Jumeau numérique · Sous-section

# Zoom sur la sécurité

Comprendre à quoi l'IA a accès sur ton ordinateur, activer le ZDR sur OpenRouter, désactiver l'entraînement sur ChatGPT, restreindre les permissions dans opencode.json, et gérer ses API keys avec Bitwarden.

Source : https://prisme.one/plateforme/ia-juillet-2026/semaine-1-securite

---

## Zoom sur la sécurité et la privacy

Travailler avec l'IA dans tes fichiers locaux, ça soulève deux questions : qu'est-ce qu'elle peut faire sur ton ordinateur, et qu'est-ce qui sort vers le provider quand tu lui parles. Cette leçon répond aux deux, dans cet ordre.

---

## 1. L'IA n'a pas accès à tout par défaut

C'est le premier point à comprendre. Quand tu lances OpenCode Desktop, **l'IA ne peut rien faire sans ta validation**. Elle a la configuration de base d'OpenCode, qui te demande explicitement avant chaque action :

- Lire ce fichier : tu valides ?
- Modifier cette note : tu valides ?
- Exécuter cette commande bash : tu valides ?

C'est volontaire. Tu gardes le contrôle de tout, mais ça peut vite devenir lourd au quotidien.

Mode strict
Validation au cas par cas
L'IA demande la permission avant chaque action. Maximum de sécurité, friction quotidienne forte.
Mode équilibré
Permissions ciblées

Tu autorises les actions sûres (lecture, écriture dans ton coffre), tu bloques les risquées (rm, curl). Équilibre fluide.
Mode auto
Auto-accept tout
L'IA peut tout faire sans demander. Fluide pour le travail intensif, à éviter sur projets clients sensibles.

## Le bouton "accepter automatiquement"

Dans les réglages d'OpenCode Desktop, tu peux activer une option qui dit à l'IA : « tu peux faire ce que tu veux sans me demander, je te fais confiance ». Pratique quand tu travailles à fond. Tu peux la désactiver à tout moment.

> **Compromis classique :** confiance maximale (fluide mais l'IA peut tout faire) vs validation au cas par cas (sécurisé mais lent). La plupart des participants finissent par activer l'auto-accept dans leur coffre, et désactivent quand ils travaillent sur des projets clients sensibles.

## Restreindre finement les permissions

Si tu veux quelque chose entre les deux extrêmes (auto-accept ou validation totale), tu peux configurer **précisément** ce que l'IA peut faire dans le fichier `opencode.json` à la racine de ton workspace.

C'est du JSON. Tu peux l'écrire toi-même en lisant la documentation officielle d'OpenCode sur les permissions. Ou, mieux : tu demandes à l'IA de le faire pour toi.

## Le mini-prompt pour configurer tes permissions sans toucher au JSON

Plutôt que d'apprendre la syntaxe `opencode.json`, copie-colle ce prompt à ton IA dans OpenCode :

Prompt à coller dans OpenCode

```
Je veux configurer tes permissions dans opencode.json.

D'abord, lis la documentation officielle ici :
https://opencode.ai/docs/permissions
```

```
Ensuite, pose-moi ces questions une par une, attends ma réponse
(oui / non / précise) avant de passer à la suivante :

1. Tu peux lire les fichiers dans Second Cerveau/ sans me demander
?
2. Tu peux écrire dans Second Cerveau/ sans me demander ?
3. Tu peux accéder aux autres dossiers de mon Workspace ?
4. Tu peux exécuter des commandes shell (git, ls, cat) sans me
demander ?
5. Tu peux exécuter des commandes réseau (curl, wget) sans me
demander ?
6. Tu peux exécuter des commandes destructrices (rm, mv, rmdir)
sans me demander ?

À la fin, mets à jour opencode.json avec mes choix, montre-moi le
résultat,
et explique-moi en une phrase ce que ça change.
```

L'IA va lire la doc à jour, te poser les questions dans ton langage, et écrire le JSON pour toi. Tu valides à la fin. Tu n'as jamais à toucher au JSON manuellement.

---

# 2. Ce qui est envoyé au modèle (et ce qui ne l'est pas)

C'est le deuxième point essentiel : **quand tu poses une question à ton IA, qu'est-ce qui sort de ton ordinateur exactement ?**

La réponse va te rassurer.

## Ton coffre n'est pas uploadé en bloc

À chaque requête, l'IA ne « charge » pas tout ton coffre en mémoire pour ensuite l'envoyer au provider. Voici ce qui se passe vraiment :

Sur ton ordinateur
Tout ton coffre reste local

1. L'IA **scanne** les noms de fichiers et la structure
2. Elle **identifie** les fichiers pertinents pour ta question
3. Elle **lit** uniquement ceux-là (3, 5, parfois 20 fichiers)
4. Elle **écrit** les modifications directement sur ton disque

Envoyé au provider IA
Uniquement des extraits

1. Ta **question** (le prompt)
2. Le **contenu** des fichiers que l'IA a lus
3. L'historique de la conversation en cours
4. Rien d'autre. Pas de scan global, pas d'upload du coffre

**Image mentale :** l'IA ouvre 3 onglets de notes dans Obsidian, lit ces 3 notes, et c'est tout ce qui sort. Le reste du coffre reste invisible pour le provider.

### Conséquence pratique

Tu **n'as pas besoin de deux coffres séparés** (un « sécurisé » et un « pas sécurisé »). L'IA ne va lire que ce dont elle a besoin pour la requête en cours.

Et tu peux utiliser **différents modèles selon la sensibilité** de chaque tâche, dans le même coffre. Par exemple :

- Tâche perso, brainstorm : DeepSeek V4 Flash sur OpenRouter ZDR
- Tâche avec données clients : GLM 5.2 via Synthetic AI (abonnement ZDR)
- Tâche créative sans enjeu : ChatGPT (GPT 5.6)

Tu bascules dans OpenCode Desktop en une seconde. Le coffre reste le même.

---

## 3. Pourquoi la confidentialité, c'est important

Si tu te dis « j'ai rien à cacher, ça me préoccupe pas », regarde cette vidéo. Elle explique en détail ce qui se passe vraiment avec tes données quand tu parles à une IA, et pourquoi c'est bien plus large qu'un simple problème individuel.

Où vont vraiment vos données quand vous parlez à l'IA

YOUTUBE
**Où vont vraiment vos données quand vous parlez à l'IA**
https://www.youtube.com/watch?v=uljp0Bx9mWg

40 minutes. Le contexte macro : Snowden, Cloud Act, marché de la prédiction, scénarios post-AGI.

