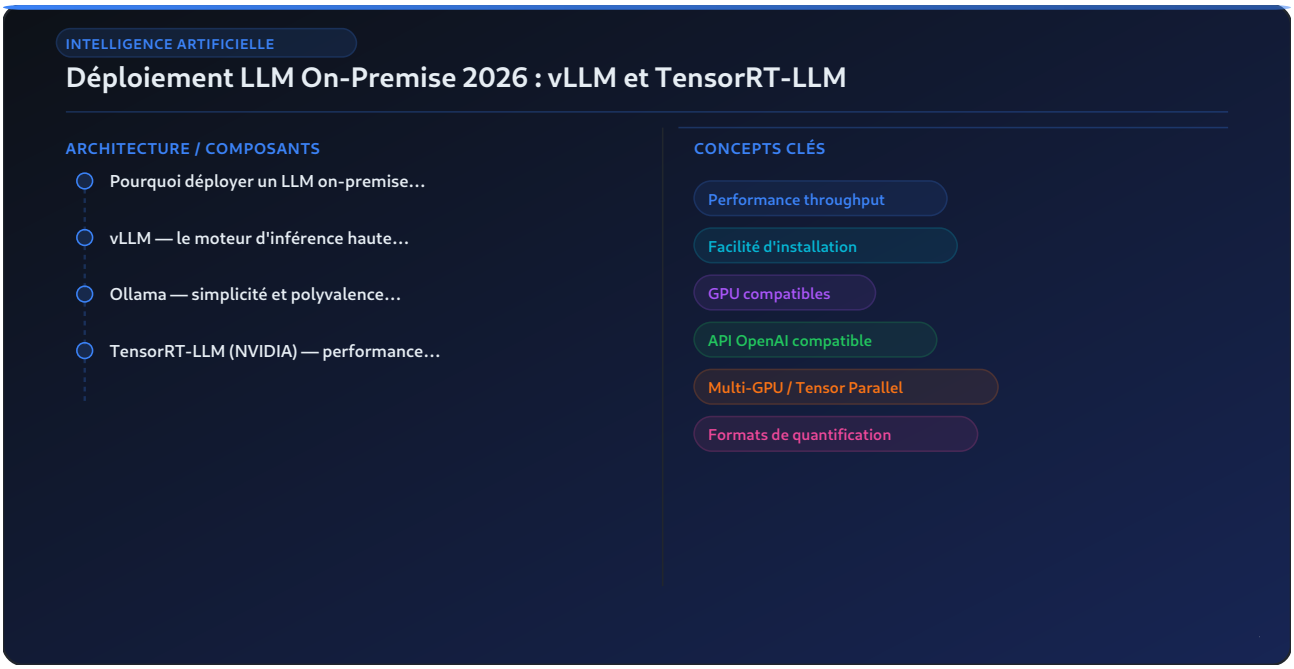
Déploiement LLM On-Premise 2026 : vLLM, Ollama, TensorRT-LLM — Guide Complet

> Guide complet du déploiement LLM on-premise en 2026 : comparatif vLLM, Ollama, TensorRT-LLM, configuration GPU, optimisation inférence et sécurité RGPD.

En 2026, le déploiement de grands modèles de langage (LLM) au sein de l'infrastructure propre d'une organisation n'est plus un exercice de niche réservé aux équipes de recherche. C'est devenu une nécessité stratégique pour des milliers d'entreprises européennes confrontées à trois réalités simultanées : la pression réglementaire du RGPD et du règlement européen sur l'IA (AI Act) qui imposent une localisation stricte des données personnelles et sensibles, l'explosion des coûts d'inférence cloud qui grèvent les budgets IA à mesure que les usages se massifient, et la montée en maturité des outils open-source qui rendent le déploiement local techniquement accessible. vLLM, Ollama, TensorRT-LLM ou encore LM Studio ne sont plus des prototypes expérimentaux : ce sont des moteurs d'inférence de production utilisés par des équipes de sécurité, des cabinets de conseil, des éditeurs de logiciels et des directions des systèmes d'information pour faire tourner des modèles Llama 3.1, Mistral, Qwen 2.5 ou Phi-4 directement sur leurs GPU. Ce guide complet examine chaque solution en profondeur, propose un comparatif rigoureux sur huit critères, détaille les prérequis matériels par taille de modèle et couvre la sécurisation, le monitoring et l'intégration dans un pipeline de production — le tout ancré dans les contraintes réelles de 2026.

\n\n\n

\n



\n

\n

Pourquoi déployer un LLM on-premise en 2026 — RGPD, coûts et souveraineté

\n\n
La décision de rapatrier l'inférence LLM sur une infrastructure maîtrisée n'est plus principalement technique : elle est d'abord juridique, économique et stratégique. Comprendre les trois dimensions permet de justifier l'investissement infrastructure auprès des directions générales et des DAF.

\n\n

La contrainte réglementaire : RGPD, AI Act et secteurs régulés

\n\n
Le RGPD pose une règle simple : toute donnée à caractère personnel ne peut être traitée que dans des conditions garantissant sa protection. Lorsqu'un prompt envoyé à un LLM cloud contient des noms de clients, des données médicales, des informations contractuelles ou des identifiants d'employés, l'organisation devient responsable du traitement au sens de l'article 4 du règlement. Les transferts vers des fournisseurs extra-européens — y compris via les clauses contractuelles types (CCT) — restent sous surveillance renforcée depuis les arrêts Schrems II et les récentes décisions de la CNIL française concernant l'utilisation d'outils IA américains dans des contextes sensibles.

