LLM Local
sur Laptop

Installer un modèle IA en 2026

sans GPU dédié · 8 Go RAM minimum · 100% souverain

CONTENU DU GUIDE

8 Go RAM	CPU seul	0 €	100%
minimum requis	pas de GPU	coût de licence	données locales

1 Pourquoi faire tourner un LLM en local ?

En 2026, la question n'est plus **« peut-on ?»** mais **« comment choisir le bon modèle ?»**. Les SLM (Small Language Models) modernes atteignent sur CPU seul des performances inimaginables il y a 18 mois.

■ Souveraineté des données

Aucune donnée ne quitte votre machine. Conformité RGPD garantie. Crucial pour PME européennes, cabinets médicaux, juridiques, RH.

■ Zéro coût de requête

Contrairement aux API cloud (ChatGPT, Claude : 0.002–0.015 $/1K tokens), un LLM local ne coûte rien à l'usage. ROI immédiat sur gros volumes.

■ Zéro latence réseau

Réponse en local < 2 s sur CPU moderne. Pas de dépendance à la connexion. Idéal pour ateliers offline ou zones sans fibre.

■ Personnalisation totale

Fine-tuning sur vos données métier. System prompt libre. Intégration directe dans vos workflows N8N ou pipelines RAG.

2 La quantification — clé qui rend tout possible

La **quantification** réduit la précision des poids du modèle (de 32 bits flottants → 4 bits entiers). Résultat : un modèle 7B qui demandait 14 Go passe à **4–5 Go** avec une perte de qualité inférieure à 3 %.

Format	Précision	RAM (modèle 7B)	Qualité	Recommandé ?
FP32	32 bits	~28 Go	100 %	Non — trop lourd
FP16	16 bits	~14 Go	99,9 %	GPU 16 Go+
Q8_0	8 bits	~7,5 Go	99,5 %	16 Go RAM
Q4_K_M	4 bits	~4,5 Go	97–98 %	■ Idéal 8 Go RAM
Q3_K_M	3 bits	~3,3 Go	94–95 %	4 Go RAM
Q2_K	2 bits	~2,5 Go	88 %	Ultra-contraint

■ **Règle de base :** RAM nécessaire ≈ Nombre de paramètres (B) × 0,6 Go en Q4. Exemples : 4B → ~2,5 Go | 7B → ~4,5 Go | 14B → ~8,4 Go

3 Panorama des SLM — État de l'art Mars 2026

Les **SLM (Small Language Models)** ont fait un bond qualitatif majeur en 2025–2026. Règle empirique : un modèle 4B bien entraîné de 2026 rivalise avec un 13B de 2023.

Modèle	Éditeur	Params	RAM Q4	CPU tok/s	Usage idéal	Licence
Qwen3 4B	Alibaba ■■	4B	~2,5 Go	15–25	Chat, raisonnement, tool use	Apache 2.0
Gemma 3 4B	Google	4B	~2,7 Go	12–20	Chat, multimodal (vision+texte)	Gemma ToS
Phi-4 Mini	Microsoft	3.8B	~2,3 Go	18–28	Code, raisonnement, STEM	MIT
Llama 3.2 3B	Meta	3B	~2,0 Go	20–35	Chat rapide, mobile, RAG	Llama 3.2
SmolLM2 1.7B	HuggingFace	1.7B	~1,1 Go	40–60	Ultra-léger, embarqué	Apache 2.0

DeepSeek-R1 1.5B	DeepSeek	1.5B	~1,0 Go	35–50	Raisonnement logique / maths	MIT
Mistral 7B v0.3	Mistral AI ■■	7B	~4,5 Go	8–15	Général, français, RAG	Apache 2.0
Qwen3 8B	Alibaba ■■	8B	~5,2 Go	6–12	Meilleur 8B général, tool use	Apache 2.0

■■ **Recommandation souveraineté européenne :** Mistral 7B (Mistral AI, Paris) offre d'excellentes performances en français. Seul modèle du top avec une entreprise européenne derrière.

Guide de choix rapide

Si vous voulez…	Modèle recommandé	RAM min.
Démarrer vite, usage général	Llama 3.2 3B	4 Go libres
Meilleure qualité sur 8 Go	Qwen3 4B ou Mistral 7B Q4_K_M	6–7 Go libres
Code Python / JS	Phi-4 Mini ou Qwen3 4B	4–5 Go libres
RAG + documents en français	Mistral 7B Q4_K_M	6 Go libres
Ultra-rapide (proto / atelier)	SmolLM2 1.7B	2 Go libres
Raisonnement logique / maths	DeepSeek-R1 1.5B ou Qwen3 4B	2–5 Go libres
Multimodal (image + texte)	Gemma 3 4B	4 Go libres

4 Ollama — 3 commandes pour démarrer

Ollama est l'outil de référence en 2026 pour exécuter des LLM localement. Il gère le téléchargement, la quantification et expose une API REST compatible OpenAI.