Les points clés en 1 minute :

- **Snowden, 2013** : les agences de renseignement américaines (NSA Prism, programme XKeyscore) ont un accès direct aux serveurs de Google, Facebook, Apple, Microsoft. La devise interne : « on enregistre tout, on filtrera plus tard si besoin ». Le programme existe toujours, en plus puissant.
- **Cloud Act, 2018** : toute entreprise US (peu importe où sont les serveurs) doit livrer les données au gouvernement américain sur demande. Donc même un data center « européen » d'AWS ou Azure est concerné.
- **Politiques de rétention en pratique** : ChatGPT conserve par défaut tant que ton compte existe. Claude conserve 5 ans + 30 jours après suppression + jusqu'à 2 ans pour modération « trust & safety » humaine. La suppression vraiment effective est rarement immédiate.
- **Le marché de la prédiction** : ton profil comportemental vaut de l'argent. C'est ce que documente Shoshana Zuboff dans *The Age of Surveillance Capitalism*. Cambridge Analytica, Brexit, élection Trump sont les exemples emblématiques.

La posture saine : considérer que tout ce qui sort vers un provider US est potentiellement archivé pour toujours, peu importe ce que dit leur politique de confidentialité officielle.

---

# 4. La confidentialité au quotidien : deux options ZDR

C'est la recommandation par défaut pour 90 % des cas.

**ZDR (Zero Data Retention)** signifie que ta requête arrive au provider, le modèle génère une réponse, et **tout est supprimé immédiatement**. Rien stocké, rien loggué, rien réutilisé pour entraîner un modèle. Le provider s'engage contractuellement et c'est auditable.

Pour ton usage privé, tu as deux options ZDR complémentaires. On **met en avant Synthetic AI** (abonnement privé) pour tout ce qui touche des données clients ou sensibles, et on garde **OpenRouter ZDR** (au token) pour DeepSeek et les tests.

## Option 1 (à privilégier pour le privé) : Synthetic AI

Abonnement privé · ZDR · mis en avant
Synthetic AI (GLM 5.2)

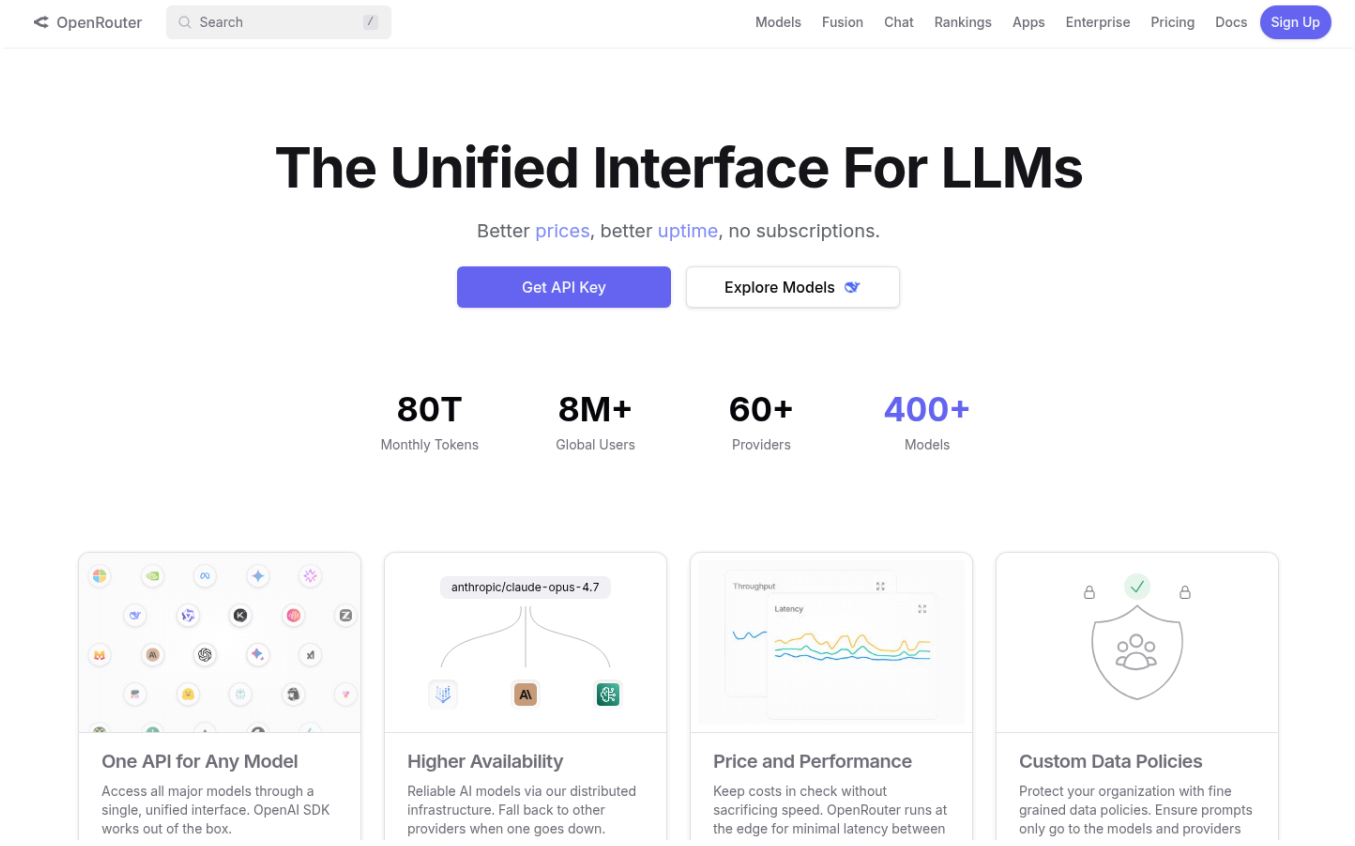
**Synthetic AI** est un **abonnement à 30 $/mois** (pas de paiement au token) qui te donne des tokens **ZDR (zéro rétention)** sur **GLM 5.2**, servis très rapidement, pour à peu près le prix d'un abonnement ChatGPT. C'est notre recommandation pour tout ce qui est privé : données clients, raisonnement sur des sujets sensibles, tâches complexes. Tu prends l'abonnement,

tu connectes GLM 5.2 dans OpenCode, et tout passe en ZDR sans jamais réfléchir au coût par requête.

synthetic.new

## Option 2 : OpenRouter ZDR, au token (DeepSeek et tests)

OpenRouter reste parfait pour **DeepSeek V4 Flash** (ton défaut ultra bon marché) et pour **tester** de nouveaux modèles à la demande, sans abonnement. Tu paies au token, tu actives le ZDR une seule fois, et tu es tranquille.




openrouter.ai : 400+ modèles, 60+ providers, une seule API.

1. Connecte-toi sur openrouter.ai
2. Va dans **Settings** (en haut à droite)
3. Section **Privacy** : active **Zero Data Retention**
4. **Save**

OpenRouter ne route alors que vers des providers qui ont signé l'engagement ZDR (Fireworks, Parasail, certaines instances DeepInfra…). Tes requêtes ne sont plus stockées.