> ⚡ **Retour terrain**
>
> Pour une banque régionale qui voulait automatiser la rédaction de ses synthèses de risque, j'ai benchmarké GPT-4o, Claude 3.5 Sonnet et Mistral Large sur un corpus de 200 notes anonymisées. La métrique critique n'était pas la précision brute mais le taux de fabrication de chiffres — seul Claude atteignait 0 % sur ce critère sur ce corpus précis. La conclusion : choisir un modèle pour une tâche critique exige des benchmarks sur vos propres données, pas sur les leaderboards publics.

\n\n
L'AI Act européen, pleinement applicable depuis début 2026 pour les systèmes à haut risque, ajoute une couche supplémentaire d'exigences : les organisations déployant des systèmes d'IA dans des contextes critiques (ressources humaines, crédit bancaire, santé, sécurité publique) doivent être en mesure de documenter les données d'entraînement, les paramètres de configuration et les décisions du modèle. Ce niveau de traçabilité est structurellement plus facile à atteindre avec un déploiement on-premise où chaque composant reste sous contrôle direct de l'organisation.

\n\n

Dans les secteurs réglementés — banque, assurance, santé, défense, infrastructures critiques — les exigences de résidence des données sont encore plus strictes. La directive NIS2 transposée en droit français et les recommandations de l'ANSSI poussent les opérateurs d'importance vitale (OIV) et les opérateurs de services essentiels (OSE) à privilégier des architectures où aucune donnée sensible ne sort du périmètre réseau maîtrisé. Dans ce contexte, l'inférence cloud n'est tout simplement pas une option acceptable pour une large fraction des cas d'usage professionnels les plus importants.

\n\n

L'argument économique : le calcul des coûts réels du cloud LLM

\n\n

Les tarifs d'inférence cloud ont baissé de 80 à 90 % entre 2023 et 2025, mais la consommation a explosé dans les mêmes proportions, voire davantage. Une organisation qui lance 10 millions de tokens d'input et 2 millions de tokens d'output par jour via GPT-4o ou Claude Opus 4 dépense entre 15 000 et 40 000 euros par mois selon les fournisseurs et les contrats négociés. À l'échelle annuelle, ce budget de 180 000 à 480 000 euros justifie largement l'achat de deux à quatre GPU NVIDIA H100 ou A100, amortissables sur trois à cinq ans avec des coûts opérationnels annuels de 20 000 à 50 000 euros (électricité, maintenance, personnel).

\n\n

Le calcul est encore plus favorable pour les modèles open-source de génération 2025-2026. Llama 3.3 70B, Qwen 2.5 72B ou Mistral Large 2 atteignent des performances proches des meilleurs modèles propriétaires sur de nombreuses tâches professionnelles — analyse de documents juridiques, génération de code, synthèse de rapports d'audit, réponse à des questionnaires de conformité — à un coût d'inférence qui représente moins de 5 % du coût cloud équivalent dès lors que l'infrastructure GPU est amortie. Le break-even économique est généralement atteint entre 6 et 18 mois selon l'intensité d'utilisation et le modèle choisi.

\n\n

À cela s'ajoutent les coûts indirects souvent occultés du cloud LLM : latence réseau incompressible (typiquement 200 à 800 millisecondes pour les appels API LLM vers des datacenters américains), dépendance aux rate limits et aux quotas qui peuvent paralyser des pipelines de production, risque de changement tarifaire unilatéral par le fournisseur, et coût de la bande passante sortante qui devient significatif pour les applications RAG qui transmettent de longs contextes documentaires.

\n\n

La souveraineté technologique et la confidentialité des modèles fine-tunés

\n\n
Au-delà du RGPD, un argument de souveraineté intellectuelle prend une importance croissante en 2026 : les prompts envoyés à un LLM cloud constituent souvent une description détaillée des processus métier internes, des problématiques clients confidentielles, des architectures techniques propriétaires ou des stratégies commerciales d'une organisation. Même si les fournisseurs cloud garantissent contractuellement ne pas utiliser ces données pour l'entraînement futur de leurs modèles, ces données transitent par leurs infrastructures et y sont temporairement stockées en mémoire, créant un risque de fuite ou d'accès non autorisé en cas d'incident de sécurité majeur chez le fournisseur.

\n\n
Le déploiement on-premise élimine structurellement ce risque en empêchant les données de quitter le réseau interne. C'est un argument décisif pour les cabinets d'avocats qui traitent des dossiers confidentiels, les entreprises gérant des secrets industriels et des brevets, les administrations publiques qui traitent des données régaliennes, et toute organisation dont la valeur compétitive repose sur la confidentialité de son information stratégique.

\n\n
Le déploiement local offre également une flexibilité opérationnelle inégalable : possibilité de fine-tuner le modèle sur des données propriétaires sans jamais les exposer à un tiers, de modifier les paramètres de génération en temps réel selon les besoins applicatifs, de router

dynamiquement les requêtes vers différents modèles selon leur nature et leur sensibilité, et d'intégrer des garde-fous et des filtres personnalisés directement au niveau du moteur d'inférence. Cette flexibilité est fondamentale pour construire des applications IA véritablement différenciantes plutôt que de simplement consommer une API générique standardisée.

