Référence rapide Claude Code

Commandes CLI, workflows, MCP, hooks, configuration

Démarrage

Installation et lancement

```
npm install -g @anthropic-ai/claude-code
claude            # démarrer une session interactive
claude --version  # vérifier la version
claude update     # mettre à jour vers la dernière version
```

Authentification

```
claude login             # OAuth via navigateur
export ANTHROPIC_API_KEY="sk-ant-..."
claude logout            # effacer la session
```

Commandes slash

Commandes de session

/help	Afficher les commandes disponibles
/clear	Effacer l'historique de conversation
/compact	Condenser le contexte pour économiser des tokens
/cost	Afficher l'utilisation de tokens et le coût
/status	Afficher les infos de session et le modèle
/new	Démarrer une nouvelle conversation
/config	Ouvrir ou consulter la configuration
/doctor	Diagnostiquer les problèmes de configuration
/init	Créer un CLAUDE.md pour le projet
/login	S'authentifier avec Anthropic
/logout	Effacer l'authentification

Commandes de workflow

/bug	Signaler un bug
/review	Demander une revue de code
/pr	Créer ou mettre à jour une pull request
/commit	Valider les changements indexés avec un message

Raccourcis clavier

Ctrl+C	Annuler la génération en cours
Ctrl+D	Quitter Claude Code (EOF)
Escape	Annuler la saisie / l'édition en cours
Tab	Autocomplétion des chemins et commandes
Haut/Bas	Naviguer dans l'historique des saisies

Options CLI

Options courantes

-p, --print	Mode non-interactif (headless)
--model	Définir le modèle : **opus**, **sonnet**, **haiku**
--output-format	Sortie en **text**, **json**, **stream-json**
--allowedTools	Restreindre les outils : **Edit,Read,Bash**
--max-turns N	Limiter le nombre de tours
--debug	Activer la journalisation de débogage
-n, --name	Nommer la session

Mode headless / scripting

```
claude -p "explain this error" < log.txt
claude -p "list todos" --output-format json
echo "fix typos" | claude -p
```

Fichiers de configuration

Hiérarchie CLAUDE.md

./CLAUDE.md	Instructions au niveau projet (dans le dépôt)
./.claude/settings.json	Paramètres projet (permissions, hooks)
~/.claude/CLAUDE.md	Instructions globales utilisateur (tous projets)
~/.claude/settings.json	Paramètres globaux utilisateur
~/.claude/projects/*/CLAUDE.md	Instructions utilisateur par projet (hors dépôt)

Variables d'environnement

ANTHROPIC_API_KEY	Clé API pour l'authentification directe
CLAUDE_MODEL	Modèle par défaut à utiliser
CLAUDE_CONFIG_DIR	Chemin du répertoire de configuration personnalisé

Permissions

settings.json

```
{
  "permissions": {
    "allow": ["Read", "Glob", "Grep"],
    "deny": ["Bash(rm *)"]
  }
}
```

Modes de permission

default	Demander avant les outils à risque
--dangerously-skip-permissions	Autoriser tous les outils (CI/scripts uniquement)
--allowedTools	Option CLI pour restreindre les outils

Serveurs MCP

Qu'est-ce que les serveurs MCP ?

Les serveurs Model Context Protocol étendent Claude Code avec des outils personnalisés — bases de données, API, services. Gérés via CLI ou `.mcp.json`.

Gestion CLI

```
claude mcp add server-name -- cmd arg1 arg2
claude mcp list
claude mcp remove server-name
```

Config .mcp.json

```
{
  "mcpServers": {
    "my-db": {
      "command": "python",
      "args": ["-m", "mcp_server_db"],
      "env": { "DB_URL": "${DATABASE_URL}" }
    }
  }
}
```

Portée

.mcp.json	Serveurs MCP au niveau projet
~/.claude/mcp.json	Serveurs MCP globaux utilisateur
claude mcp add -s user	Ajouter à la portée utilisateur
claude mcp add -s project	Ajouter à la portée projet

Hooks

Événements de hook

PreToolUse	S'exécute avant l'exécution d'un outil
PostToolUse	S'exécute après la fin d'un outil
Notification	S'exécute quand Claude envoie une notification
Stop	S'exécute quand Claude termine une réponse

Exemple settings.json

```
{
  "hooks": {
    "PreToolUse": [{
      "matcher": "Bash",
      "hooks": [{
        "type": "command",
        "command": "check-command.sh $INPUT"
      }]
    }]
  }
}
```

Comportement des hooks

exit 0	Autoriser l'exécution de l'outil
exit 2	Bloquer l'exécution de l'outil
stdout	Retour JSON à Claude

SDK et automatisation

Scripting headless

```
# Prompt unique, sortie JSON
claude -p "summarize main.py" \
  --output-format json

# Entrée via pipe
git diff | claude -p "review this diff"
```

CI / GitHub Actions

```
- uses: anthropics/claude-code-action@v1
  with:
    claude_args: >
      --max-turns 5
      --model claude-sonnet-4-6
```

Conseils

Utiliser `--max-turns` pour limiter les coûts en automatisation. Utiliser `--output-format json` pour analyser les résultats programmatiquement. Combiner avec `--allowedTools` pour la sécurité.

Modèles

Sélection du modèle

```
claude --model opus     # le plus performant
claude --model sonnet   # équilibré (par défaut)
claude --model haiku    # le plus rapide et économique
```

Raccourcis de modèles

opus	Claude Opus — capacité maximale
sonnet	Claude Sonnet — par défaut, équilibré
haiku	Claude Haiku — rapide et économique
--model full-name	ex. **claude-sonnet-4-6**

Bonnes pratiques

Rédiger CLAUDE.md

Mettre les conventions du projet, la stack technique, les commandes de build et les instructions de test dans CLAUDE.md. Rester concis — Claude le lit à chaque session. Utiliser `/init` pour le créer.

Gestion des coûts

/cost	Vérifier l'utilisation de tokens en cours
/compact	Compresser le contexte quand il devient grand
--max-turns	Limiter les tours en automatisation
sonnet / haiku	Utiliser des modèles moins chers pour les tâches simples

Prompting efficace

Être précis	"Corriger la vérification null dans auth.ts:42" > "corriger le bug"
Donner du contexte	Référencer des fichiers, fonctions, messages d'erreur
Utiliser @fichier	Référencer directement : **@src/main.ts**
Utiliser des images	Glisser-déposer des captures d'écran pour le contexte visuel
Itérer	Affiner avec des suivis ; utiliser **/clear** pour réinitialiser