Jumeau numérique · Sous-section

Comment utiliser OpenCode Desktop

Tout pour devenir autonome sur OpenCode Desktop en 20 minutes : brancher tes modèles et tes outils, comprendre l'interface (panneaux, raccourcis, modes build et plan) et ouvrir l'IA au bon endroit pour chaque type de travail.

Objectif

Exemples : OpenAI, Anthropic, OpenRouter, Fireworks, DeepInfra, Infomaniak.

Output attendu

Exemples : GPT 5.6, Fable / Claude Opus 4.8, DeepSeek V4 Flash, GLM 5.2, Qwen 3.6.

Comment utiliser OpenCode Desktop

Dans cette leçon, je t'explique comment configurer OpenCode Desktop pour qu'il soit parfaitement opérationnel, puis comment l'utiliser au quotidien.

Comment utiliser OpenCode Desktop

La formation est en cours de construction

Vidéo à venir

Walkthrough complet : ce qu'est OpenCode, configurer fournisseurs et modèles, connecter des outils, comprendre l'interface, modes plan/build, ouvrir l'IA au bon dossier.
Note importante : "Prisme Workspace", c'est OpenCode Desktop

Dans la vidéo, je parle de **"Prisme Workspace"**. C'est un **abus de langage** : Prisme Workspace n'existe plus (en tout cas pas pour le moment, et pas sous cette forme). L'outil qu'on te recommande d'utiliser, c'est tout simplement **OpenCode Desktop**.

Donc à chaque fois que tu m'entends dire "Prisme Workspace" dans la vidéo, remplace mentalement par **OpenCode Desktop** : c'est le même outil, la même interface, la même configuration. Tout ce que je montre marche exactement pareil.

OpenCode, c'est un **harnais agentique open-source** : tu y ouvres un LLM (un modèle qui prédit le token suivant) et le harnais ajoute autour tout ce qui transforme un simple chatbot en agent (coder, appeler des outils, déléguer à des sous-agents, se planifier, s'auto-évaluer, respecter des permissions). C'est pour ça qu'on l'utilise plutôt qu'un chatbot question-réponse : la boucle agentique fait le travail, pas juste la réponse. C'est aussi un outil **ouvert** : tu y branches n'importe quel modèle (ChatGPT, Claude, DeepSeek, GLM, Qwen...) sans dépendre d'un seul fournisseur, et l'interface desktop reste très simple à prendre en main.

Étape 1 : Configurer tes fournisseurs et tes modèles

C'est la première chose à faire en arrivant. Clique sur la **roue crantée** en bas à gauche, puis va dans la section **Fournisseurs** et **Modèles**.

Fournisseur vs modèle, c'est quoi la différence ?

Fournisseur (provider)
L'infrastructure qui fait tourner l'IA

Ce sont les serveurs sur lesquels tourne le modèle. Chaque fournisseur a ses propriétés : prix, vitesse, confidentialité, infrastructure géographique.

Exemples : OpenAI, Anthropic, OpenRouter, Fireworks, DeepInfra, Infomaniak.

Modèle
Le cerveau, exposé par le fournisseur

Chaque fournisseur propose un catalogue de modèles différents. Certains sont propriétaires (GPT, Claude), d'autres open source (DeepSeek, Qwen, GLM, Kimi, Llama, Gemma).

Exemples : GPT 5.6, Fable / Claude Opus 4.8, DeepSeek V4 Flash, GLM 5.2, Qwen 3.6.

Pour résumer : **tu choisis ton fournisseur en fonction de ses propriétés (prix, confidentialité, vitesse), puis tu actives chez ce fournisseur les modèles que tu veux**

utiliser. Par exemple, Fireworks est un fournisseur qui propose uniquement des modèles open source très optimisés pour tourner vite.

Les deux modes de connexion

La plupart du temps
Connexion par clé API

Tu vas sur le site du fournisseur, tu te connectes, tu génères une **clé API** dans les paramètres de ta plateforme, et tu la colles dans OpenCode Desktop.

Concerne : OpenRouter, Fireworks, Infomaniak, DeepInfra, Vercel, Anthropic, et la grande majorité des fournisseurs.

Cas plus rares
Connexion par OAuth (compte)

Tu te connectes directement avec ton compte existant via une fenêtre du navigateur. Pas de clé API à gérer, et tu utilises ton **abonnement** au lieu de payer à la consommation.

Concerne : GitHub Copilot, OpenAI (ChatGPT Plus / Pro), Google dans certains cas.

Tutoriel : connecter OpenAI via abonnement ChatGPT

Stop : lis ça avant de connecter OpenAI
Si tu as déjà ChatGPT Plus ou Pro, NE mets PAS ta clé API

C'est **l'erreur n°1 qui coûte de l'argent** à ceux qui débutent. La **clé API OpenAI** facture **chaque token** que tu consommes, et ça chiffre très vite. Ton abonnement ChatGPT, lui, est **déjà payé** : tu peux l'utiliser dans OpenCode Desktop **sans payer un centime de plus**.