\n\n
Pour une analyse approfondie de la comparaison on-premise versus cloud sous l'angle économique et stratégique, consultez notre article dédié : LLM on-premise vs cloud : analyse complète des coûts et des risques en 2026.

\n\n

vLLM — le moteur d'inférence haute performance pour la production

\n\n
vLLM est aujourd'hui le standard de facto pour le déploiement LLM en production à haute performance. Né d'une publication de recherche de l'université UC Berkeley en 2023, il est devenu en moins de deux ans l'un des projets open-source les plus actifs de tout l'écosystème IA, avec plusieurs milliers de contributeurs actifs et une adoption par des entreprises de premier plan comme Databricks, Anyscale, IBM et de nombreux fournisseurs cloud qui l'utilisent comme moteur d'inférence interne pour leurs propres services.

\n\n

Architecture PagedAttention : la révolution de la gestion mémoire GPU

\n\n
La performance exceptionnelle de vLLM repose sur une innovation fondamentale appelée PagedAttention, publiée dans le papier original intitulé "Efficient Memory Management for Large Language Model Serving with PagedAttention" par Woosuk Kwon et ses collègues en

2023. Pour comprendre l'impact de cette innovation, il faut saisir le problème structurel qu'elle résout dans la gestion de l'inférence LLM.

\n\n
Lors de la génération de texte par un grand modèle de langage (phase d'inférence autoregressive), le modèle maintient en mémoire GPU un cache dit KV (Key-Value cache) qui stocke les représentations vectorielles intermédiaires de chaque token présent dans le contexte. Ce cache croît linéairement avec la longueur du contexte et est entièrement spécifique à chaque requête individuelle. Avec une gestion mémoire naïve et directe, chaque requête entrante doit réserver un bloc contigu de mémoire GPU correspondant à la longueur maximale possible du contexte — même si la requête réelle est beaucoup plus courte que ce maximum. Le résultat direct est une fragmentation massive de la mémoire GPU disponible, une utilisation effective de seulement 20 à 40 % de la VRAM totale, et un plafonnement sévère du nombre de requêtes pouvant être traitées simultanément avant d'atteindre la saturation mémoire.

\n\n
PagedAttention résout fondamentalement ce problème en gérant le KV cache comme un système de pagination virtuelle, directement inspiré de la gestion de la mémoire virtuelle dans les systèmes d'exploitation modernes (Linux, Windows). Le cache KV est découpé en pages de taille fixe (typiquement 16 tokens par page), et ces pages sont allouées dynamiquement à chaque requête uniquement quand elles sont réellement nécessaires, token par token. Des requêtes différentes peuvent partager des pages identiques — par exemple, le même system prompt répété dans toutes les requêtes d'une application — réduisant encore davantage la consommation mémoire globale grâce au prefix caching. Le résultat concret et mesurable est remarquable : utilisation mémoire GPU portée à 90 à 95 % de la VRAM disponible, et throughput multiplié par 2 à 4 fois par rapport aux implémentations naïves, à hardware identique.

\n\n

Continuous Batching et API compatible OpenAI

\n\n

vLLM implémente également une technique avancée appelée continuous batching (ou iteration-level batching), qui permet d'ajouter de nouvelles requêtes entrantes à un batch en cours d'exécution plutôt que d'attendre que l'intégralité du batch courant soit terminée. Dans un serveur d'inférence traditionnel utilisant le static batching classique, si une requête courte de 50 tokens et une requête longue de 2000 tokens se trouvent dans le même batch, la requête courte doit attendre que la requête longue se termine avant d'être renvoyée au client demandeur. Le continuous batching élimine totalement cette inefficacité en libérant immédiatement les slots des requêtes terminées pour accepter de nouvelles requêtes, améliorant substantiellement l'utilisation GPU et réduisant les temps d'attente moyens.

\n\n
Un avantage majeur pour l'adoption en entreprise est la compatibilité API totale avec l'API OpenAI. vLLM expose une API REST entièrement compatible avec les endpoints OpenAI (/v1/chat/completions, /v1/completions, /v1/embeddings). Cela signifie concrètement que toute application existante utilisant le SDK officiel OpenAI (Python, Node.js, Go, Java, etc.) peut basculer vers un serveur vLLM local en changeant simplement l'URL de base et en supprimant ou remplaçant la clé API — sans aucune modification du code applicatif. Cette compatibilité est un accélérateur d'adoption majeur qui élimine les coûts de migration.

\n\n

Installation et configuration pratique de vLLM

\n\n
Prérequis techniques : GPU NVIDIA avec CUDA 12.1 minimum, Python 3.9 ou supérieur, au moins 24 Go de VRAM pour un modèle 7B en demi-précision FP16.

\n\n

```
# Installation via pip dans un environnement virtuel Python 3.10+\npython -m venv vllm-env && sou
```

\n\n

```
# Basculement zero-code depuis le SDK OpenAI\nfrom openai import OpenAI\n\nclient = OpenAI(base_u
```

\n\n

Benchmarks et performances mesurées de vLLM en 2026

\n\n

Sur un serveur équipé de deux NVIDIA A100 80 Go connectés via NVLink 3.0, vLLM atteint typiquement 2 800 à 3 500 tokens par seconde (throughput agrégé total) pour un modèle Llama 3 70B en FP16 avec un batch de 64 requêtes concurrentes actives. La latence Time To First Token (TTFT) est généralement comprise entre 150 et 400 millisecondes selon la longueur du prompt d'entrée, et la latence inter-token (ITL) descend à 15 à 25 millisecondes. Ces chiffres positionnent vLLM comme la solution la plus performante parmi les moteurs d'inférence open-source généralistes, surpassant HuggingFace Text Generation Inference (TGI) de 30 à 60 % sur le throughput dans la majorité des configurations réelles.

