AI TOOLS WIKI · MÉMO

Claude Code *mémo*

Les commandes, raccourcis et fichiers à connaître pour **Claude Code**, l'agent de développement d'Anthropic. Voir aussi la fiche **Claude Code**, la synthèse **des avis** et, pour la revue de code en détail, **Claude Code Review mémo**.

Famille Agents de développement · **Éditeur** Anthropic · **Licence** Propriétaire (Anthropic Commercial Terms)
Note du wiki 13/15 · **Commandes vérifiées le** 25 septembre 2026
Fiche, notes et avis aitoolswiki.viellard.fr/outils/agents-de-developpement/claude-code

Installation

`curl -fsSL https://claude.ai/install.sh | bash`	macOS, Linux, WSL (installateur natif, mises à jour automatiques)
`irm https://claude.ai/install.ps1 | iex`	Windows PowerShell
`brew install --cask claude-code` `winget install Anthropic.ClaudeCode`	Homebrew (canal stable) ou WinGet : pas de mise à jour automatique
`cd mon-projet` `claude`	Lancer dans un projet (connexion demandée au premier lancement)

Nécessite un abonnement Claude payant (Pro, Max, Team, Enterprise) ou un compte Anthropic Console. Autres surfaces : extension VS Code et Cursor, plugin JetBrains, application de bureau, claude.ai/code et application mobile.

Lancer une session

`claude`	Session interactive.
`claude "explique ce projet"`	Session interactive avec un premier prompt.
`claude -p "requête"`	Mode non interactif : répond puis quitte (scripts, CI).
`claude -c`	Reprend la dernière conversation du dossier.
`claude -r "nom"`	Reprend une session par ID ou par nom (sélecteur sans argument).
`claude -n "nom"`	Nomme la session.
`claude -w [nom]`	Démarre dans un git worktree isolé.
`claude --model opus`	Choisit le modèle : **`sonnet`**, **`opus`**, **`haiku`**, **`fable`** ou un ID complet.

Lancer une session (suite)

`claude --permission-mode plan`	Démarre dans un mode de permission (voir plus bas).
`claude --cloud "tâche"`	Crée une session dans le cloud.
`claude --teleport`	Rapatrie une session cloud dans le terminal.
`claude --chrome`	Active l'intégration avec Chrome.
`claude agents`	Vue des sessions d'arrière-plan : lancer, suivre, reprendre la main.
`claude update`	Met Claude Code à jour.
`claude doctor`	Diagnostic de l'installation et des réglages.

Scripts et CI

`tail -200 app.log | claude -p "signale les anomalies"`	Envoyer une entrée par pipe
`claude -p "liste les TODO du projet" --output-format json`	Sortie structurée
`claude -p "lance les tests" --permission-mode dontAsk --allowedTools "Bash(npm test)" "Read"`	CI verrouillée : seuls les outils listés sont autorisés
`claude -p "corrige le lint" --max-turns 10 --max-budget-usd 5`	Plafonds de sécurité
`claude --bare -p "résume ce diff"`	Démarrage minimal, sans hooks, skills, plugins, MCP ni CLAUDE.md

Commandes slash

Taper `/` pour afficher la liste, ou `/` suivi de quelques lettres pour filtrer.

Démarrer un projet

`/init`	Génère un **`CLAUDE.md`** de départ à partir du dépôt.
`/memory`	Édite les fichiers **`CLAUDE.md`** et gère la mémoire automatique.
`/mcp`	Gère les serveurs MCP et leur authentification.
`/permissions`	Règles allow, ask et deny par outil.
`/plugin`	Installe, active ou désactive des plugins.
`/skills`	Liste les skills ; **`t`** trie par coût en tokens.

Commandes slash (suite)

`/import` `codex`	Importe la configuration d'OpenAI Codex, de Gemini CLI ou de Cursor.

Pendant une tâche

`/plan` *`[tâche]`*	Passe en mode plan : Claude analyse et propose avant de modifier.
`/model`	Change de modèle.
`/effort` *`[low…max]`*	Règle le niveau de raisonnement.
`/fast`	Active ou désactive le mode rapide.
`/context`	Affiche le remplissage du contexte et les éléments les plus coûteux.
`/compact` *`[consignes]`*	Résume la conversation pour libérer du contexte.
`/btw` `question`	Question annexe qui n'entre pas dans l'historique.
`/goal` `condition`	Claude continue jusqu'à ce que la condition soit remplie.
`/loop` *`[intervalle]`* `prompt`	Répète un prompt tant que la session reste ouverte.
`/add-dir` `chemin`	Donne accès à un répertoire supplémentaire.

