AI TOOLS WIKI · MÉMO

wshobson/agents *mémo*

Les commandes à connaître pour **Wshobson Agents**, la marketplace de 94 plugins (202 agents, 183 skills, 105 commandes) pour Claude Code, également utilisable dans Codex, Cursor, OpenCode ou Copilot. Chaque plugin est une unité indépendante : on n'installe que ce dont on a besoin.

Famille Orchestration et catalogues d'agents · **Éditeur** wshobson · **Licence** MIT
Note du wiki 11/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/orchestration-et-catalogues/wshobson-agents

Installation

`/plugin marketplace add wshobson/agents`	Claude Code : ajouter la marketplace (ne charge rien dans le contexte)
`/plugin install python-development`	Puis installer 2 ou 3 plugins utiles à ton domaine
`npx codex-marketplace add wshobson/agents`	Codex
`npx skills add wshobson/agents --skill python-testing-patterns`	Une skill seule, dans n'importe quel agent

`/plugin` liste les commandes des plugins installés.

Format des commandes

`/nom-du-plugin:nom-de-la-commande` *[arguments]*, par exemple `/tdd-workflows:tdd-cycle`.
On peut aussi appeler un agent en langage naturel : « Utilise backend-architect pour concevoir l'API d'authentification ».

Commandes les plus utiles

Développement et tests

`/full-stack-orchestration:full-stack-feature`	Fonctionnalité complète, du backend au frontend, avec plusieurs agents.
`/backend-development:feature-development`	Fonctionnalité backend de bout en bout.

Commandes les plus utiles (suite)

`/tdd-workflows:tdd-cycle`	Cycle TDD complet (aussi `tdd-red`, `tdd-green`, `tdd-refactor`).
`/unit-testing:test-generate`	Génère des tests unitaires.

Revue, débogage, sécurité

`/comprehensive-review:full-review`	Revue multi-angles (architecture, sécurité, performance…).
`/git-pr-workflows:pr-enhance`	Améliore une pull request.
`/debugging-toolkit:smart-debug`	Débogage interactif.
`/error-debugging:error-trace`	Analyse d'une stack trace.
`/incident-response:incident-response`	Gestion d'un incident en production.
`/security-scanning:security-sast`	Analyse de sécurité statique.
`/security-scanning:security-dependencies`	Vulnérabilités des dépendances.
`/accessibility-compliance:accessibility-audit`	Audit WCAG.

Maintenance et documentation

`/code-refactoring:refactor-clean`	Nettoyage et refactoring.
`/code-refactoring:tech-debt`	Gestion de la dette technique.
`/framework-migration:deps-upgrade`	Mise à jour des dépendances.
`/code-documentation:doc-generate`	Génère la documentation.
`/c4-architecture:c4-architecture`	Documentation d'architecture au format C4.
`/context-management:context-save` / `context-restore`	Sauvegarde et restaure le contexte d'une conversation.

Démarrer un projet

`/python-development:python-scaffold`	Projet FastAPI ou Django.
`/javascript-typescript:typescript-scaffold`	Projet Next.js ou React + Vite.

Commandes les plus utiles (suite)

`/systems-programming:rust-project`	Projet Rust.

Modèles utilisés par les agents

Niveau	Modèle	Pour
1	Opus	Architecture, sécurité, revue de code, code critique
2	hérité de ta session	Backend, frontend, IA
3	Sonnet	Docs, tests, débogage
4	Haiku	Tâches rapides : SEO, déploiement, contenu

Conseils d'utilisation

- **Installer 2 ou 3 plugins, pas tout le catalogue** : chaque plugin installé ajoute ses agents et skills au contexte.
- **Parcourir le catalogue avant d'installer** : `docs/plugins.md` et `docs/agents.md` du dépôt listent tout sans rien charger.
- **Préférer les commandes d'orchestration** (`full-stack-feature`, `full-review`) pour les gros sujets, et les agents appelés par leur nom pour les tâches ciblées.
- **Surveiller le coût** : les orchestrations enchaînent plusieurs agents, dont certains sur Opus.
- **Évaluer une skill** avec le plugin `plugin-eval` avant de l'adopter en équipe.

Sources (documentation officielle)

- README wshobson/agents
- Guide d'utilisation (docs/usage.md)
- Catalogue des plugins