Beaucoup de gens croient que "c'est impossible" d'utiliser son abonnement ChatGPT dans un outil externe. **C'est faux.** Il suffit de choisir **"se connecter avec un abonnement existant"** (connexion OAuth / login OpenAI) au lieu de coller une clé API. C'est exactement ce qu'on fait juste en dessous.

Piège fréquent avec OpenRouter

Dans OpenRouter, **décoche TOUS les modèles GPT**. Sinon tu risques de payer GPT une **deuxième fois**, au token, via OpenRouter, alors que tu l'as déjà via ton abonnement. GPT ne doit être coché **que** dans le fournisseur OpenAI connecté avec ton abonnement.

Quand tu cliques pour ajouter OpenAI, OpenCode Desktop te demande :

1. **Veux-tu mettre une clé API ?** (paiement à la consommation, donc plus cher si tu utilises beaucoup) ou
2. **Veux-tu te connecter avec un abonnement existant ?** (recommandé si tu as déjà ChatGPT Plus ou Pro)

Choisis l'option 2. Une page s'ouvre dans ton navigateur, tu te connectes, tu cliques sur **Continue**, et c'est plié. OpenAI est connecté, et tu peux maintenant sélectionner les modèles que tu veux activer.

À savoir sur les abonnements

- Le **mode Pro de GPT 5.6** n'est **pas exposé** via le login ChatGPT dans OpenCode : tu auras **GPT 5.6** et **GPT 5.6 Fast**, largement suffisants pour la plupart des tâches.
- **Anthropic** n'autorise plus de connexion par abonnement Claude Pro ou Max dans des apps tierces. Il faut une clé API, donc tu payes à la consommation, ce qui peut vite chiffrer. C'est l'exception. Si tu veux Claude via abonnement, passe par l'app Claude Code en CLI (voir la leçon dédiée).

La barre du bas et le sélecteur de modèles

Une fois tes fournisseurs configurés, tu retrouves tout en bas de ton écran :

- Le **sélecteur de modèles** : tous les modèles activés sont listés là.
- Le bouton **Plus** : te redirige vers la fenêtre "Connecter un fournisseur".
- Le bouton **Paramètres** : ouvre l'écran de configuration par fournisseur, où tu coches/décoches quels modèles sont visibles dans le sélecteur.

Astuce : **décoche les modèles que tu n'utilises jamais**, sinon ta barre devient un fouillis. Par exemple, sur OpenRouter, il y a des centaines de modèles disponibles. Garde-en 5 à 10 pertinents et désactive le reste pour avoir une interface propre.

Étape 2 : Connecter tes outils (MCP)

Sous le capot, OpenCode Desktop fait tourner OpenCode. Donc connecter des outils à OpenCode Desktop, c'est en réalité connecter des outils à OpenCode. La bonne nouvelle : tu as déjà tout ce qu'il faut dans ton coffre du bootcamp.

La méthode recommandée : `/connect-mcp`

Dans OpenCode Desktop, tape simplement :

```
/connect-mcp
```

Puis le nom de l'outil que tu veux ajouter (par exemple `notion`, `firecrawl`, `playwright`, `microsoft 365`). L'IA te guide étape par étape : elle identifie le bon package, récupère les variables d'environnement nécessaires, patche ta configuration sans rien casser de ce qui existe déjà.

Par défaut, le MCP est ajouté à la **configuration OpenCode globale** de ton ordinateur. Cela veut dire que **tous tes workspaces** ont accès à l'outil. Si tu changes de workspace (par exemple, du workspace démo à ton workspace principal), tu retrouves les mêmes MCP disponibles.

MCP et contexte : active-les seulement quand tu en as besoin

Un MCP chargé au démarrage rajoute facilement 15 000 à 20 000 tokens de contexte à chaque conversation : ça te coûte de l'argent et ça remplit le contexte plus vite. La bonne pratique : démarre tes conversations avec **zéro MCP activé**, et active-les uniquement pour la tâche qui en a besoin.

Quand c'est possible, privilégie la version **CLI** d'un outil (ligne de commande) plutôt que sa version MCP : c'est plus rapide, plus déterministe, et ça consomme beaucoup moins de tokens. La plupart des outils que tu connectes en MCP ont un équivalent CLI.

Les trois sections de Paramètres

Serveurs MCP
Liste des MCP connectés. Tu n'as pas à toucher cette section directement, `/connect-mcp` s'en occupe.
Serveurs (non-MCP)
Tu n'as pas besoin de t'en occuper pour le moment.
Plugins OpenCode
Si tu vois un plugin OpenCode intéressant (par exemple "Oh My OpenCode"), tu peux demander à l'IA "installe-moi ce plugin OpenCode" et il apparaîtra directement dans cette section.