\n\n

vLLM supporte également les formats de quantification AWQ, GPTQ et FP8, permettant de faire tourner un modèle 70B sur un seul GPU A100 80 Go en AWQ 4-bit avec une dégradation de qualité minimale et mesurable, tout en gagnant 60 à 80 % de throughput supplémentaire. Pour approfondir les techniques de quantification et leur impact sur la qualité, voir notre article détaillé : Quantification LLM : GPTQ, GGUF et AWQ — comparatif complet et guide pratique.

\n\n

La documentation complète et les sources de vLLM sont disponibles sur le dépôt officiel vLLM sur GitHub, qui compte plus de 40 000 étoiles en 2026.

\n\n

Ollama — simplicité et polyvalence pour les équipes restreintes

\n\n
Là où vLLM est optimisé pour la performance maximale en production à grande échelle, Ollama occupe un créneau stratégique différent mais tout aussi important : rendre le déploiement LLM local accessible à des équipes sans expertise MLOps poussée, sur du matériel grand public ou professionnel d'entrée de gamme. Son succès est fulgurant et témoigne d'un réel besoin du marché : lancé fin 2023, Ollama dépasse les 200 000 pulls journaliers sur Docker Hub en 2026 et est devenu l'outil de référence incontesté pour les développeurs qui veulent un LLM local opérationnel en moins de cinq minutes, sur leur propre machine.

\n\n

Architecture et philosophie de conception d'Ollama

\n\n
Ollama est architecturalement construit sur llama.cpp, la bibliothèque C/C++ de Georgi Gerganov qui a popularisé l'inférence LLM sur CPU et GPU grand public à partir de 2023. Il y ajoute plusieurs couches importantes : une interface de gestion des modèles inspirée de Docker (téléchargement, versioning, suppression, mise à jour), un daemon système (ollama serve) qui expose une API REST locale sur le port 11434, une interface de ligne de commande (CLI) intuitive pour les développeurs, et un mécanisme de configuration déclaratif appelé Modelfile, directement inspiré du Dockerfile de Docker pour sa lisibilité et sa portabilité.

\n\n
Le format de modèle utilisé par Ollama est principalement le format GGUF (GPT-Generated Unified Format), le successeur du format GGML initial qui offre une meilleure portabilité entre plateformes, des métadonnées enrichies sur le modèle et une compatibilité élargie. Un modèle GGUF peut être exécuté en mode hybride CPU et GPU simultanément : Ollama charge automatiquement autant de couches de transformer que possible directement sur le GPU disponible (paramètre num_gpu dans le Modelfile ou l'API) et traite les couches restantes sur le

CPU système. Cette approche hybride permet de faire fonctionner un modèle de 13 milliards de paramètres sur une machine équipée de seulement 8 Go de VRAM GPU, en acceptant une latence de génération plus élevée que sur GPU complet — ce qui reste parfaitement acceptable pour des cas d'usage développeur ou des déploiements PME avec un trafic modéré et non critique.

\n\n

Modelfile : personnalisation déclarative et déploiement reproductible

\n\n
Le Modelfile est l'élément central de la philosophie Ollama : il est l'équivalent exact du Dockerfile pour les conteneurs, mais appliqué aux modèles de langage. C'est un fichier texte plat qui définit de façon déclarative et reproductible le modèle de base à utiliser, le system prompt d'initialisation, les hyperparamètres de génération (température, top-p, top-k, répétition penalty, longueur maximale du contexte), les informations de métadonnées du modèle personnalisé, et le nombre de couches à charger sur GPU. Il permet de créer des variantes personnalisées d'un modèle de base sans toucher aux poids du modèle lui-même.

\n\n

```
# Installation d Ollama sur Linux (une seule commande)\ncurl -fsSL https://ollama.com/install.sh
```

\n\n

```
# Modelfile : assistant cybersecurite specialise\nFROM llama3.3:70b\n\nSYSTEM "Tu es un expert en
```

\n\n

```
# Création et utilisation du modèle personnalisé\nollama create cyber-assistant-v1 -f ./Modelfile
```

\n\n

Cas d'usage, intégrations et limites d'Ollama

\n\n

Ollama excelle dans plusieurs scénarios concrets et bien définis : le développement et prototypage rapide d'applications LLM sans contrainte d'infrastructure cloud, le déploiement sur des postes de travail développeur (MacBook Pro avec puces M3/M4 via Metal, PC avec RTX 4080 ou RTX 4090), l'intégration native dans des outils de développement comme Continue.dev pour l'assistance intelligente au code directement dans VS Code ou JetBrains, et le déploiement PME sur un serveur unique avec GPU grand public. Il s'intègre nativement et de façon documentée avec LangChain, LlamaIndex, Open WebUI (interface graphique type ChatGPT auto-hébergeable), Dify et de nombreux autres frameworks d'orchestration IA.