Mis en avant · privé
Synthetic AI (GLM 5.2, abonnement ZDR)

- Entraînement sur tes données : **non**
- Stockage de tes conversations : **non**
- Accès humain à tes échanges : **non**
- Modèle et tarif : **GLM 5.2, 30 $/mois**

Recommandé · au token
OpenRouter ZDR activé

- Entraînement sur tes données : **non**
- Stockage de tes conversations : **non**
- Accès humain à tes échanges : **non**
- Idéal pour : **DeepSeek et les tests**

À éviter pour le sensible
Abonnements ChatGPT Plus / Claude Pro

- Entraînement : oui par défaut, opt-out manuel
- Stockage : tant que le compte existe
- Accès humain : possible (modération abus)
- Rétention minimum : **30 jours après suppression**

### Sur ChatGPT, désactive l'entraînement

Si tu utilises ChatGPT Plus connecté à OpenCode Desktop, fais ce réglage manuellement :

1. Va sur chatgpt.com, clique sur ton avatar en haut à droite
2. **Settings**, puis **Data Controls**
3. Désactive **« Improve the model for everyone »**

OpenAI ne peut plus utiliser tes conversations pour entraîner ses modèles. Note : la rétention reste tant que ton compte existe, ce n'est pas du ZDR. Pour les tâches sensibles, préfère Synthetic AI (GLM 5.2) ou OpenRouter ZDR.

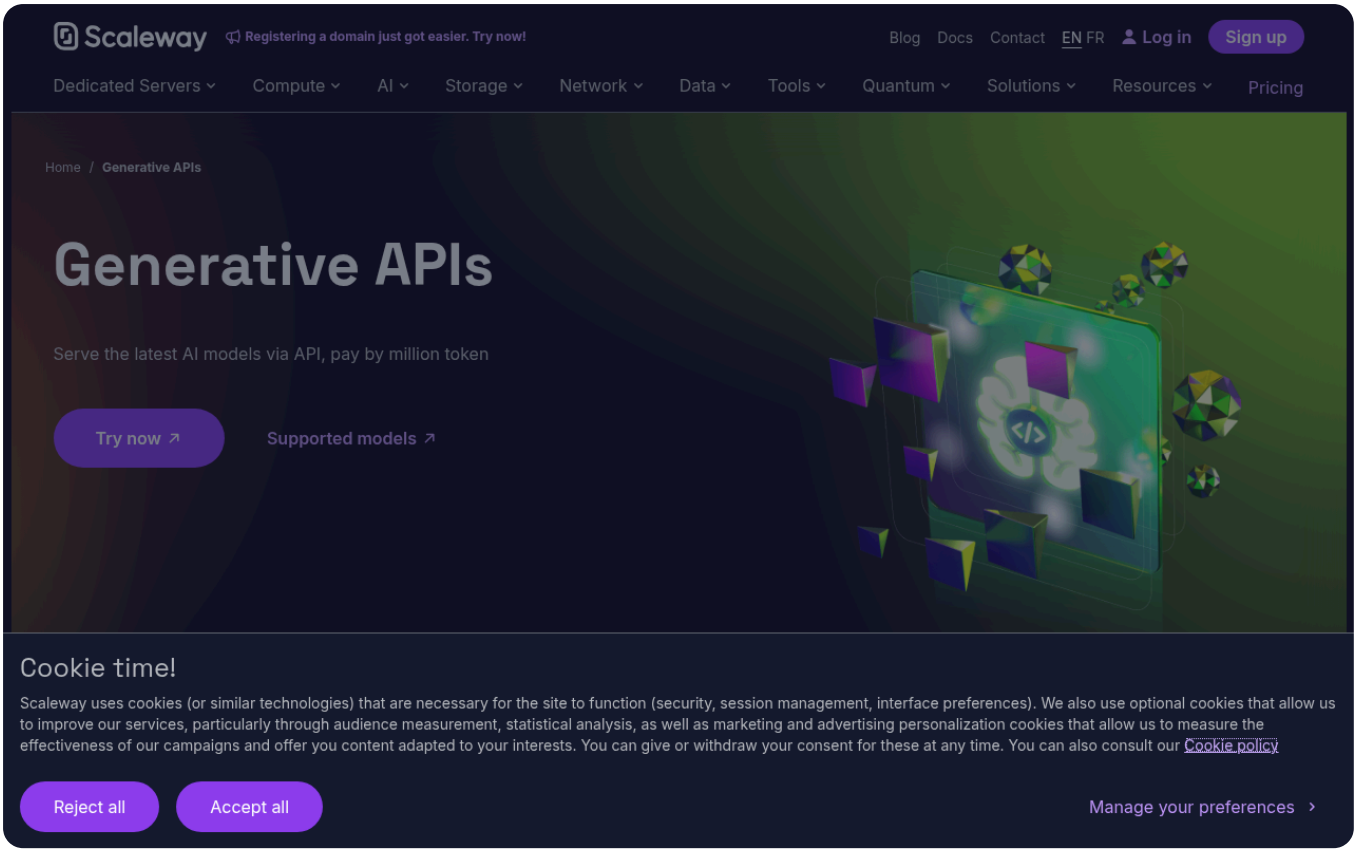
---

## 5. Si tu préfères héberger en Europe

Pour ceux qui veulent une infrastructure non-US par principe (souveraineté, RGPD strict, Cloud Act), voici les options réelles à l'été 2026.