■ Linux / macOS

```
# Installation en une ligne
curl -fsSL https://ollama.com/install.sh | sh

# macOS via Homebrew
brew install ollama

# Démarrer le serveur
ollama serve
```

■ Windows

1. Télécharger l'installateur **.exe** sur **ollama.com/download**
2. Double-cliquer → Next → Finish
3. Ollama se lance automatiquement en arrière-plan
4. Ouvrir un terminal PowerShell et taper :

```
ollama run qwen3:4b
```

Commandes essentielles

Commande	Description
ollama run qwen3:4b	Télécharger + lancer le modèle (interactif)
ollama run mistral	Lancer Mistral 7B
ollama list	Lister les modèles installés
ollama pull phi4-mini	Télécharger sans lancer
ollama rm mistral	Supprimer un modèle
ollama ps	Modèles actifs en mémoire
ollama show qwen3:4b	Détails : params, quantization, taille
OLLAMA_NUM_CTX=2048 ollama serve	Réduire le contexte → gain de vitesse

API locale compatible OpenAI

Une fois Ollama actif, tout outil compatible OpenAI peut pointer vers **http://localhost:11434/v1** — sans changer une ligne de code.

```
# Test API depuis un terminal
curl http://localhost:11434/v1/chat/completions \
-H 'Content-Type: application/json' \
-d '{"model":"qwen3:4b","messages":[{"role":"user","content":"Bonjour !"}]}'

# Python / LangChain
from langchain_ollama import OllamaLLM
llm = OllamaLLM(model='qwen3:4b')
print(llm.invoke('Résume ce contrat...'))

# LightRAG / CaféIA — fichier .env
LLM_BINDING=openai
LLM_BINDING_HOST=http://localhost:11434/v1
LLM_MODEL=qwen3:4b
LLM_BINDING_API_KEY=ollama
```

5 LM Studio — Interface graphique sans ligne de commande

Pour ceux qui préfèrent éviter le terminal, **LM Studio** (lmstudio.ai) est l'alternative graphique de référence. Même API locale, même résultat.

Installation

→ Télécharger sur **lmstudio.ai** (Windows / Mac / Linux)
→ Installation standard double-clic
→ Parcourir et télécharger les modèles depuis l'interface
→ Filtrage automatique selon la RAM disponible

Fonctionnalités clés

→ Chat intégré avec historique
→ Serveur API local port 1234 (compatible OpenAI)
→ Réglage température, context, system prompt
→ Détection automatique GPU / CPU disponible
→ GPU offloading partiel (layers sur VRAM)

Ollama vs LM Studio

Critère	Ollama
Interface	CLI + API
Public cible	Développeurs
API locale	Port 11434
Open source	■ Oui
Déploiement serveur	■ Facile
Gestion GPU	Automatique

6 Optimiser les performances sur CPU

Sur un laptop **sans GPU dédié**, quelques réglages permettent de gagner 30–50 % de vitesse sans changer de matériel.

Conseil	Impact	Mise en œuvre
Choisir Q4_K_M plutôt que Q8	2× plus rapide, qualité quasi identique	Par défaut dans Ollama
Réduire le context window	+20–40 % vitesse	OLLAMA_NUM_CTX=2048
Fermer les applications inutiles	+15–25 % vitesse	Libérer RAM pour le modèle
Activer tous les threads CPU	Linéaire avec le nb de cœurs	llama.cpp : --threads $(nproc)
RAM swap si 8 Go insuffisants	Permet modèles > RAM physique	Windows : fichier de page ≥ 16 Go
Préchauffer le modèle	1re réponse 3× plus rapide	ollama run model 'warmup'

Vitesses de référence — CPU uniquement (Intel i7 12e Gen / AMD Ryzen 7 5800H, 8 Go RAM)

Modèle	Quant.	Tok/sec	Réponse courte	500 tokens
SmolLM2 1.7B	Q4_K_M	40–60	< 0,5 s	~10 s
Llama 3.2 3B	Q4_K_M	20–35	~1 s	~15–25 s
Qwen3 4B	Q4_K_M	15–25	~1,5 s	~20–35 s
Phi-4 Mini 3.8B	Q4_K_M	18–28	~1 s	~18–28 s
Mistral 7B	Q4_K_M	8–15	~2–3 s	~35–65 s
Qwen3 8B	Q4_K_M	6–12	~3–4 s	~45–85 s

Mesures indicatives — varient selon OS, température CPU, longueur du contexte.

7 Intégration dans vos workflows — RAG & N8N

Une fois Ollama actif, il expose une **API REST locale compatible OpenAI**. Tout outil supportant OpenAI peut pointer vers votre LLM local sans modifier le code.