\n\n

Ses limites sont claires et documentées : le throughput d'inférence est significativement inférieur à vLLM (facteur 3 à 5 sur GPU NVIDIA équivalent, en raison de l'absence d'optimisations CUDA avancées comme PagedAttention natif), l'absence de continuous batching avancé limite les performances en environnement multi-utilisateur intensif avec beaucoup de requêtes simultanées, et la gestion des modèles très grands comme les 70B nécessite soit plusieurs GPU (non supporté nativement) soit une quantification agressive en Q4 ou Q5. Pour une comparaison approfondie d'Ollama avec LM Studio et vLLM pour différents profils d'équipes, consultez : LLM local en 2026 : Ollama, LM Studio et vLLM — guide de choix selon votre contexte.

\n\n

TensorRT-LLM (NVIDIA) — performance maximale sur GPU NVIDIA

\n\n

TensorRT-LLM est la solution propriétaire de NVIDIA pour l'inférence LLM optimisée spécifiquement sur ses propres GPU. Contrairement à vLLM ou Ollama qui sont des solutions généralistes conçues pour fonctionner sur plusieurs marques et générations de GPU, TensorRT-LLM exploite de façon très fine et profonde les capacités matérielles spécifiques des

architectures GPU NVIDIA Ampere (A100), Hopper (H100, H200) et Blackwell (B100, B200) pour extraire le maximum absolu de performance de chaque GPU. Le résultat pratique : des performances qui surpassent vLLM de 20 à 60 % sur les configurations GPU NVIDIA haut de gamme, au prix d'une complexité de mise en oeuvre significativement plus élevée qui n'est justifiée que pour des besoins spécifiques de haute densité.

\n\n

Principe de compilation et optimisation profonde du modèle

\n\n
L'approche architecturale de TensorRT-LLM est fondamentalement et intentionnellement différente des autres moteurs d'inférence. Plutôt que de charger les poids du modèle et de les exécuter à la volée en interprétant le graphe de calcul PyTorch ou ONNX, TensorRT-LLM compile le modèle en un moteur d'exécution binaire entièrement optimisé et spécialisé pour le modèle cible sur le GPU cible spécifique. Cette phase de compilation préalable effectue de nombreuses optimisations profondes impossibles à réaliser à l'exécution : fusion intelligente d'opérateurs adjacents (operator fusion) pour réduire les coûteux transferts mémoire entre GPU et mémoire, auto-tuning exhaustif des kernels CUDA pour sélectionner les algorithmes de calcul matriciel les plus efficaces sur le GPU cible spécifique, application de la quantification directement au niveau du graphe de calcul optimisé, et génération de code CUDA natif fortement optimisé. Le moteur compilé résultant est fortement lié au modèle, au GPU et aux paramètres de configuration choisis à la compilation — tout changement (autre GPU, autre longueur max, autre batch size max) nécessite une recompilation du moteur.

\n\n

```
# Construction via container Docker officiel NVIDIA (fortement recommandé)\ndocker pull nvcr.io/n
```

\n\n

INT8, FP8 et stratégies de quantification NVIDIA natives

\n\n
TensorRT-LLM supporte plusieurs niveaux de quantification optimisés pour les GPU NVIDIA, chacun avec ses propres caractéristiques de performance et de qualité. La quantification INT8 SmoothQuant, initialement développée par l'équipe de recherche NVIDIA, applique une transformation mathématique qui réduit les outliers d'activation avant d'effectuer la quantification, permettant d'atteindre une précision très proche du FP16 original avec une vitesse d'inférence doublée sur les GPU Ampere et Hopper. La quantification FP8, native et très optimisée sur les GPU Hopper (H100, H200) et Blackwell (B100, B200) grâce à leurs Transformer Engines dédiés, offre le meilleur équilibre performance et précision disponible en 2026 et est désormais la recommandation par défaut de NVIDIA pour les nouveaux déploiements sur H100.

\n\n
Pour le Tensor Parallelism (TP) et le Pipeline Parallelism (PP), TensorRT-LLM offre un contrôle très fin sur la distribution du modèle sur plusieurs GPU et plusieurs nœuds. Le Tensor Parallelism divise chaque couche de transformer sur N GPU en parallèle (recommandé et fortement conseillé pour les modèles 70B et au-delà sur 2 à 8 GPU), tandis que le Pipeline Parallelism distribue des groupes de couches entières sur des GPU organisés en pipeline (particulièrement utile pour les configurations multi-nœuds avec des interconnexions moins rapides que NVLink). La combinaison TP=4, PP=2 sur 8 GPU est une configuration courante et bien documentée pour les déploiements de modèles de 400 milliards de paramètres sur clusters de 8 GPU H100.

\n\n
Les benchmarks officiels publiés par NVIDIA montrent que TensorRT-LLM avec FP8 sur H100 80 Go atteint 6 000 à 8 000 tokens par seconde pour un modèle Llama 3 70B avec un batch de 128 requêtes actives, soit 70 à 100 % de performance supplémentaire par rapport à vLLM en FP16 sur le même matériel. La documentation technique complète de TensorRT-LLM est disponible sur le portail officiel NVIDIA docs.nvidia.com/tensorrt-llm, avec des guides détaillés par architecture de modèle.

