

# Créez votre IA Locale 🚀

Un guide pratique pour maîtriser l'intelligence artificielle sur votre machine.

## Qu'est-ce qu'une IA locale ?

- **Confidentialité :** Vos données restent sur votre machine, jamais envoyées à des serveurs externes.
- **Autonomie :** Fonctionne sans connexion internet, idéal pour les environnements isolés.
- **Maîtrise :** Contrôle total sur l'IA et son fonctionnement.

## De quoi avez-vous besoin ?

### Matériel

- **Processeur (CPU) :** Intel i5/Ryzen 5 (minimum), i7/Ryzen 7 (recommandé).
- **Mémoire vive (RAM) :** 16 Go (minimum), 32 Go ou plus (recommandé).
- **Carte graphique (GPU) :** NVIDIA RTX 3060 (minimum), RTX 4070+ (recommandé) pour de meilleures performances.
- **Stockage :** SSD 500 Go (minimum), 1 To+ (recommandé).

### Logiciels

- **Système d'exploitation :** Windows 10/11, macOS, Linux.
- **Python :** Version 3.9 ou supérieure.
- **Ollama :** Pour exécuter les modèles de langage localement.
- **Bibliothèques Python :** LangChain, Pydantic, FastAPI, etc.

## Les 5 grandes étapes

1. **1**  
Définir votre besoin : Comprendre le problème à résoudre.
2. **2**  
Préparer vos données : Collecte, nettoyage et formatage.
3. **3**  
RAG et Fine-tuning : Choisir et adapter l'approche IA.
4. **4**  
Installation complète : Mettre en place l'environnement.
5. **5**  
Créer votre système RAG ! : Développer et tester votre solution.

## Étape 1 : Définir votre besoin

Avant de commencer, posez-vous les bonnes questions :

- Quel est le problème spécifique que l'IA doit résoudre ?
- Quel type de tâche l'IA effectuera-t-elle (réponse à des questions, résumé, génération de texte, classification) ?
- Qui sont les utilisateurs finaux et quelles sont leurs attentes ?

Exemples : assistant pour la recherche documentaire, chatbot interne pour le support client, outil de résumé de rapports.

## Étape 2 : Préparer vos données

La qualité des données est cruciale pour la performance de votre IA.

### Processus typique :

- **\*\*Collecte :\*\*** Rassembler les documents pertinents (PDF, DOCX, TXT, HTML).
- **\*\*Extraction :\*\*** Convertir les documents en texte brut.
- **\*\*Nettoyage :\*\*** Supprimer les éléments indésirables (balises HTML, URLs, caractères spéciaux, erreurs OCR).
- **\*\*Déduplication :\*\*** Éliminer les doublons pour éviter les biais.
- **\*\*Formatage :\*\*** Structurer les données pour l'entraînement ou l'indexation (ex: JSONL).

Un script Python peut automatiser ces tâches pour un traitement par lots.

## Étape 3 : RAG et Fine-tuning

### RAG (Retrieval Augmented Generation)

L'IA "cherche" des informations pertinentes dans une base de connaissances avant de générer une réponse. Idéal pour des réponses factuelles et à jour.

```
# Pseudo-code RAG
query = "Quelle est la capitale de la France ?"
documents = vector_store.retrieve(query) # Recherche
context = combine(documents)
answer = llm.generate(query, context) # Génération
```

### Fine-tuning (Ajustement fin)

Adapter un modèle de langage pré-entraîné à un domaine ou un style spécifique avec vos propres données. Utile pour des tâches très spécifiques ou un ton particulier.

Le choix dépend de votre cas d'usage : RAG pour la précision factuelle, Fine-tuning pour la spécialisation comportementale.