### Recommandation : Scaleway Generative APIs

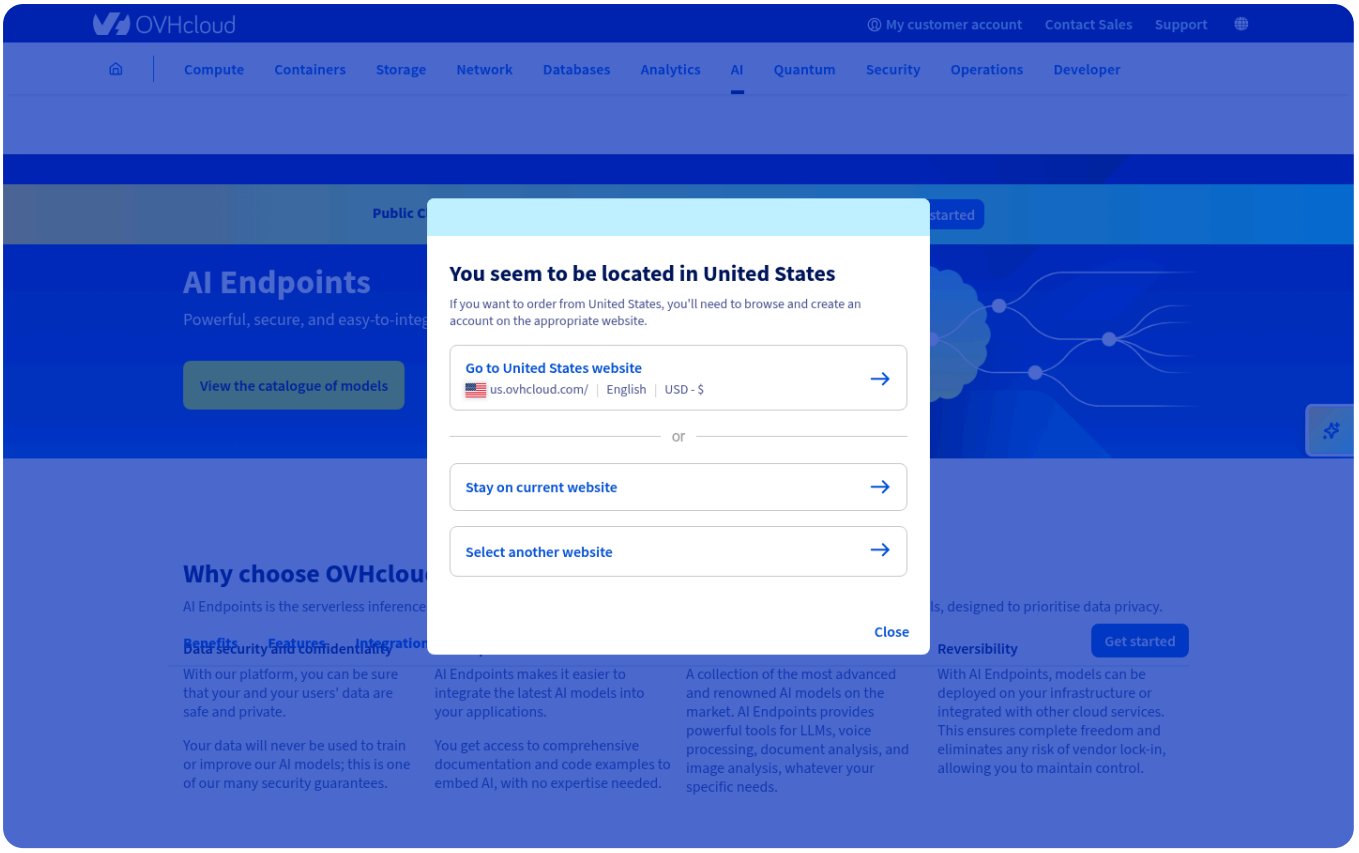
Scaleway est la meilleure option européenne pure-play. Data centers en France, ZDR par défaut (pas besoin d'activer une option), pas d'entraînement sur tes données, RGPD natif. Ils proposent Llama 3, Mistral, Mixtral, DeepSeek R1. Pas dépendant d'AWS / Azure / GCP : infrastructure propre.




scaleway.com/en/generative-apis

## Alternative équivalente : OVHcloud AI Endpoints

OVHcloud AI Endpoints, data centers à Gravelines, plus de 40 modèles open source, certifié ISO 27001/27017/27018/27701. Souveraineté française complète, infrastructure 100 % propre.

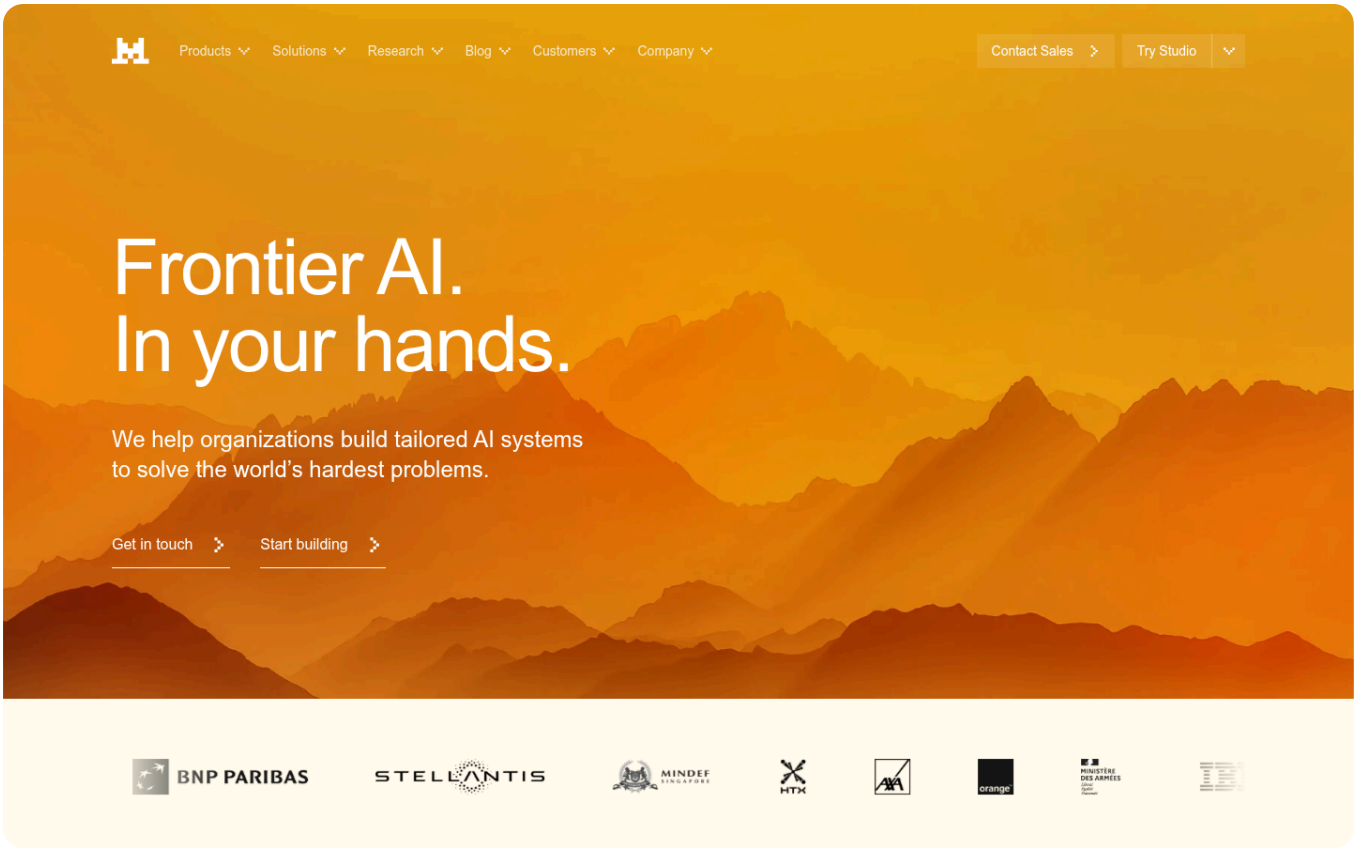



ovhcloud.com/en/public-cloud/ai-endpoints

## Mistral AI : possible mais avec nuances

Mistral AI est français, leurs modèles (Mistral Large 3, Magistral, Mistral Small 4, Codestral) sont solides en 2026. Mais attention :