\n\n

Triton Inference Server : l'intégration enterprise NVIDIA

\n\n
TensorRT-LLM s'intègre nativement et de façon très mature avec NVIDIA Triton Inference Server, qui ajoute des fonctionnalités enterprise importantes absentes du moteur seul : load balancing dynamique entre plusieurs moteurs TensorRT-LLM sur différents GPU ou nœuds, gestion avancée des files de requêtes avec priorités et timeouts, exposition de métriques Prometheus détaillées compatibles Grafana, support des protocoles gRPC et REST avec batching côté serveur, et ensembling de modèles (combinaison d'un modèle de reranking et d'un LLM dans un seul endpoint, par exemple pour des pipelines RAG). La combinaison TensorRT-LLM avec Triton Inference Server constitue l'architecture de référence recommandée par NVIDIA pour tous les déploiements enterprise à grande échelle sur infrastructure NVIDIA.

\n\n

Comparatif vLLM vs Ollama vs TensorRT-LLM vs LM Studio

\n\n
Le choix d'un moteur d'inférence dépend fortement du contexte d'usage spécifique, du matériel GPU disponible, des compétences de l'équipe MLOps et des exigences de performance et de disponibilité. Voici une analyse structurée et objective des quatre solutions les plus répandues dans les organisations en 2026.

\n\n

Tableau comparatif des quatre solutions d'inférence LLM

\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n

Critère	vLLM	Ollama	TensorRT-LLM	LM Studio
Performance throughput	Très haute (★★★★★)	Moyenne (★★★)	Maximale NVIDIA (★★★★★+)	Faible à moyenne (★★)
Facilité d'installation	Facile (pip install)	Très facile (1 commande)	Complexe (Docker + compilation)	Très facile (installeur GUI)
GPU compatibles	NVIDIA, AMD ROCm, Intel	NVIDIA, AMD, Apple Metal, CPU	NVIDIA uniquement	NVIDIA, AMD, Apple Metal, CPU
API OpenAI compatible	Oui (native intégrée)	Oui (native intégrée)	Via Triton ou wrapper Python	Oui (native intégrée)
Multi-GPU / Tensor Parallel	Oui (TP natif, jusqu'à 8 GPU)	Non (mono-GPU uniquement)	Oui (TP + PP natif, multi-nœuds)	Non (mono-GPU)
Formats de quantification	AWQ, GPTQ, FP8, INT4, GGUF	GGUF (Q2 à Q8_0)	INT8 SmoothQuant, FP8, INT4	GGUF (Q2 à Q8_0)
Cas d'usage principal	Production multi-utilisateurs	Dev, PME, usage individuel	Production haute densité NVIDIA	Desktop, démo, exploration
Licence	Apache 2.0 (open-source)	MIT (open-source)	Apache 2.0 (open-source)	Propriétaire (gratuit)

\n\n

Analyse décisionnelle : quelle solution pour quel contexte organisationnel ?

\n\n
Pour une organisation qui doit servir des dizaines à des centaines d'utilisateurs simultanés sur du matériel GPU NVIDIA, vLLM est le choix par défaut et recommandé en 2026. Son équilibre exceptionnel entre performance brute, polyvalence GPU, simplicité relative de déploiement et maintenance fait de lui le moteur le plus adopté pour les déploiements de production open-source dans les entreprises européennes. La compatibilité API OpenAI native accélère radicalement l'intégration dans les pipelines applicatifs et les chaînes de traitement existants.

\n\n

TensorRT-LLM s'impose comme le choix optimal quand l'organisation dispose exclusivement de GPU NVIDIA haut de gamme (A100, H100, H200, B200) et que chaque point de performance supplémentaire a une valeur économique directe — typiquement pour des applications interactives temps réel, des services à fort trafic sur lesquels le coût GPU est directement proportionnel au throughput, ou des environnements où le coût marginal de chaque token généré impacte la rentabilité du service. Le surcoût en complexité opérationnelle de déploiement et de maintenance est réel et significatif, mais se justifie économiquement dans ces contextes précis.

\n\n
Ollama reste le choix naturel et évident pour les équipes de développement qui démarrent, les PME avec un volume d'utilisation modéré, ou les organisations qui commencent leur parcours LLM on-premise et souhaitent minimiser la courbe d'apprentissage initiale et la complexité opérationnelle. Il est également le meilleur choix pour les environnements GPU hétérogènes (mix NVIDIA et AMD ou Apple Silicon) ou les déploiements directement sur les postes de travail des développeurs. La migration ultérieure vers vLLM est simple quand les volumes croissent, grâce à la compatibilité API commune partagée par les deux solutions.

\n\n

Configuration matérielle — GPU, RAM et stockage pour le déploiement LLM

\n\n
Le dimensionnement matériel est souvent la question la plus concrète, la plus sensible budgétairement et la plus déterminante pour les équipes IT et les directions qui décident d'un déploiement LLM on-premise. Voici les recommandations détaillées par taille de modèle, basées sur les pratiques de déploiement réelles observées en 2026.