Travailler en parallèle

`/tasks`	Liste les travaux d'arrière-plan de la session (shells, sous-agents).
`/subtask` `tâche`	Lance un sous-agent qui hérite de toute la conversation.
`/background`	Détache la session pour qu'elle continue en arrière-plan.
`/fork`	Copie la conversation dans une nouvelle session d'arrière-plan.
`/batch` `instruction`	Découpe un gros changement en 5 à 30 unités exécutées dans des worktrees.
`/deep-research` `question`	Recherche web multi-sources avec rapport cité.
`/workflows`	Suit, met en pause ou reprend les dynamic workflows.

Commandes slash (suite)

Avant de livrer

`/diff`	Affiche les changements du working tree.
`/code-review [niveau] [--fix]`	Cherche les bugs du diff, d'une PR ou d'une branche (alias `/review`).
`/code-review ultra`	Revue multi-agents dans le cloud, plus lente et payante.
`/security-review`	Cherche les failles de sécurité de la branche.
`/simplify`	Quatre agents proposent réutilisation, simplification et optimisation, puis appliquent.
`/verify`	Construit et lance l'application pour constater que le changement fonctionne.
`/autofix-pr`	Session cloud qui corrige la PR quand la CI échoue ou qu'un relecteur commente.

Entre deux sessions

`/clear`	Nouvelle conversation ; la mémoire du projet est conservée.
`/resume`	Reprend une conversation passée.
`/branch`	Crée une branche de la conversation pour tester une autre piste.
`/rename`	Renomme la session.
`/teleport`	Rapatrie une session cloud.
`/remote-control`	Rend la session pilotable depuis claude.ai ou le téléphone.
`/desktop`	Continue la session dans l'application de bureau.
`/export`	Exporte la conversation en texte.

En cas de problème

`/rewind`	Revient à un checkpoint : code, conversation ou les deux (alias `/undo`).
`/doctor`	Bilan de santé de la configuration, avec corrections proposées.
`/debug`	Active les logs de debug et les analyse.

Commandes slash (suite)

`/usage`	Coût de la session, limites du forfait et statistiques (alias `/cost`).
`/status`	Version, modèle, compte et connectivité.
`/feedback`	Signale un bug avec le contexte de la session.

Raccourcis clavier

Raccourci	Effet
`Esc`	Interrompt Claude en gardant le travail déjà fait.
`Esc Esc`	Efface le brouillon ; sur une entrée vide, ouvre le menu de retour arrière.
`Shift+Tab`	Fait défiler les modes de permission.
`Ctrl+C`	Interrompt ou efface la saisie ; deux fois pour quitter.
`Ctrl+B`	Passe une commande ou un agent en arrière-plan.
`Ctrl+O`	Affiche la transcription détaillée des appels d'outils.
`Ctrl+T`	Affiche la liste de tâches de Claude.
`Ctrl+R`	Recherche dans l'historique des prompts.
`Ctrl+G`	Édite le prompt dans l'éditeur externe.
`Ctrl+S`	Met le prompt de côté, puis le restaure.
`Ctrl+V` (`Cmd+V` sous iTerm2, `Alt+V` sous Windows)	Colle une image.
`Option+P` / `Alt+P`	Change de modèle sans effacer le prompt.
`Option+T` / `Alt+T`	Active ou désactive le raisonnement étendu.
`Option+O` / `Alt+O`	Active ou désactive le mode rapide.
`\` + `Entrée`, `Ctrl+J` ou `Shift+Entrée`	Nouvelle ligne dans le prompt.
`Ctrl+L`	Redessine l'écran si l'affichage est corrompu.
`?` sur une entrée vide	Affiche l'aide des raccourcis.

Sur macOS, les raccourcis `Option` nécessitent d'activer « Option as Meta » dans le terminal.

Raccourcis clavier (suite)

Préfixes

Préfixe	Effet
`/`	Commande ou skill.
`!`	Mode shell : exécute la commande et Claude réagit à sa sortie.
`@`	Mentionne un fichier, ou une autre session Claude.

Modes de permission

À changer avec `Shift+Tab` ou `--permission-mode`.

Mode	Ce qui passe sans demander	Pour
`default` (Manual)	Les lectures seulement.	Travail sensible, code inconnu.
`acceptEdits`	Lectures, éditions et commandes de fichiers courantes.	Itérer en relisant.
`plan`	Lectures ; aucune édition avant validation du plan.	Explorer avant de modifier.
`auto`	Tout, avec un classifieur qui bloque les actions à risque.	Tâches longues. Mode par défaut sur Pro, Max et Team depuis le 14 août 2026.
`dontAsk`	Les outils pré-autorisés ; le reste est refusé.	CI et scripts verrouillés.
`bypassPermissions`	Tout.	Conteneur ou VM isolés uniquement.

Les règles `deny` s'appliquent dans tous les modes, y compris `bypassPermissions`.