- **Le Chat (gratuit ou payant)** : entraînement sur tes données activé par défaut, opt-out manuel, **pas de ZDR disponible**. À éviter pour un usage privacy-first.
- **API / La Plateforme** : ZDR disponible **sur demande au support** avec justification. Pas activable d'un clic. Rétention par défaut 30 jours.
- **Cloud Act** : Mistral utilise Microsoft Azure et Google Cloud comme sous-traitants pour l'inférence. Donc le Cloud Act peut techniquement s'appliquer, même si les serveurs sont en UE. Mistral Compute (data center propre en France, 18 000 GPU) est annoncé mais pas pleinement opérationnel.




mistral.ai

Verdict : Mistral via API + DPA signé + ZDR activé est acceptable. Mistral Le Chat, non recommandé pour des données sensibles.

### Autres alternatives à connaître

- **LLMBase** : trop petite et non vérifiable (entité légale opaque, revenus très modestes). **Non recommandé** pour un usage sérieux.
- **Albert AI** (DINUM) : réservé aux agents de l'État, pas accessible aux particuliers.
- **Lucie** (Linagora) : modèle français mais performances actuelles insuffisantes pour un usage knowledge worker.

---

## 6. Pour les données très sensibles : faire tourner le modèle en local

Si tu travailles avec des données de santé, juridiques, hautement confidentielles, ou si tu veux la souveraineté absolue, **fais tourner le modèle sur ton propre matériel**. Rien ne sort de ton réseau.

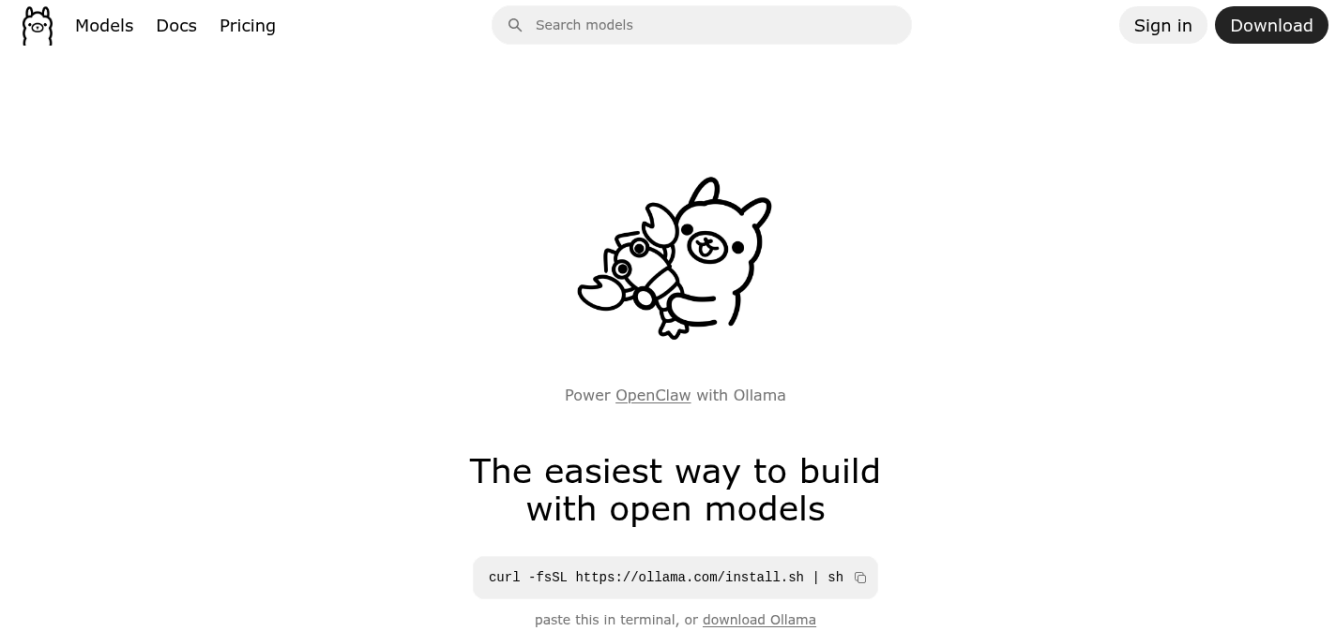
La règle de base : pour que ça soit utilisable, le modèle doit tenir **entièrement dans ta RAM** (sur Mac, c'est de la RAM unifiée partagée avec le GPU). S'il déborde sur le disque, la vitesse

s'effondre. Ton plafond de RAM détermine donc directement les modèles auxquels tu as accès.

Le plus simple (CLI)

Ollama

Une commande pour tout installer (`curl -fsSL ollama.com/install.sh | sh`), puis `ollama run qwen3:8b`. Idéal pour OpenCode Desktop qui s'y connecte en un clic.

