AI TOOLS WIKI · MÉMO

Planning with Files *mémo*

L'essentiel pour utiliser **Planning with Files**, la skill qui garde le plan de l'agent sur disque dans trois fichiers Markdown. Le plan survit à `/clear`, à la compaction et aux plantages : la session suivante reprend à la bonne phase.

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

Installation

`/plugin marketplace add OthmanAdi/planning-with-files` `/plugin install planning-with-files@planning-with-files`	Claude Code – plugin complet (skill + hooks + slash commands)
`npx skills add OthmanAdi/planning-with-files --skill planning-with-files -g`	Codex, Cursor et 60+ autres agents (standard Agent Skills)

Seul le plugin Claude Code apporte les slash commands et les hooks automatiques.

Les trois fichiers

Fichier	Rôle
`task_plan.md`	Phases et cases à cocher. C'est le point de reprise après `/clear`.
`findings.md`	Notes de recherche et décisions, ajoutées au fil de l'eau.
`progress.md`	Journal de session et résultats de tests.

Pour plusieurs tâches en parallèle : un dossier `.planning/AAAA-MM-JJ-slug/` par tâche, avec les mêmes trois fichiers.

Commandes (plugin Claude Code)

Taper la forme courte, l'autocomplétion trouve la commande complète `/planning-with-files:<verbe>`.

`/plan`	Crée les trois fichiers et démarre la session planifiée.

Commandes (plugin Claude Code) (suite)

`/pwf`	Alias court de `/plan`. Options : `--autonomous`, `--gated`.
`/status`	Rapport d'un coup d'œil : phase en cours et avancement.
`/plan-doctor`	Auto-diagnostic : une ligne PASS/WARN/FAIL par mécanisme (injection, attestation, installation, latence).
`/plan-attest`	Verrouille `task_plan.md` par empreinte SHA-256 : un plan modifié à ton insu est refusé. `--show` / `--clear`.
`/plan-goal`	Travaille jusqu'à ce que le plan soit terminé, en s'appuyant sur `/goal` de Claude Code.
`/plan-loop`	Cadence de reprise sur `/loop`, toutes les 10 minutes par défaut.
`/plan-de`, `-es`, `-ar`, `-zh`, `-zht`	Démarre le plan en allemand, espagnol, arabe ou chinois.

Modes

Mode	Activation	Effet
Standard	`/plan`	Le plan est réinjecté à chaque tour et rappelé après les écritures.
Autonome	`/pwf --autonomous`	Moins de rappels à chaque appel d'outil, attestation activée par défaut. Pour les longues tâches sans supervision.
Gated	`/pwf --gated`	L'agent ne peut pas s'arrêter tant que le plan n'est pas terminé.

Variables d'environnement utiles

Variable	Effet
`PLANNING_DISABLED=1`	Ignore le plan pour cette session (utile en CI ou pour une tâche ponctuelle dans le même dossier).
`PLAN_ID=<slug>`	Attache le terminal à un plan précis de `.planning/`.
`PWF_INJECT=smart`	N'injecte que l'essentiel : objectif, prochaine étape, phase en cours, trois dernières décisions.

Les règles que suit l'agent

1. **Plan d'abord** : ne jamais démarrer une tâche complexe sans `task_plan.md`.
2. **Règle des 2 actions** : sauvegarder les découvertes dans `findings.md` toutes les deux lectures ou recherches.
3. **Noter toutes les erreurs** pour ne pas les répéter.
4. **Ne jamais retenter à l'identique** un échec : changer d'approche.

Conseils d'utilisation

- **À réserver aux tâches longues** : recherche, refonte, tout ce qui demande plus de cinq appels d'outils ou plusieurs sessions. Pour une petite correction, c'est inutile.
- **Garder la compaction automatique activée** : le hook `PreCompact` sauvegarde l'avancement juste avant.
- **Relire `task_plan.md` avant chaque décision importante**, ou demander à l'agent de le faire.
- **Les fichiers sont ignorés par git par défaut.** Les committer si tu veux garder une trace des décisions.
- **Si le plan semble ignoré**, lancer `/plan-doctor` avant toute autre hypothèse.

Sources (documentation officielle)

- README Planning with Files
- Guide d'installation
- Planning With Files – Claude Code Marketplaces