Connexion N8N → Ollama

Dans N8N, choisir le nœud **Ollama Chat Model** ou configurer OpenAI avec base URL = `http://localhost:11434/v1`

Connexion LightRAG / CaféIA

```
# fichier .env LightRAG
LLM_BINDING=openai
LLM_BINDING_HOST=http://localhost:11434/v1
LLM_MODEL=qwen3:4b
LLM_BINDING_API_KEY=ollama
```

Continue.dev (IDE)

Extension VS Code / JetBrains. Pointer vers Ollama pour de l'autocomplétion de code 100 % locale, sans envoyer votre code vers un serveur tiers.

Cas d'usage concrets

→ **Résumé automatique** de documents entrants (factures, contrats)
→ **Réponse aux emails** avec LLM local via N8N
→ **RAG sur documents internes** — base de connaissances privée
→ **Extraction d'entités** (NER) : clients, produits, dates
→ **Chatbot support** — données jamais dans le cloud
→ **Génération de code** dans votre IDE via Continue.dev

■ **Note souveraineté :** Mistral AI (France) propose aussi **mistral-small** via API cl
localité, tout en restant européen.

8 Bonus — LLM sur smartphone (Android & iPhone) 2026

En 2026, les smartphones haut de gamme (Snapdragon 8 Gen 3, Apple A18) disposent de NPU capables de faire tourner des modèles 1B–4B à **30–50 tokens/seconde**. C'est utilisable en production.

Application	OS	Modèles supportés	Particularité
MLC Chat	Android + iOS	Llama 3.2, Gemma 2/3, Phi-3.5, Qwen 2.5	Optimisé NPU Snapdragon
SmolChat	Android	Tous formats GGUF (Llama, Gemma 3n…)	Open source, Google Play
Google AI Edge Gallery	Android	Gemma 3 1B–4B, Qwen 2.5 1.5B	App officielle Google
PocketPal AI	Android + iOS	Phi, Llama, Qwen (format GGUF)	HuggingFace intégré
LM Studio Mobile	iOS	Llama, Mistral, Phi, Qwen	Sync modèles avec desktop

Réalités terrain (tests 2025–2026)

→ Snapdragon 8 Gen 2+ : Llama 3.2 3B à 8–12 t/s — utilisable

→ Galaxy A54 (6 Go RAM) : Gemma 2B ou Phi-3-mini seulement

→ iPhone 15 Pro (A17) : Gemma 2B à 15–20 t/s via Core ML

→ Pixel 8 Pro : Gemma optimisé NPU, 20–30 t/s

→ Batterie : 50 % en 90 min sur modèle 3B actif

Modèles recommandés mobile

→ **Gemma 3n 1B** (Google) : multimodal, ~0,8 Go

→ **Qwen3 0.6B** : 40 t/s sur Pixel 8, ~350 Mo

→ **SmolLM2 135M** : ultra-léger, IoT

→ **Phi-3 Mini 3.8B** : meilleure qualité si ≥ 6 Go RAM

Google Gemma 3n — innovation 2026

Gemma 3n est la première architecture **nativement multimodale** pour mobile : texte, image, vidéo et audio sur un même modèle. Disponible via Google AI Edge (LiteRT) sur Android et iOS. RAG et Function Calling directement on-device.

ExecuTorch (Meta) — Qwen3 sur Android

ExecuTorch propulse les features IA d'Instagram, WhatsApp et Messenger. Avec Unsloth + ExecuTorch, déploiement de Qwen3-0.6B sur Pixel 8 et iPhone 15 Pro à **~40 tokens/seconde**.

Limites actuelles

→ Modèles > 4B : risque d'OOM sur smartphones < 8 Go

→ Pas de fine-tuning local en temps réel

→ Consommation batterie élevée (~1–3W en inférence)

→ Pas de remplacement des LLM cloud pour tâches complexes

✓ Checklist — Déploiement LLM local en 20 minutes

	Étape	Détail
■	Vérifier la RAM disponible	Minimum 8 Go total. Laisser 2–3 Go pour l'OS. Fermer le navigateur.
■	Installer Ollama	ollama.com → installateur 1 clic (Windows / Mac / Linux)
■	Choisir le bon modèle	8 Go RAM → Qwen3 4B ou Mistral 7B Q4_K_M
■	Télécharger le modèle	ollama run qwen3:4b (télécharge + lance automatiquement)
■	Tester l'API locale	curl http://localhost:11434/api/generate -d '{"model":"qwen3:4b","prompt":"Bonjour"}'
■	Connecter à votre outil	N8N, LangChain, LightRAG → base_url = http://localhost:11434/v1
■	Vérifier la vitesse	Objectif : ≥ 10 tokens/sec pour usage interactif
■	Préparer offline	ollama pull model → stocké dans ~/.ollama/models/ (copier sur USB)

■ Prochaine étape

Intégrer votre LLM local dans un pipeline **RAG** avec **LightRAG + Qdrant** pour interroger vos documents internes. C'est l'objet du workshop **CaféIA — GraphRAG** (Mensaflow × UPVD).