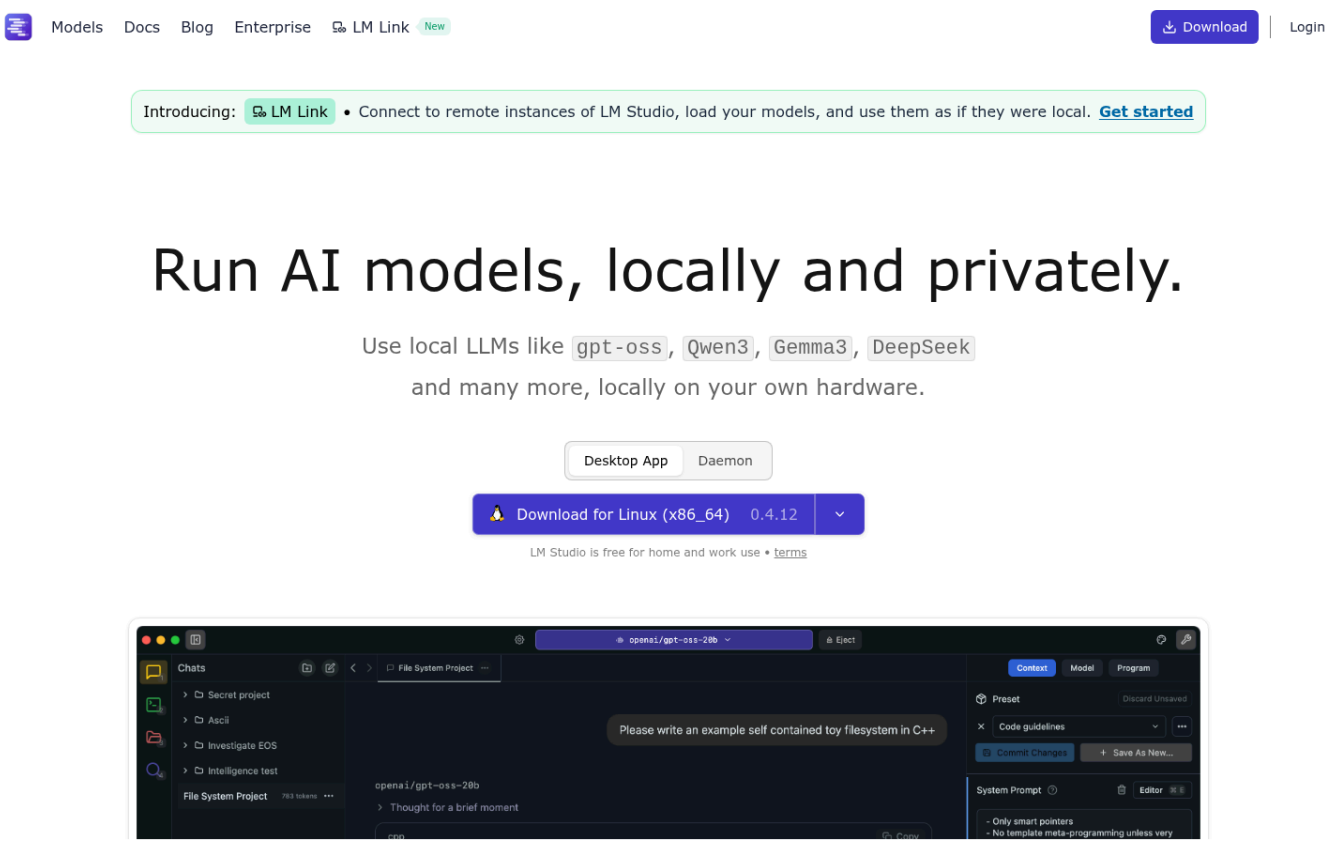



ollama.com

Interface graphique

LM Studio

Une app desktop avec catalogue de modèles, téléchargement en un clic, chat intégré, et un serveur local compatible OpenAI. Plus accessible si tu fuis le terminal.




lmstudio.ai

## Tier 1 : 16 Go de RAM (MacBook Air M2, MacBook Pro M3 base)

Tu peux faire tourner des modèles de 3 à 8 milliards de paramètres en Q4 (quantisation 4 bits). C'est suffisant pour du chat, de l'aide à la rédaction, des recherches simples. **Pas suffisant pour de l'agentique avancé** (multi-step, tool calls complexes).

Modèles à essayer :

- **Gemma 4 E2B** (Google, ~3,5 Go en Q4). C'est le nouveau modèle Google sorti en avril 2026. Multimodal (texte + audio + image). DeepMind a annoncé en mai un projet « Accelerating Gemma 4 » qui ajoute du **multi-token prediction** (le modèle prédit plusieurs tokens à la fois, ce qui accélère l'inférence sans dégrader la qualité). À surveiller.
- **Qwen 3 4B** ou **Qwen 3 8B** : très bon rapport qualité/taille, contexte 256K.
- **Llama 3.2 3B** : rapide, qualité correcte pour du chat.

Vitesse attendue : 30 à 60 tokens/seconde sur Mac M3 16 Go. Confortable pour du chat, juste pour de l'agentique.

L'intérêt à ce tier, c'est surtout de **comprendre comment le local fonctionne** et de tester. Pour un usage knowledge worker complet, tu seras vite limité.

## Tier 2 : 32 à 64 Go de RAM (MacBook Pro M3 Max, Mac mini M4 Pro)

C'est le tier où le local devient vraiment utilisable au quotidien. Les modèles 12 à 32 milliards de paramètres deviennent confortables.

- **Gemma 4 26B-A4B** (Google, MoE, ~18 Go en Q4) : 3,8B de paramètres actifs seulement à chaque token, donc rapide.
- **Qwen 3 32B** (Q4) : meilleur open weight 30B à l'été 2026.
- **DeepSeek R1 32B** (Q4) : fort en raisonnement.
- **Mistral Small 24B** (Q4) : inférence très rapide.

Sur 64 Go, les modèles 70B (~43 Go en Q4) deviennent accessibles : Llama 3.3 70B, DeepSeek R1 70B.

## Tier 3 : 128 Go et plus (Mac Studio M3/M4 Ultra)

C'est ici que tu peux faire tourner les **gros modèles open source frontières** (GLM 5.2, DeepSeek V4 Flash, Qwen 3 235B). Ordre de prix :

- **Mac Studio M3 Ultra 192 Go** : 10 000 à 13 000 €
- **Mac Studio M4 Ultra 512 Go** : 25 000 à 30 000 € (haut de gamme grand public absolu)

À 192 Go, tu peux faire tourner :

- **DeepSeek R1 70B en Q8** (~80 Go) : qualité quasi équivalente à la version full precision
- **Qwen 3 235B MoE en Q2** (~90-100 Go) : 22B actifs à chaque token
- **GLM 5.2 744B MoE en Q2** (~150 Go estimé) : 40B actifs
- **Mistral Large 123B en Q4** (~75 Go)

À 512 Go, **DeepSeek V4 Flash en Q4** ou même certains modèles 400B+ deviennent faisables.

## Comprendre la quantisation (Q2, Q4, Q5...)

Un modèle « brut » stocke ses paramètres en 16 bits (FP16). Le quantiser, c'est compresser ces paramètres sur moins de bits, pour qu'il prenne moins de RAM.

| Niveau | Bits/poids | Effet sur la qualité | Quand l'utiliser |
|---|---|---|---|
| **Q5** | ~5 bits | Quasi identique au full precision | Si tu as la RAM |

| Niveau | Bits/poids | Effet sur la qualité | Quand l'utiliser |
|---|---|---|---|
| **Q4** | ~4 bits | Légère perte, peu perceptible | **Le défaut recommandé** |
| **Q2** | ~2 bits | Dégradation visible : hallucinations, raisonnement moins fiable | Uniquement pour les très gros modèles (100B+) quand tu n'as pas le choix pour tenir en RAM |

Le suffixe `K_M` ou `K_S` (K-quant Medium / Small) indique une variante optimisée qui préserve mieux les couches importantes du modèle. Pour Ollama, le défaut est généralement `Q4_K_M`, c'est très bien.

