AI TOOLS WIKI · MÉMO

Everything Claude Code (ECC) *mémo*

Les commandes à connaître pour **Everything Claude Code**, la collection d'agents, skills, hooks, règles et configurations MCP d'Affaan Mustafa pour Claude Code, Codex, Cursor et OpenCode.

Le projet s'appelle désormais **ECC** ; les commandes ci-dessous correspondent à la version 2.2 de affaan-m/ECC.

Famille Méthodes de développement · **Éditeur** Affaan Mustafa · **Licence** MIT
Note du wiki 10/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/methodes-de-developpement/everything-claude-code

Installation

`npx ecc-universal@2.2.2 setup`	Recommandé : installation guidée (Claude Code, Codex, Kimi Code)
`/plugin marketplace add https://github.com/affaan-m/ECC` `/plugin install ecc@ecc`	Ou les commandes natives de Claude Code
`/plugin list ecc@ecc`	Vérifier ce qui est installé

Une seule méthode par agent. Installer ECC deux fois dans le même agent (plugin + copie manuelle) duplique skills, commandes et hooks. Les plugins Claude Code ne peuvent pas distribuer de règles : copier à la main seulement `rules/common` et le pack du langage utilisé.

Par quoi commencer

Ce que tu fais	Commande ou skill
Construire une fonctionnalité	`/ecc:plan "décris la fonctionnalité"`, puis la skill `tdd-workflow`
Corriger un bug	Reproduire avec un test qui échoue, puis `tdd-workflow`
Relire du code	`/code-review` (revue dans un contexte neuf)
Réparer un build	`/build-fix`
Nettoyer le code mort	`/refactor-clean`
Vérifier la pression sur le contexte	`/context-budget`
Terminer une longue session	`/save-session` ou `/learn-eval`
Reprendre plus tard	`/resume-session`
Auditer la sécurité	`/security-scan`

Commandes principales

Commande	Ce qu'elle fait	Agent utilisé
`/ecc:plan "…"`	Plan d'implémentation détaillé d'une fonctionnalité.	planner (+ architect)
`/code-review`	Revue qualité et sécurité du code récent.	code-reviewer
`/build-fix`	Diagnostique et corrige un build en échec.	build-error-resolver
`/refactor-clean`	Supprime le code mort.	refactor-cleaner
`/security-scan`	Audit OWASP Top 10.	security-reviewer
`/update-docs`	Met la documentation à jour.	doc-updater
`/test-coverage`	Vérifie la couverture de tests (objectif 80 %).	–
`/go-review`, `/python-review`	Revue spécialisée par langage.	go-reviewer, python-reviewer
`/context-budget`	Mesure ce que chaque agent, skill, serveur MCP et règle coûte en tokens, et propose des économies.	–
`/save-session`	Sauvegarde l'état de la session (ce qui a marché, échoué, reste à faire) dans `~/.claude/session-data/`.	–
`/resume-session`	Reprend une session sauvegardée.	–
`/learn-eval`	Extrait les apprentissages de la session.	–
`/ecc:configure-ecc`	Reconfigure ECC après installation.	–

Avec le plugin, les commandes ont le préfixe `ecc:` (ex. `/ecc:plan`). Une installation manuelle peut exposer la forme courte (`/plan`). Les anciens raccourcis comme `/tdd` ou `/eval` ne sont plus actifs par défaut.

Skills clés (déclenchées par l'agent ou par leur nom)

Skill	Rôle
`tdd-workflow`	Impose d'écrire les tests avant le code.
`e2e-testing`	Tests de bout en bout des parcours critiques (Playwright).
`context-budget`	Audit de la consommation de contexte.
`verification-loop`, `eval-harness`	Boucles de vérification et évaluation.

Workflows types

- **Nouvelle fonctionnalité** : `/ecc:plan "…"` → `tdd-workflow` → `/code-review`

Workflows types (suite)

- **Bug** : `tdd-workflow` (test qui reproduit) → correctif → `/code-review`
- **Avant la production** : `/security-scan` → `e2e-testing` → `/test-coverage`

Réglages utiles (variables d'environnement)

Variable	Effet
`ECC_SESSION_START_MAX_CHARS=8000`	Plafonne le contexte injecté au démarrage (valeur par défaut).
`ECC_SESSION_START_CONTEXT=off`	Désactive cette injection (configurations légères ou modèles locaux).
`ECC_DISABLED_MCPS="github,supabase"`	Évite les doublons avec tes propres serveurs MCP.
`CLAUDE_PACKAGE_MANAGER=pnpm`	Force le gestionnaire de paquets (sinon détecté automatiquement ; aussi via `/setup-pm`).

Conseils d'utilisation

- **Partir du besoin, pas du catalogue** : ECC compte plus de 260 skills. N'en charger que ce qui correspond à ta stack.
- **Commencer par un profil minimal** : chaque règle, skill ou MCP chargé consomme du contexte dans toutes les sessions. `/context-budget` aide à faire le tri.
- **Activer les serveurs MCP à la demande** (via `/mcp`) plutôt que tous en même temps.
- **Sauvegarder avant de fermer** une longue session (`/save-session`), pour reprendre sans tout réexpliquer.
- **Pour un usage occasionnel** (quelques sessions par semaine), un simple `CLAUDE.md` bien écrit suffit souvent.
- **En cas de doublons**, désinstaller proprement avant de réinstaller (section *Reset / Uninstall* du README).

Sources (documentation officielle)

- README ECC (affaan-m/ECC)
- Longform Guide
- Everything Claude Code – tutoriel DataCamp
- Skill context-budget