Jumeau numérique · Sous-section

Utiliser Claude Code dans son coffre

Alternative à OpenCode Desktop pour les utilisateurs Claude Max : différence entre Claude Desktop (app) et Claude Code (CLI terminal), où sont configurés les MCP user-scope dans chacun, et comment partager les skills du coffre via un symlink (.claude/skills → Second Cerveau/4 TOOLS/skills).

Source : https://prisme.one/plateforme/ia-juillet-2026/semaine-1-claude-code-coffre

---

Utiliser Claude Code dans son coffre

LOOM

Utiliser Claude Code dans son coffre

https://www.loom.com/share/6978abe2509a4c588afa4ef1756fdd73

> **Cette leçon s'adresse aux utilisateurs de Claude Code.** OpenCode Desktop reste la stack recommandée par défaut dans ce bootcamp. Mais si tu as un abonnement Claude Max et que tu veux travailler dans Claude Code (CLI terminal ou Claude Desktop), cette leçon te montre comment l'intégrer proprement avec ton coffre.

Deux Claude Code, deux configurations

Anthropic propose deux façons d'utiliser Claude Code :

Claude Desktop (app GUI)
→ App graphique (macOS / Windows)
→ Chat classique avec interface visuelle
→ Idéal pour les discussions sans manipulation de fichiers lourde
→ MCP configurés dans un fichier JSON
Claude Code (CLI terminal)
→ Outil en ligne de commande, lancé dans un dossier

→ Voit tous les fichiers du dossier, peut les éditer
→ Idéal pour travailler dans ton coffre, comme OpenCode
→ MCP configurables via commande ou fichier JSON

Tu peux utiliser les deux, mais elles ne partagent pas leur configuration : tu vas devoir installer tes MCP **deux fois** si tu veux les retrouver dans les deux interfaces.

Dans le contexte de ce bootcamp (travailler dans ton coffre avec un agent qui voit tes fichiers), c'est **Claude Code (CLI terminal)** qui est l'équivalent direct d'OpenCode Desktop.

Ouvrir Claude Code dans ton coffre

Si Claude Code n'est pas encore installé, suis la page d'installation officielle d'Anthropic (méthodes natives recommandées : Homebrew sur macOS, WinGet sur Windows, apt/dnf/apk sur Linux). Une fois installé, pour l'utiliser dans ton système :

```
# Va dans ton Workspace racine
cd ~/Documents/WORKSPACE

# Lance Claude Code
claude
```

Claude Code démarre, lit l'`AGENTS.md` ou `CLAUDE.md` à la racine, remonte les parents s'il y en a, et a accès à tous les fichiers du dossier (Second Cerveau + autres projets).

> **Astuce :** crée un alias dans ton `.zshrc` ou `.bashrc` :
>
> ```
> alias cco="cd ~/Documents/WORKSPACE && claude"
> ```
>
> Tu tapes `cco` n'importe où, tu atterris dans ton coffre avec Claude Code lancé.

Où sont rangés les MCP au niveau user

Pour qu'un MCP soit dispo dans **toutes tes sessions** (peu importe le dossier où tu lances Claude Code), tu dois le configurer au **scope user** (ce que les anciennes versions appelaient

« global »).

Pour Claude Code (CLI terminal)

Deux options :

Option 1 : la commande `claude mcp add` (la plus simple)

```
claude mcp add --scope user firecrawl npx -y firecrawl-mcp
claude mcp add --scope user playwright npx -y
@playwright/mcp@latest
```

Ces MCP sont disponibles partout, dans n'importe quel dossier où tu lances `claude`.

Option 2 : éditer le fichier de config user

Le fichier de config global de Claude Code CLI est `~/.claude.json` (dans ton home, le même pour toutes tes sessions). Il contient ton OAuth, tes MCP au **scope user** (dispos partout) **et** au **scope local** (privés à un projet précis, indexés par chemin), plus l'état de chaque projet. Tu peux y ajouter une section MCP à la main, mais la commande `claude mcp add` est plus propre : elle écrit au bon scope, fait des backups automatiques, et garde les 5 derniers backups.

> **Attention** : `~/.claude.json` ≠ `~/.claude/`. Le **fichier** `.json` héberge la config MCP et l'état. Le **dossier** `.claude/` (sans extension) héberge tes skills, agents et commandes user-level (cf. section suivante).

Pour Claude Desktop (app GUI)

Pas de commande dédiée. Tu édites directement le fichier JSON :

- **macOS** : `~/Library/Application Support/Claude/claude_desktop_config.json`
- **Windows** : `%APPDATA%\Claude\claude_desktop_config.json`

Format :

```
{
  "mcpServers": {
```

```
    "firecrawl": {
      "command": "npx",
      "args": ["-y", "firecrawl-mcp"],
      "env": {
        "FIRECRAWL_API_KEY": "fc_xxxxx"
      }
    },
    "playwright": {
      "command": "npx",
      "args": ["-y", "@playwright/mcp@latest"]
    }
  }
}
```

> **Important :** après avoir édité ce fichier, **quitte complètement Claude Desktop** (Cmd+Q sur macOS, pas juste fermer la fenêtre) et relance. L'app ne recharge pas la config à chaud.
>
> Pour debug : les logs sont dans `~/Library/Logs/Claude/`. Le fichier `mcp.log` donne les connexions, et `mcp-server-<NOM>.log` donne les erreurs spécifiques à chaque serveur.