### Pour aller plus loin

Sur le hardware et l'optimisation du local, je te recommande la chaîne YouTube **@xcreate** qui couvre le sujet en profondeur : choix de hardware, quantisation, comparaisons de modèles, retours d'expérience sur les Mac Studio.

### Cas pratique : une équipe qui mutualise

Pour une boîte de 5 à 50 personnes, **un Mac Studio Full Spec partagé** devient rapidement rentable comparé à 10 ou 50 abonnements API. Tu installes Ollama dessus, tu exposes l'API en réseau local, chacun connecte son OpenCode Desktop dessus. Rien ne sort de l'infra. ROI typique : moins d'un an de mutualisation par rapport aux abonnements individuels.

---

## 7. Gérer ses secrets : API keys et mots de passe

Commençons par dédramatiser. La dernière promo a passé un temps fou à installer et brancher Bitwarden. Dans la grande majorité des cas, tu n'en as pas besoin.

### Si ton coffre est en Obsidian Sync, tes clés sont déjà chiffrées

Quand ton coffre est synchronisé avec **Obsidian Sync**, tout est **chiffré de bout en bout** (E2E) avec ton mot de passe d'encryption. Même Obsidian ne peut pas lire tes fichiers.

Conséquence directe : avoir tes clés API en clair dans tes fichiers de config (`opencode.json`, `.env`) ou dans une note, **ce n'est pas un problème grave**. C'est chiffré E2E, le risque réel est très faible.

Tu n'es pas le Pentagone. Tu es un knowledge worker qui veut avancer vite avec son IA. Privilégie la **simplicité d'usage** plutôt que la complexité d'un password manager que tu vas galérer à connecter. Des clés en clair dans un coffre chiffré E2E, c'est très bien pour démarrer, et très bien pour durer.

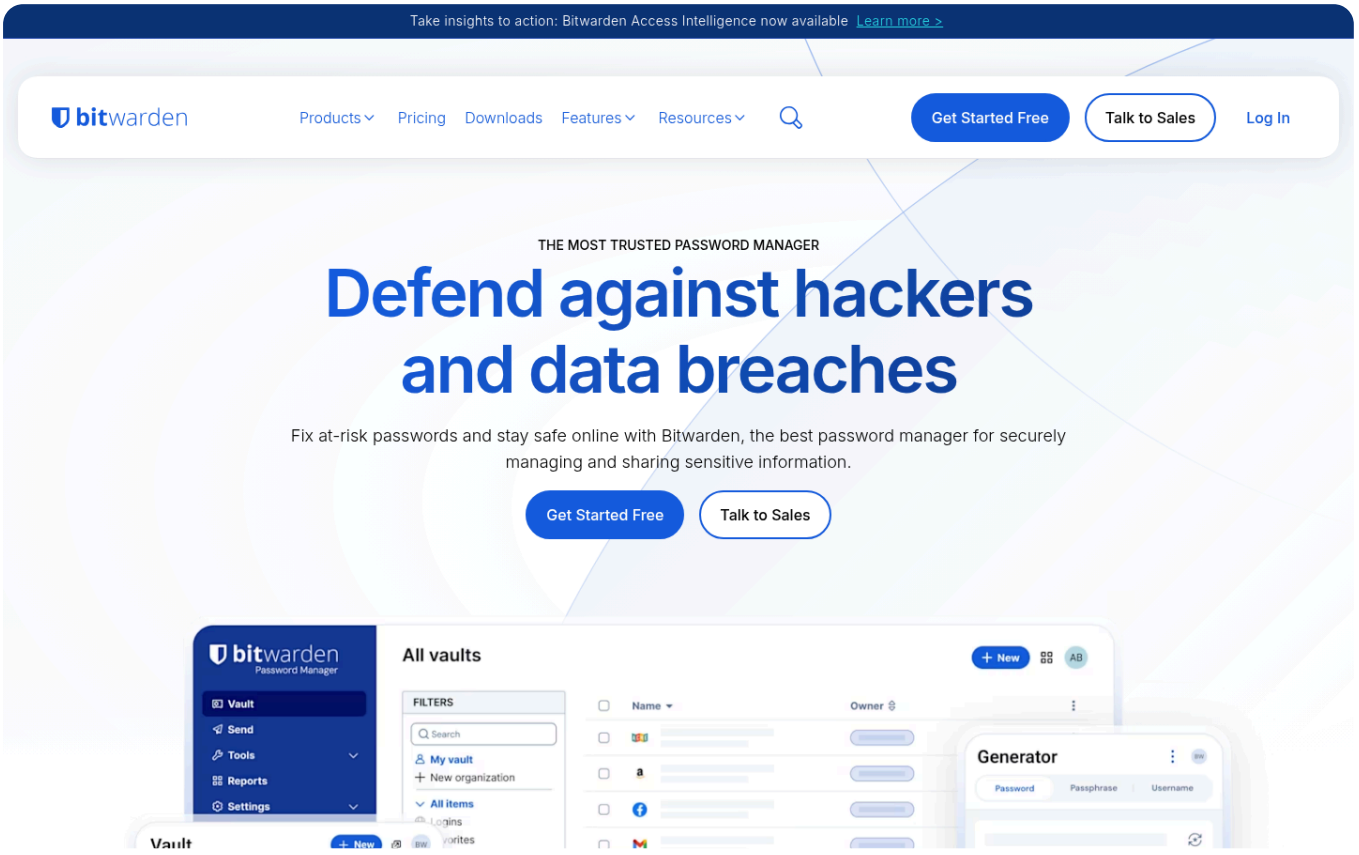
## Le bon sens minimal : 2 règles suffisent

1. **Ne colle jamais une clé dans un espace public** : session ChatGPT partagée, Discord, forum, ticket support. Là, elle sort vraiment en clair chez un tiers.
2. **Ne commite jamais une clé dans un repo Git public non chiffré.** Un `.env` poussé sur un GitHub public, c'est la fuite classique. Ajoute `.env` à ton `.gitignore` (détaillé en semaine 3) et le sujet est réglé.

Le reste, c'est du confort.

## Optionnel (avancé) : Bitwarden comme coffre central

Tu n'en as pas besoin pour le bootcamp. Mais si tu veux aller plus loin, ou si tu gères des **données client très sensibles** (santé, juridique, secrets d'une boîte tierce), un password manager centralise proprement tous tes secrets (API keys, mots de passe, tokens, clés SSH). Bitwarden est gratuit, open source, chiffré de bout en bout, avec app mobile et extension navigateur.



bitwarden.com

## Connecter Bitwarden à ton IA via Bitwarden CLI

Bitwarden propose un CLI (documentation officielle) qui permet à ton IA de **récupérer un secret à la volée** dans tes scripts ou tes sessions, sans jamais que tu n'aies à le coller en clair quelque part.