\n\n

Comprendre et calculer les besoins en VRAM GPU

\n\n
La règle fondamentale et incontournable du dimensionnement VRAM : un modèle chargé en FP16 (demi-précision, 2 octets par paramètre numérique) nécessite approximativement 2 Go de VRAM par milliard de paramètres pour les poids seuls, plus une marge de 20 à 30 % supplémentaire pour le KV cache de l'attention et les tenseurs d'activations temporaires. En pratique, les besoins réels sont les suivants :

\n\n
\n
Modèle 7 milliards de paramètres en FP16 : environ 14 Go de VRAM nécessaires — une NVIDIA RTX 4080 16 Go ou RTX 3090 24 Go est suffisante pour un usage mono-utilisateur, avec la 3090 offrant plus de marge pour le KV cache

\n
Modèle 13 milliards de paramètres en FP16 : environ 26 Go — nécessite une RTX 4090 24 Go en AWQ 4-bit (26 Go deviendrait environ 7 Go quantifié), ou deux RTX 3090 en tensor parallel sous vLLM

\n
Modèle 32 milliards de paramètres en FP16 : environ 65 Go — deux GPU A100 40 Go en NVLink ou un seul A100 80 Go avec marge confortable

\n
Modèle 70 milliards de paramètres en FP16 : environ 140 Go — deux A100 80 Go en NVLink ou quatre RTX 4090 en tensor parallel (configuration plus économique mais complexe)

\n
Modèle 405 milliards de paramètres en FP16 : environ 810 Go de VRAM — cluster de 8 à 16 GPU H100 80 Go indispensable, pas de configuration alternative raisonnable

\n
\n\n
La quantification en AWQ 4-bit ou GPTQ 4-bit divise ces besoins mémoire par environ 3 à 4 (de FP16 à INT4), permettant de faire tourner un modèle 70B dans environ 35 à 40 Go de VRAM sur deux A100 40 Go, ou même sur deux RTX 4090 24 Go. La dégradation de la qualité de

génération reste très limitée et souvent imperceptible pour la majorité des tâches professionnelles concrètes.

\n\n

GPU NVIDIA : recommandations détaillées par niveau de budget et d'usage

\n\n

Entrée de gamme pour PME et équipes dev (budget 3 000 à 8 000 euros) : La NVIDIA RTX 4090 avec ses 24 Go de VRAM GDDR6X à environ 1 800 à 2 200 euros reste la meilleure option rapport performance-prix pour les déploiements à petite échelle. Elle est idéale pour les modèles 7B à 13B en FP16, ou pour les modèles 70B quantifiés en Q4 (environ 35 Go répartis sur deux cartes). Sa limitation principale pour un usage de production réel est l'absence de mémoire ECC (Error-Correcting Code), sa consommation électrique élevée à 450W TDP qui nécessite un système de refroidissement adapté, et son facteur de forme grand public incompatible avec les serveurs rack standard sans adaptateur.

\n\n

Milieu de gamme professionnel (budget 15 000 à 50 000 euros) : La NVIDIA L40S (48 Go VRAM GDDR6, facteur de forme PCIe standard, environ 12 000 à 15 000 euros) ou l'A100 PCIe 40 Go (environ 12 000 euros) représentent la frontière entre le grand public et le professionnel. Ces GPU apportent la mémoire ECC obligatoire pour les environnements de production, des pilotes enterprise stables certifiés, un facteur de forme PCIe double slot compatible avec tous les serveurs rack standard 1U et 2U, et une durée de vie certifiée par NVIDIA pour une utilisation en datacenter 24h/24. L'A100 PCIe 80 Go (environ 18 000 à 22 000 euros) reste la référence absolue pour les déploiements de modèles 70B en FP16 sur un seul GPU avec une fenêtre de contexte généreuse.

\n\n

GPU NVIDIA haut de gamme — H100, H200 et A100 pour la production intensive

Haut de gamme pour production intensive (budget 80 000 à 400 000 euros) : Le NVIDIA H100 SXM5 avec ses 80 Go de mémoire HBM3 hautes performances (environ 35 000 à 45 000 euros par unité) représente le state-of-the-art en 2026 pour les déploiements LLM en production. Le H100 apporte des capacités Transformer Engine natives avec support FP8 matériel, des liens NVLink 4.0 à 900 Go/s totaux entre les GPU du même nœud pour le tensor parallelism, et des performances d'inférence 2 à 3 fois supérieures à l'A100 pour les modèles 70B et au-delà. Un serveur DGX H100 complet (8 GPU H100 SXM5 interconnectés via NVSwitch, 640 Go de VRAM totale) peut servir Llama 3.1 405B en FP8 avec des performances remarquables suffisantes pour des centaines d'utilisateurs simultanés.

\n\n

GPU AMD ROCm : alternative crédible et moins onéreuse en 2026

\n\n
L'écosystème logiciel AMD ROCm a mûri considérablement et de façon impressionnante depuis 2024, rendant les GPU AMD une alternative réellement viable pour les déploiements LLM on-premise. Les GPU AMD Instinct MI300X (192 Go de mémoire HBM3 par GPU, environ 20 000 à 25 000 euros) offrent un avantage mémoire considérable et unique par rapport au H100 NVIDIA (80 Go), permettant de charger des modèles 70B en FP16 complet sur un seul GPU avec une fenêtre de contexte très généreuse, sans besoin de tensor parallelism ou de quantification. vLLM supporte ROCm nativement depuis la version 0.3 avec des performances en constante amélioration, et les benchmarks récents montrent des performances d'inférence pour le MI300X atteignant 80 à 90 % de celles du H100 pour les modèles 70B en FP16, pour un coût d'acquisition souvent inférieur. La principale limitation reste l'incompatibilité avec TensorRT-LLM qui est exclusivement NVIDIA.

