T03 # Modes de Permission

Technique Beginner

Contrôle des accès outils, du plus sûr au plus permissif

Modes Disponibles

Mode	Nom canonique	Activation	Usage recommandé
Default	`default`	(aucun)	Développement quotidien
Auto-accept edits	`acceptEdits`	`Shift+Tab`	Revues de code
Plan	`plan`	`Shift+Tab x2` ou `/plan`	Analyse sans modification
Don't Ask	`dontAsk`	`/permissions` ou règles `permissions.allow`	Workflows restrictifs, refus silencieux si non pré-approuvé
Auto (classifier IA)	`auto`	`permissions.defaultMode: "auto"`	Tâches longues, moins d'interruptions
Bypass total	`bypassPermissions`	`--dangerously-skip-permissions`	CI/CD headless, sandboxé
Moins de prompts	—	`/fewer-permission-prompts`	Génère un allowlist à partir des transcripts (livrée sous le nom `/less-permission-prompts` en v2.1.111)

Activation en CLI : `claude --permission-mode <mode>` accepte `default`, `plan`, `acceptEdits`, `bypassPermissions`. **Activation persistante :** clé `permissions.defaultMode` dans `settings.json`.

Note : `--dangerously-skip-permissions` ne prompte plus pour les écritures dans `.claude/skills/`, `.claude/agents/` et `.claude/commands/` (v2.1.121) ; le reste de `.claude/` (settings, hooks) et `.git/` restent protégés même en bypass total. Le mode `auto` s'appuie sur un modèle classifieur qui évalue chaque appel d'outil avant exécution : friction en moins, pas une frontière de sécurité.

Whitelist d'Outils

```
# Autoriser seulement certains outils
claude --allowedTools "Read,Grep,Glob"

# Bloquer des outils spécifiques
claude --disallowedTools "Bash,Write"

# Combinaisons utiles
claude --allowedTools "Read,Edit,Bash(git*)"
```

Configuration dans settings.json

```
{
  "permissions": {
    "allow": [
      "Bash(git log*)",
      "Bash(npm test*)",
      "Read",
      "Edit"
    ],
    "deny": [
      "Bash(rm*)",
      "Bash(sudo*)"
    ]
  }
}
```

Hiérarchie des Permissions

Les permissions se cumulent et s'héritent dans cet ordre :

1. `~/.claude/settings.json` : global user
2. `.claude/settings.json` : projet (partagé)
3. `.claude/settings.local.json` : projet (local, gitignored)
4. Flags CLI : session uniquement

Glob Patterns pour Bash

```
# Autoriser git seulement
"Bash(git *)"

# Autoriser npm test et build
"Bash(npm test*)",
"Bash(npm run build*)"

# Autoriser lecture fichiers
"Bash(cat *)",
"Bash(ls *)"
```

Bonnes Pratiques

CI/CD : toujours utiliser `--dangerously-skip-permissions` avec un environnement sandboxé (Docker, container éphémère). Ne jamais sur une machine de production partagée.

Projets sensibles : restreindre les outils Bash avec des globs précis dans `.claude/settings.json`. Committer ce fichier pour que toute l'équipe utilise les mêmes contraintes.

Audit : les actions de Claude sont loggées dans `~/.claude/logs/`. Vérifiable à tout moment.

Threat Intelligence

La base de données des menaces connues (v2.27.0) recense 118 CVEs et 93 skills malveillants identifiés. Parmi les vulnérabilités actives : CVE-2026-33032 (nginx-ui MCPwn, CVSS 9.8, exploitation active) et CVE-2026-30623 (LiteLLM, patchée). Vérifier régulièrement les outils MCP tiers avant intégration.