 30 réponses. Un script extrait qui est cité et à quel rang ; un autre génère une page publique de pré-audit, en `noindex`.
2. **Le cold mail qui montre la faille** : chaque pré-audit devient une ligne de séquence (constat, chiffre, concurrent cité à sa place, lien). Le premier mail montre le problème ; il ne le répare pas.
3. **Le lead magnet outil** : un simulateur public (le nôtre note une page selon le référentiel GEO Ready) qui capte les demandes entrantes.
4. **Les démos de site** : pour un commerce local, une maquette de son futur site, dans sa charte et avec ses vraies informations, envoyée en disant clairement que c'est une démo.

Ce que je ne te donne pas : un taux de réponse. Je ne l'ai pas encore mesuré proprement, je ne vais pas en inventer un.

Ce que Claude Code ne fait pas à ta place

- **Cliquer sur « Demander l'indexation »** par une API : elle n'existe pas. Il prépare la liste ; l'astuce Claude in Chrome est à tester.
- **Choisir tes priorités business** : quelle offre compte ce trimestre, c'est toi qui le sais.
- **Garantir une citation dans les IA** : il rend la page citable et mesure. Le moteur décide.
- **Se passer de relecture** : vérifie le rendu réel de chaque page avant de publier.

Sources

- Plugin decupler-seo v3.7.1 : https://github.com/NathanFenina/decupler-seo (README, docs/, skills/)
- Documentation de Claude Code (installation, MCP, Chrome, routines) : https://code.claude.com/docs, relevée le 05/10/2026
- Les chiffres de decupler.com : Search Console et page de suivi du site, septembre et octobre 2026
- La page en ligne : https://decupler.com/claude-code-seo-os/