AI TOOLS WIKI · MÉMO

Oh My Claude Code *mémo*

Les commandes à connaître pour **Oh My Claude** (oh-my-claudecode), le framework d'orchestration multi-agents pour Claude Code. OMC a deux surfaces : des **skills en session** (`/...`, à taper dans Claude Code) et un **CLI terminal** (`omc ...`).

Famille Orchestration et catalogues d'agents · **Éditeur** Yeachan-Heo · **Licence** MIT
Note du wiki 10/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/orchestration-et-catalogues/oh-my-claude

Installation

`/plugin marketplace add https://github.com/Yeachan-Heo/oh-my-claudecode` `/plugin install oh-my-claudecode`	Dans Claude Code — une ligne à la fois
`npm i -g oh-my-claude-sisyphus@latest`	Ou le CLI via npm
`/omc-setup` `omc setup`	Configuration (dans la session, ou depuis le terminal)

Pour le mode équipe natif, activer dans `~/.claude/settings.json` : `"CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS": "1"`.

Skills en session

`/deep-interview "idée floue"`	Clarifie les besoins par questions socratiques. **Point de départ si l'idée n'est pas claire.**
`/ralplan "planifie cette feature"`	Planification itérative jusqu'à consensus. Mot-clé équivalent : `ralplan`.
`/autopilot "construis une API REST"`	Exécution autonome de bout en bout : besoins, plan, implémentation, QA, validation. Mot-clé : `autopilot: …`.
`/execute "refactor auth"`	Mène une tâche approuvée jusqu'au code vérifié, avec boucles vérification/correction.
`/ralph "refactor auth"`	Mode persistance : l'agent vérifie, corrige et relance jusqu'à ce que ce soit vraiment fini. Mot-clé : `ralph`.

Skills en session (suite)

`/team 3:executor "corrige toutes les erreurs TypeScript"`	Orchestration d'équipe : pipeline plan → PRD → exécution → vérification → correction, avec plusieurs agents sur une liste de tâches partagée.
`/ask codex "revois ce patch"`	Demande l'avis d'un autre modèle (claude, codex, gemini, antigravity, grok, cursor) et sauvegarde la réponse en Markdown.
`/skill list`, `add`, `remove`, `edit`, `search`	Gère les skills personnalisées du projet (`.omc/skills/`) ou de l'utilisateur (`~/.omc/skills/`).
`/skillify`	Extrait une skill réutilisable de la session en cours.
`/oh-my-claudecode:hud setup`	Configure la barre d'état qui affiche l'orchestration en temps réel.

Mots-clés à écrire dans le prompt

Mot-clé	Effet
`deepsearch …`	Recherche ciblée dans le codebase.
`ultrathink …`	Mode de raisonnement approfondi.
`stopomc` ou `cancelomc`	Arrête tous les modes OMC actifs.

CLI terminal

`omc team 2:codex "review auth flow"`	Lance des vrais processus CLI (codex, gemini, claude, cursor, grok, antigravity) dans des panneaux tmux.
`omc team status <nom>` / `omc team shutdown <nom>`	Suit ou arrête une équipe tmux.
`omc ask <provider> --prompt "…"`	Même avis multi-modèles que `/ask`, depuis le terminal ou la CI.
`omc lookout scan --brief "…"`	Analyse la tâche et l'espace de travail avant un run autonome et signale les risques.
`omc wait --start` / `--stop`	Relance automatiquement la session quand la limite de débit se réinitialise (nécessite tmux).

CLI terminal (suite)

`omc session friction report --since 24h`	Résume les signaux de contexte surchargé et de friction des dernières sessions.
`omc hud`	Affiche la HUD en direct.

Quel mode choisir ?

Besoin	Mode
Idée vague, besoins à préciser	`/deep-interview`
Fonctionnalité complète avec peu de cérémonie	`/autopilot`
Tâche qui doit aller au bout, sans résultat partiel	`/execute` ou `/ralph`
Plusieurs agents en parallèle	`/team` (en session) ou `omc team` (tmux)
Deuxième avis d'un autre modèle	`/ask codex …`

Conseils d'utilisation

- **Une seule boucle maître par session** : ne pas combiner `/goal` natif, `/execute`, `/team` et `/ralph` en même temps.
- **Pas de slash commands en CI** : `/autopilot`, `/ralph`, `/execute` et `/team` exigent une session Claude Code active. En CI, utiliser `omc setup`, `omc ask`, etc.
- **Répartir les rôles entre modèles** : Claude orchestre, Codex fait la revue ou la sécurité, Gemini l'UI ou la documentation.
- **Committer** `.omc/skills/` pour partager les skills apprises avec l'équipe ; le reste de `.omc/` est de l'état local (prompts, transcriptions) à ne pas committer.
- **Worktrees** : l'état `.omc/` disparaît avec le worktree. Définir `OMC_STATE_DIR` pour le conserver.
- **Mots-clés supprimés** : `swarm` est remplacé par `/team`, et « plan this » par `ralplan`.

Sources (documentation officielle)

- README oh-my-claudecode
- Référence complète (docs/REFERENCE.md)
- Oh My Claude (omc) – cmaven
- oh-my-claudecode Review – Vibe Coding Hub