Différence clé à retenir

	Claude Code (CLI)	Claude Desktop (app)
Config MCP user	`~/.claude.json` ou `claude mcp add --scope user`	`~/Library/Application Support/Claude/claude_desktop_config.j`
Config MCP projet	`.mcp.json` à la racine du projet	Pas de scope projet, tout est global
Skills	`~/.claude/skills/` (user) ou `.claude/skills/` (projet)	Lus depuis `~/.claude/skills/` également

	Claude Code (CLI)	Claude Desktop (app)
Rechargement config	À chaque `claude` (auto)	Restart complet de l'app obligatoire

---

Skills dans Claude Code : pas de `skills.paths` déclaratif

C'est **la grosse différence** avec OpenCode. Dans OpenCode, tu listes des dossiers de skills dans `opencode.json` :

```
{
  "skills": {
    "paths": [
      "./Second Cerveau/4 TOOLS/skills",
      "./skills-equipe"
    ]
  }
}
```

Et OpenCode charge tous les skills des chemins listés. **Claude Code ne supporte pas ça.** Il lit les skills **uniquement** dans deux endroits :

- **`~/.claude/skills/`** : skills **user / globaux** (perso, dispos dans toutes tes sessions, peu importe le dossier où tu lances `claude`)
- **`.claude/skills/`** : skills **projet** (à la racine du dossier où tu lances `claude`, non visibles ailleurs)

Deux détails utiles :

- **Hot reload** : ajouter, éditer ou supprimer un skill prend effet dans la session courante, pas besoin de redémarrer Claude Code.
- **Conflit de nom** : si un skill du même nom existe à plusieurs niveaux, l'ordre de priorité est `enterprise > user > project`.

Si tes skills vivent dans `Second Cerveau/4 TOOLS/skills/` (la convention de ce bootcamp), Claude Code ne les voit pas par défaut.

La solution : symlink

Tu crées un **symlink** (Mac/Linux) ou **junction** (Windows) entre `.claude/skills` et le dossier réel de tes skills dans le coffre.

Mac / Linux :

```
# À la racine de ton Workspace (là où tu lances claude)
cd ~/Documents/WORKSPACE

# Crée le dossier .claude s'il n'existe pas
mkdir -p .claude

# Symlink : .claude/skills pointe vers le vrai dossier
# Important : le path source doit être relatif depuis
l'emplacement du symlink,
# donc on entre d'abord dans .claude puis on remonte d'un cran.
cd .claude && ln -s "../Second Cerveau/4 TOOLS/skills" skills &&
cd ..
```

> **Si ta structure de coffre est custom** (par exemple tu as renommé "Second Cerveau" en "REFLECTIA" ou utilisé une arbo différente d'IPCRA) : plutôt que de te battre avec les symlinks, demande à Claude Code de faire une passe sur l'ensemble du dossier pour corriger tous les chemins, et stocke tes skills directement dans `.claude/skills/`. C'est plus simple, ça évite les couches indirectes, et tu économises les tokens consommés par grep + résolution de symlink à chaque session.

Windows (junction, sans droits admin) :

```
cd "C:\Users\Toi\Documents\WORKSPACE"
mkdir .claude
mklink /J ".claude\skills" "Second Cerveau\4 TOOLS\skills"
```

Vérifie :

```
ls -la .claude/skills/
# Doit afficher le contenu de Second Cerveau/4 TOOLS/skills/
```

À partir de là, Claude Code voit tous tes skills (`/initialisation`, `/import`, `/weekly-review`, `/done`...) comme s'ils étaient nativement dans `.claude/skills/`. Tu tapes `/` dans Claude Code, tu vois la liste complète.

Variante : skills dispos partout via `~/.claude/skills`

Si tu veux que tes skills coffre soient dispos **dans tous les projets** (pas seulement dans le Workspace), fais le symlink depuis `~/.claude/skills` à la place :

```
# Attention : si tu as déjà des skills perso dans
~/.claude/skills, sauvegarde-les avant
ln -s ~/Documents/WORKSPACE/Second\ Cerveau/4\ TOOLS/skills
~/.claude/skills
```

Trade-off : tu n'as plus de dossier user séparé pour des skills uniquement perso. Pour la plupart des cas, le symlink projet (`.claude/skills` dans le Workspace) suffit largement.

Récap : lancer Claude Code dans ton coffre proprement

1.
Installer Claude Code
Page d'installation officielle
2.
Symlink des skills
`ln -s "Second Cerveau/4 TOOLS/skills" .claude/skills`
3.
Ajouter tes MCP user-scope
`claude mcp add --scope user firecrawl ...`
4.
Lancer Claude Code dans le Workspace
`cd ~/Documents/WORKSPACE && claude`

- [ ] Je connais la différence entre Claude Desktop (app GUI) et Claude Code (CLI terminal)
- [ ] Claude Code installé via la page de doc officielle (https://docs.claude.com/en/docs/claude-code/setup)
- [ ] Symlink .claude/skills → Second Cerveau/4 TOOLS/skills créé à la racine du Workspace

- [ ] Je sais où sont configurés les MCP user-scope dans Claude Code (~/.claude.json ou claude mcp add --scope user)
- [ ] Je sais où sont configurés les MCP dans Claude Desktop (~/Library/Application Support/Claude/claude_desktop_config.json sur macOS)
- [ ] Si j'utilise Claude Desktop : je sais qu'il faut quitter complètement (Cmd+Q) et relancer après chaque modif du JSON
- [ ] Claude Code lancé depuis mon Workspace : je vois mes skills via /

Sources

- Set up Claude Code (page d'installation officielle)
- Connect Claude Code to tools via MCP
- Extend Claude with skills
- Claude Code settings
- Connect to local MCP servers (Claude Desktop)

Vidéos

LOOM

Utiliser Claude Code dans son coffre

https://www.loom.com/share/6978abe2509a4c588afa4ef1756fdd73

Extraction verbatim depuis Prisme One · https://prisme.one/plateforme/ia-juillet-2026/semaine-1-claude-code-coffre