Étape 3 : Comprendre l'interface

La barre latérale (icône crayon)

Quand tu cliques sur le **crayon** en haut à droite, une barre latérale s'ouvre. Elle te donne plusieurs vues utiles :

Vue Statistiques
Comprendre ce qui se passe dans ta session

- Nombre de tokens en entrée et en sortie
- Modèle utilisé pour chaque message
- Voir le **JSON exact** envoyé à l'IA (très utile pour comprendre/déboguer)

Vue Console
Suivre l'utilisation du contexte

- Pourcentage d'appels d'outils (par exemple "67 % d'appels d'outils")
- Limite de contexte et où tu en es
- Utilisation du contexte par message

L'éditeur de notes intégré

Toujours dans la barre latérale, tu peux ouvrir le **panneau Documents** et éditer une note directement. C'est un éditeur Markdown léger, parfait pour travailler sur le même fichier que l'IA dans la même interface.

- Tu peux ouvrir **plusieurs notes en même temps** et les empiler sur la droite de ton écran.
- Syntaxe Markdown : barre en haut pour les commandes courantes, ou directement avec les caractères (`###` pour un titre de niveau 3, `**texte**` pour gras, etc.).

Étape 4 : Ouvrir l'IA dans le bon dossier

C'est le principe le plus important. **L'IA voit ce qui est dans le dossier où tu l'ouvres.** Rien d'autre.

Si tu suis le bootcamp
Ouvre ton Workspace complet

Le dossier `Workspace` (avec ton `Second Cerveau`, ton `AGENTS.md`, tes skills) devient le projet global d'OpenCode Desktop. L'IA a accès à tout ton contexte et à tous tes projets séquencés à l'intérieur.

Pour un projet isolé

Ouvre uniquement le dossier du projet

Tu travailles sur une présentation, un site, une analyse client ? Crée un dossier dédié, mets-y tes notes et tes fichiers, ouvre OpenCode Desktop à l'intérieur. L'IA n'a que ce contexte, plus rapide et plus précis.

Attacher un fichier ou pointer avec `@`

Tu peux enrichir ton contexte de deux manières en plein milieu d'une conversation :

- **Attacher un fichier** : clique sur l'icône trombone, sélectionne un fichier sur ton ordinateur.
- **Arobase (`@`)** : tape `@` puis le début du nom du fichier (par exemple `@weekly` ou `@embracing`), et tu obtiens une recherche rapide dans ton coffre. Tu sélectionnes la note et elle est référencée. **Excellent réflexe** pour pointer vers une note précise sans avoir à la copier-coller.

Étape 5 : Niveau d'effort et modes (build / plan)

Niveau d'effort

Selon le modèle, tu as parfois une option **niveau d'effort** (thinking on/off). Plus le niveau est élevé, plus le modèle va prendre le temps de réfléchir avant de répondre. Certains modèles te le proposent, d'autres sont en non-thinking par défaut (réponse directe, pas de phase de réflexion).

Build mode vs Plan mode

Tu as deux modes principaux dans le sélecteur en bas de la barre de prompt :

Plan mode
L'IA réfléchit, lit les fichiers, propose un plan structuré. Elle ne touche à rien tant que tu n'as pas validé.
Build mode
L'IA exécute le plan validé : écrit, édite, crée. À utiliser après le plan mode pour le travail concret.

Tu peux aussi **ajouter des modes custom** selon tes usages. Tout cela est détaillé dans la prochaine leçon "Bien utiliser l'IA".

Étape 6 : Recherche rapide avec `Cmd+K`

Le raccourci magique : **`Cmd+K`** ouvre une barre de recherche universelle. Tu peux chercher :

- **Des fichiers** : par exemple `meeting review skill` pour ouvrir la note correspondante.
- **Des sessions** : revenir à une conversation précédente.
- **Des commandes** : lancer une commande (skill) sans avoir à la taper.

C'est l'équivalent du Spotlight du Mac, mais pour ton workspace. À utiliser tout le temps.

Si tu rencontres un problème

Si tu rencontres un bug ou un comportement étrange dans OpenCode Desktop :

1. **Reporter sur GitHub** : ouvre une issue sur le repo public d'OpenCode.
2. **Nous écrire un email** à `contact@perspectives.ac` avec une description et idéalement un screenshot.

On corrige vite, et chaque retour participant améliore l'app pour tout le monde.

- [ ] Au moins 1 fournisseur configuré (clé API ou OAuth)
- [ ] Modèles que j'utilise vraiment cochés dans le sélecteur (les autres décochés)
- [ ] Au moins 1 MCP connecté via /connect-mcp (Firecrawl ou autre)
- [ ] Je sais ouvrir la barre latérale (crayon) pour voir mes statistiques de contexte
- [ ] Je sais utiliser @ pour pointer un fichier de mon coffre dans le prompt
- [ ] Je connais Cmd+K pour la recherche rapide
- [ ] Je sais différencier Plan mode (réfléchir) et Build mode (exécuter)