**Installer le CLI** :

```
# Mac
brew install bitwarden-cli

# Windows (via Chocolatey ou exécutable natif)
choco install bitwarden-cli

# Cross-platform via npm
npm install -g @bitwarden/cli
```

**Workflow concret** :

1. Une fois par session, tu déverrouilles ton coffre manuellement dans ton terminal :

```
bw unlock
```

Bitwarden te demande ton mot de passe maître. Une fois validé, il te retourne une commande à copier-coller :

```
export BW_SESSION="eyJhbGciOi..."
```

1. Tu colles cette commande dans ton shell. **Tant que cette variable d'env existe**, toutes les commandes Bitwarden fonctionnent sans demander à nouveau ton mot de passe.
2. Maintenant, ton IA dans OpenCode peut récupérer n'importe quel secret à la volée :

```
# Lister tes secrets
bw list items --search "OpenAI"

# Récupérer juste le mot de passe d'un item
```

```
bw get password "OpenAI API"

# Récupérer l'item complet en JSON
bw get item "OpenAI API"
```

1. Quand tu as fini ta session, verrouille à nouveau :

```
bw lock
```

Le token expire automatiquement après inactivité, donc même si tu oublies, tu n'es pas exposé indéfiniment.

**Avantage** : ton IA peut accéder à tes secrets quand elle en a besoin (pour faire un appel API, configurer un MCP), mais **elle ne voit jamais ton mot de passe maître**. Seul le token de session temporaire transite. Et tu peux révoquer l'accès à tout moment avec `bw lock`.

## Prompt à donner à ton IA

Quand tu veux qu'elle utilise un secret dans une tâche, dis-lui simplement :

```
J'ai déverrouillé mon coffre Bitwarden. Mon BW_SESSION est
exporté.
Récupère ma clé API OpenAI avec `bw get password "OpenAI API"`
et utilise-la pour [tâche].
```

Elle exécute, fait son boulot, et ton secret n'est jamais écrit en clair quelque part de persistant.

## Et les secrets dans Obsidian ?

C'est parfaitement OK, **à condition que ton coffre soit synchronisé via Obsidian Sync** (chiffrement de bout en bout avec ton mot de passe d'encryption). Dans ce cas, une clé API dans une note ou dans `opencode.json` est chiffrée dès qu'elle quitte ta machine. C'est le setup par défaut recommandé.

Deux réflexes de bon sens, quand même :

- Ton coffre est en clair **sur ton disque local** (seul le transfert vers le cloud est chiffré). Sur ta machine perso, ce n'est pas un souci. Sur une machine partagée, active le chiffrement disque (FileVault sur Mac, activé par défaut).

- **Ne pousse pas ton vault sur un GitHub public.** Si tu veux le versionner sur Git, garde le repo privé, ou mieux, auto-hébergé (Gitea, voir semaine 3).

Autrement dit : coffre en Obsidian Sync, tes clés y sont bien. Un password manager n'ajoute qu'un cran de sécurité optionnel, utile surtout pour des données client très sensibles.

### Dans tes projets de développement

Pour les projets de code (sites web, apps), utilise un fichier `.env` à la racine et ajoute-le dans `.gitignore` :

```
# .gitignore
.env
.env.local
.env.*.local
```

Les valeurs peuvent rester en clair dans le `.env` (il est ignoré par git). Si tu utilises un password manager, tu peux aussi les injecter via son CLI au moment du déploiement (CI/CD ou script local), mais ce n'est pas obligatoire.

---

## 8. Synchronisation du coffre

Quand tu synchronises ton coffre avec Obsidian Sync, GitHub ou autre :

- **Obsidian Sync** (recommandé) : tes données passent par les serveurs d'Obsidian, mais sont **chiffrées de bout en bout** avec ton mot de passe d'encryption. Même Obsidian ne peut pas les lire.
- **GitHub** (repo privé) : stockage sur serveurs Microsoft (US, soumis au Cloud Act). GitHub ne lit pas le contenu pour entraîner Copilot tant que tu ne l'actives pas explicitement.
- **GitLab self-hosted** ou **Gitea** sur ton propre serveur : alternative souveraine, plus technique. On installe Gitea en semaine 3 si tu prends ce chemin.

---

## 9. Récapitulatif

| Question | Réponse |
| --- | --- |
| L'IA peut-elle tout faire par défaut sur mon ordi ? | Non. Elle demande validation au cas par cas. Tu peux activer l'auto-accept dans les réglages. |
| Mes fichiers quittent-ils mon ordi en bloc ? | Non. L'IA ne lit que les fichiers pertinents pour ta requête, puis envoie ces extraits seulement. |
| Faut-il deux coffres pour séparer perso et sensible ? | Non. Tu peux changer de modèle selon la tâche dans le même coffre. |
| Comment avoir 0 conservation chez le provider ? | Synthetic AI (abonnement 30 $/mois, GLM 5.2, ZDR) pour le privé, ou OpenRouter ZDR (au token) pour DeepSeek et les tests. |
| Et chez ChatGPT ? | Désactive l'entraînement dans Data Controls. Reste rétention tant que le compte existe. |
| Existe-t-il des alternatives européennes ? | Oui : Scaleway et OVHcloud sont les meilleures pures options souveraines. Mistral via API + ZDR est acceptable. |
| Pour des données ultra sensibles ? | Modèle local (Mac Studio Full Spec, Ollama / LM Studio). |
| Où stocker mes API keys ? | En clair dans tes fichiers suffit, tant que ton coffre est en Obsidian Sync (chiffré E2E). Bitwarden est optionnel. Jamais dans un repo public ni un chat public. |

---

- [ ] Je comprends que l'IA demande validation par défaut, et que je peux activer l'auto-accept
- [ ] J'ai utilisé le mini-prompt pour configurer mes permissions opencode.json
- [ ] J'ai compris que l'IA ne lit que les fichiers pertinents (pas tout le coffre)
- [ ] Je connais Synthetic AI comme option ZDR privée mise en avant (GLM 5.2, abonnement 30 $/mois)
- [ ] J'ai activé ZDR sur OpenRouter (Settings, Privacy) pour DeepSeek et les tests
- [ ] J'ai désactivé l'entraînement sur ChatGPT (Data Controls)

- [ ] Je sais que des alternatives européennes existent (Scaleway, OVHcloud) pour les cas où c'est nécessaire
- [ ] Mes clés API sont dans un coffre chiffré E2E (Obsidian Sync), jamais dans un repo public ni un chat public

## Vidéos

YOUTUBE

**Où vont vraiment vos données quand vous parlez à l'IA**

https://www.youtube.com/watch?v=uIjp0Bx9mWg

---

## Assets

- OpenRouter homepage
- Scaleway Generative APIs
- OVHcloud AI Endpoints
- Mistral AI homepage
- Ollama homepage
- LM Studio homepage
- Bitwarden homepage

---

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