Superpowers *mémo*

L'essentiel pour utiliser **Superpowers**, la méthodologie de Jesse Vincent (obra) qui impose à Claude Code ou Codex un cycle brainstorming → plan → TDD → revue. Les skills se déclenchent automatiquement : il n'y a presque rien à taper.

Famille Méthodes de développement · **Éditeur** obra · **Licence** MIT
Note du wiki 11/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/methodes-de-developpement/superpowers

Installation

`/plugin install superpowers@claude-plugins-official`	Claude Code – marketplace officielle Anthropic
`/plugin marketplace add obra/superpowers-marketplace` `/plugin install superpowers@superpowers-marketplace`	Claude Code – marketplace de l'auteur (avec plugins associés)
`/plugins`	Codex CLI : ouvrir la recherche de plugins, chercher « superpowers », puis Install

Vérifier l'installation : dans une nouvelle session, écrire « Let's make a react todo list ». Si tout fonctionne, l'agent lance le brainstorming avant d'écrire la moindre ligne de code.

Le workflow (déclenché automatiquement)

Étape	Skill	Ce qu'elle fait
1	`brainstorming`	Avant tout code : pose des questions, explore des alternatives, présente le design par morceaux à valider, sauvegarde un design doc.
2	`using-git-worktrees`	Après validation : crée un espace isolé sur une nouvelle branche et vérifie que les tests passent au départ.

Le workflow (déclenché automatiquement) (suite)

Étape	Skill	Ce qu'elle fait
3	`writing-plans`	Découpe le travail en tâches de 2 à 5 minutes, avec chemins de fichiers, code complet et étapes de vérification.
4	`subagent-driven-development` ou `executing-plans`	Exécute le plan : un sous-agent neuf par tâche avec revue après chacune (le plus rigoureux), ou tout dans la session avec une seule revue finale (le moins coûteux).
5	`test-driven-development`	Impose RED → GREEN → REFACTOR : test qui échoue, code minimal, test qui passe, commit. Supprime le code écrit avant son test.
6	`requesting-code-review`	Entre les tâches : revue par rapport au plan, problèmes classés par gravité ; les critiques bloquent.
7	`finishing-a-development-branch`	À la fin : vérifie les tests, propose merge / PR / garder / abandonner, nettoie le worktree.

Autres skills utiles

Skill	Quand elle sert
`systematic-debugging`	Débogage en 4 phases par cause racine.
`verification-before-completion`	Oblige à prouver qu'un correctif marche avant de déclarer « c'est fait ».
`dispatching-parallel-agents`	Lance plusieurs sous-agents en parallèle (recherche, analyses indépendantes).
`receiving-code-review`	Traite proprement des retours de revue.
`writing-skills`	Crée de nouvelles skills en suivant les bonnes pratiques.

Autres skills utiles (suite)

Skill	Quand elle sert
`diagnosing-superpowers`	Analyse une session qui s'est mal passée. Déclencheur : « figure out what went wrong with superpowers in this session ».

Conseils d'utilisation

- **Décrire le problème, pas la solution.** Les skills choisissent elles-mêmes l'étape suivante ; inutile de les invoquer à la main.
- **Choisir le mode d'exécution selon le budget** : `subagent-driven-development` isole chaque tâche dans un contexte neuf (plus fiable, plus d'appels) ; `executing-plans` est moins cher. Un utilisateur mesure 40 à 50 % du contexte consommé par brainstorming + plan avant la première ligne de code.
- **Sauter le workflow pour les petites choses** : une faute de frappe, une retouche CSS ou une urgence en production ne justifient pas le cycle complet.
- **Éviter les équipes mixtes** : si un coéquipier utilise Claude Code « nu », les design docs et plans de Superpowers peuvent embrouiller plus qu'aider.
- **Placer les worktrees sur un disque rapide** : le workflow crée beaucoup de branches temporaires.
- **Désactiver la télémétrie** du compagnon visuel si besoin : `SUPERPOWERS_DISABLE_TELEMETRY=1`.

Sources (documentation officielle)

- README Superpowers
- How I use obra/superpowers – Gautam Khorana
- Issue #512 – efficacité de brainstorming/writing-plans
- Issue #703 – exécuter writing-plans dans un sous-agent