## Étape 4 : Installation complète

### Installer Ollama

Téléchargez et installez Ollama depuis [ollama.com](https://ollama.com).

```
# Télécharger un modèle (ex: Llama 3)
ollama pull llama3

# Tester le modèle
ollama run llama3 "Bonjour, comment allez-vous ?"
```

### Installer Python et dépendances

Assurez-vous d'avoir Python 3.9+ et installez les bibliothèques :

```
# Vérifier Python
python3 --version

# Installer les dépendances
pip install langchain ollama pydantic fastapi uvicorn
```



## Vérification et choix du modèle

### Vérification de l'installation

Un script simple pour s'assurer qu'Ollama et les embeddings fonctionnent :

```
# verif_ollama.py
from langchain_community.llms import Ollama
from langchain_community.embeddings import OllamaEmbeddings

llm = Ollama(model="llama3")
embeddings = OllamaEmbeddings(model="llama3")

print(llm.invoke("Salut !"))
print(embeddings.embed_query("Ceci est un test."))
```

### Choisir le bon modèle

Considérez la VRAM de votre GPU, la vitesse et la qualité souhaitée. Ex: Llama 3 8B pour 8Go VRAM.

## Étape 5 : Créer votre système RAG !

Voici un aperçu simplifié d'un pipeline RAG en Python :

```
# rag_simple.py
from langchain_community.document_loaders import TextLoader
from langchain_community.embeddings import OllamaEmbeddings
from langchain_community.vectorstores import Chroma
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import RetrievalQA
from langchain_community.llms import Ollama

# 1. Charger le document
loader = TextLoader("mon_document.txt")
documents = loader.load()

# 2. Diviser le texte en "chunks"
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
chunks = text_splitter.split_documents(documents)

# 3. Créer les embeddings et le Vector Store
embeddings = OllamaEmbeddings(model="llama3")
vector_store = Chroma.from_documents(chunks, embeddings)

# 4. Configurer le modèle de langage
llm = Ollama(model="llama3")

# 5. Créer la chaîne RAG
qa_chain = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=vector_store.as_retriever()
)

# 6. Poser une question
question = "Quel est le sujet principal du document ?"
```

```
response = qa_chain.invoke({"query": question})  
print(response["result"])
```

## Exemple concret : Assistant de cours

### Cas d'usage :

Un étudiant doit analyser une thèse de 350 pages pour préparer un examen.

### Solution :

Mise en place d'un système RAG local avec la thèse comme base de connaissances.

### Résultats :

- **\*\*Avant :\*\*** 3-4 heures de recherche manuelle.
- **\*\*Après :\*\*** 8 minutes pour obtenir des réponses précises et sourcées.
- **\*\*Gain de temps :\*\*** -82% !

L'IA locale transforme la productivité académique.

## Problèmes courants & Optimisations

### Problèmes fréquents






- **\*\*Erreur GPU :\*\*** Pilotes non à jour, VRAM insuffisante.
- **\*\*Modèle lent :\*\*** Modèle trop grand pour le matériel, pas d'accélération GPU.
- **\*\*Réponses imprécises :\*\*** Mauvaise qualité des données, chunking inadapté.
- **\*\*Ollama non trouvé :\*\*** Chemin d'accès incorrect, service non démarré.

### Astuces d'optimisation







- **\*\*Chunking :\*\*** Ajuster `chunk\_size` et `chunk\_overlap`.
- **\*\*Cache :\*\*** Utiliser un cache pour les embeddings et les réponses.
- **\*\*GPU :\*\*** S'assurer que l'accélération GPU est active.
- **\*\*Modèle :\*\*** Choisir un modèle adapté à votre matériel.

## Comparaison Local vs Cloud

### IA Locale

-  Confidentialité maximale
-  Coûts maîtrisés (investissement initial)
-  Indépendance (pas de dépendance externe)
-  Scalabilité limitée par le matériel
-  Nécessite des compétences techniques

### IA Cloud

-  Scalabilité illimitée
-  Facilité de déploiement
-  Accès à des modèles de pointe
-  Coûts récurrents élevés
-  Dépendance à un fournisseur tiers
-  Questions de confidentialité des données

# Conclusion : Lancez-vous !

L'IA locale est une technologie accessible et puissante qui vous offre contrôle et confidentialité.

Commencez par un petit projet, expérimentez et découvrez son potentiel !

## **Merci ! Questions ?**

N'hésitez pas à poser vos questions.

Contact : [votre.email@example.com](mailto:votre.email@example.com)