Fichiers de configuration

Fichier	Rôle
`CLAUDE.md` ou `.claude/CLAUDE.md`	Instructions du projet, partagées via git.
`CLAUDE.local.md`	Préférences personnelles pour le projet (à ajouter au `.gitignore`).
`~/.claude/CLAUDE.md`	Instructions personnelles pour tous les projets.
`AGENTS.md`	Instructions communes aux agents de code, lues seules ou avec `CLAUDE.md`.

Fichiers de configuration (suite)

Fichier	Rôle
`.claude/settings.json`	Réglages partagés : permissions, hooks, variables d'environnement.
`.claude/settings.local.json`	Réglages personnels pour le projet.
`~/.claude/settings.json`	Réglages utilisateur.
`.claude/skills/<nom>/SKILL.md`	Skill de projet (`~/.claude/skills/` pour une skill personnelle).
`.claude/agents/<nom>.md`	Sous-agent personnalisé.
`.claude/rules/*.md`	Règles chargées selon les chemins (`paths`).
`.mcp.json`	Serveurs MCP partagés au niveau du projet.
`~/.claude/projects/<projet>/memory/`	Mémoire automatique écrite par Claude.

Ordre de priorité des réglages : managed settings > ligne de commande > `settings.local.json` > `settings.json` du projet > réglages utilisateur.

Serveurs MCP

`claude mcp add --transport http notion https://mcp.notion.com/mcp`	Serveur distant en HTTP
`claude mcp add --transport stdio db -- npx -y @bytebase/dbhub --dsn "postgresql://..."`	Serveur local en stdio (tout ce qui suit -- est la commande à lancer)
`claude mcp add --transport http sentry --scope project https://mcp.sentry.dev/mcp`	Partager avec l'équipe via .mcp.json

`claude mcp list`
`claude mcp get notion`
`claude mcp remove notion`
`claude mcp login notion`
	Gérer

Portées : `local` (par défaut, ce projet et toi seul), `project` (fichier `.mcp.json` versionné), `user` (tous tes projets).

Hooks

Exemple dans `.claude/settings.json` : formater chaque fichier édité et notifier quand Claude attend une réponse (macOS).

Hooks (suite)

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [{ "type": "command", "command": "jq -r
'.tool_input.file_path' | xargs npx prettier --write" }]
      }
    ],
    "Notification": [
      {
        "matcher": "",
        "hooks": [{ "type": "command", "command": "osascript -
e 'display notification \"Claude Code attend une réponse\" with
title \"Claude Code\"'" }]
      }
    ]
  }
}
```

Événement	Moment
`SessionStart` / `SessionEnd`	Début ou reprise, puis fin de session.
`UserPromptSubmit`	À l'envoi d'un prompt, avant son traitement.
`PreToolUse`	Avant un appel d'outil ; peut le bloquer.
`PostToolUse`	Après un appel d'outil réussi.
`Notification`	Quand Claude Code envoie une notification.
`Stop` / `SubagentStop`	Quand Claude ou un sous-agent termine.
`PreCompact`	Avant la compaction du contexte.

Conseils d'utilisation

- **Toujours donner un moyen de vérifier** : tests, script, capture d'écran attendue. C'est le premier conseil de la documentation officielle ; sans vérification, ne pas livrer.
- **Explorer, planifier, puis coder** : `/plan` (ou `Shift+Tab`) pour les changements importants, validation du plan, puis implémentation.
- `/clear` **entre deux tâches sans rapport** : un contexte encombré dégrade les réponses. Après deux corrections ratées, repartir de zéro avec un meilleur prompt.
- **Garder** `CLAUDE.md` **court** : un fichier trop long est en partie ignoré. Supprimer ce que Claude fait déjà bien, ou le transformer en hook.
- **Délimiter les investigations** ou les confier à un sous-agent, pour ne pas remplir le contexte principal de centaines de fichiers.
- **Les checkpoints ne remplacent pas git** : `/rewind` ne restaure pas les modifications faites par des commandes Bash, par les sous-agents ou hors de Claude Code.

Conseils d'utilisation (suite)

- **Surveiller la consommation** avec `/usage` et `/context` : les sessions et sous-agents en parallèle multiplient les tokens.
- **Réserver** `--dangerously-skip-permissions` **à un conteneur ou une VM**, et ne pas considérer le mode auto comme une garantie absolue.
- **Vérifier un plugin avant de l'installer** : il peut exécuter du code et des hooks sur la machine.
- **Mettre à jour régulièrement** si l'installation passe par Homebrew ou WinGet, qui ne se mettent pas à jour seuls.

Sources (documentation officielle)

- Documentation – référence des commandes
- Documentation – référence de la CLI
- Documentation – mode interactif et raccourcis
- Documentation – modes de permission
- Documentation – mémoire et CLAUDE.md
- Documentation – MCP
- Documentation – guide des hooks
- Documentation – bonnes pratiques