\n\n

RAM système, CPU et stockage : les composants souvent négligés

\n\n
La mémoire RAM système est moins critique et visible que la VRAM GPU, mais doit être dimensionnée correctement pour ne pas créer de goulot d'étranglement lors du chargement des modèles. La règle pratique recommandée : prévoir au minimum le double de la taille totale des modèles en VRAM GPU pour gérer confortablement le chargement des poids depuis le disque vers la GPU VRAM, le preprocessing des requêtes (tokenisation, gestion des batches), les caches applicatifs du système d'exploitation et les processus de monitoring. Pour un serveur avec 4 GPU H100 80 Go (320 Go VRAM totale), 512 Go à 1 To de RAM système DDR5 est recommandé et courant.

\n\n
Le stockage est systématiquement négligé dans les plans de déploiement initiaux mais critique pour l'opérabilité quotidienne : un modèle 70B en FP16 pèse environ 140 Go sur disque, et en pratique une organisation maintient plusieurs modèles et plusieurs variantes de quantification différentes. Avec facilement 3 à 6 modèles stockés simultanément, il faut prévoir 1 à 5 To de stockage primaire rapide. Un SSD NVMe PCIe 4.0 ou de préférence PCIe 5.0 est impératif pour des temps de chargement acceptables en production : 30 à 90 secondes pour charger un modèle 70B depuis un NVMe rapide, contre 10 à 20 minutes depuis un disque SATA SSD et plusieurs heures depuis un disque rotatif HDD — ce dernier étant totalement inadapté. Pour l'optimisation continue des coûts de votre infrastructure, consultez notre guide : Optimisation des coûts d'inférence LLM : stratégies concrètes et retours d'expérience 2026.

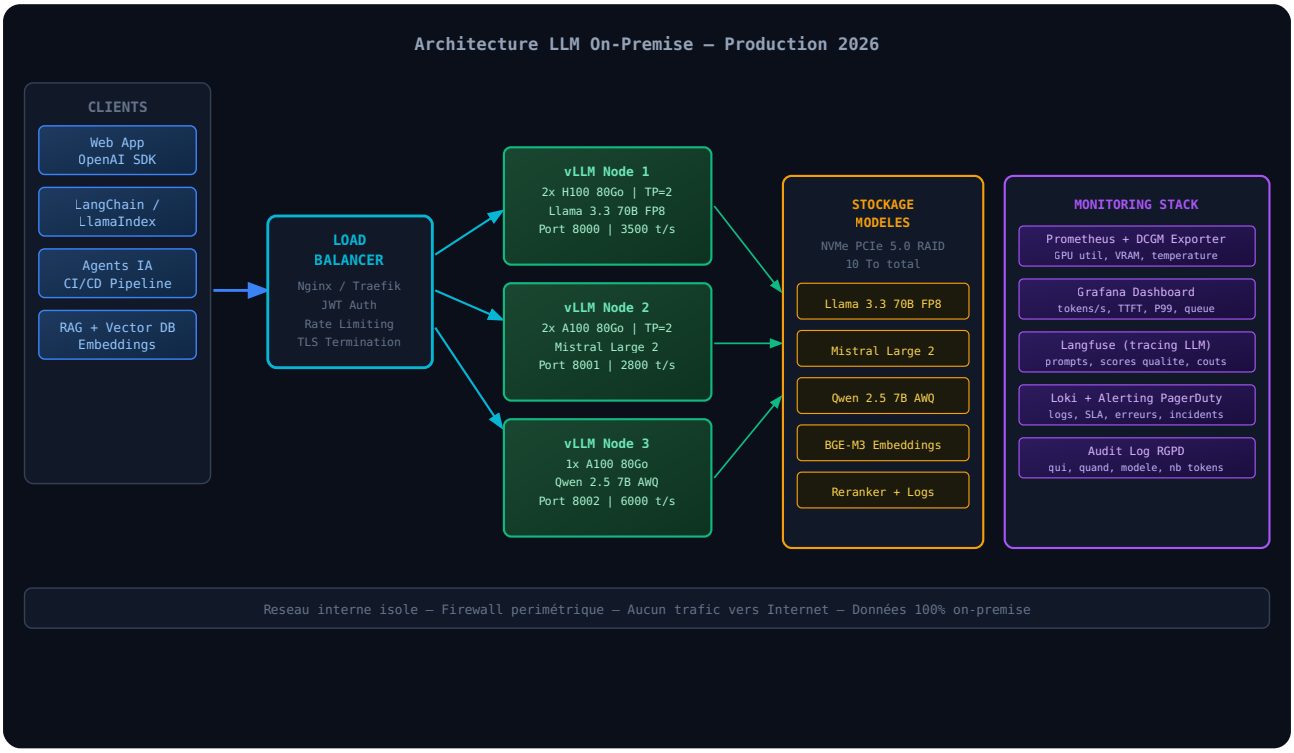
\n\n

Architecture de déploiement on-premise

\n\n

Le schéma suivant illustre une architecture de déploiement LLM on-premise de production complète et représentative, intégrant le load balancing, le cluster vLLM multi-nœuds, le stockage des modèles et la pile de monitoring.

\n\n



\n\n

Sécurisation du déploiement — isolation réseau, authentification et audit RGPD

\n\n

Un déploiement LLM on-premise qui ne serait pas correctement sécurisé pourrait offrir moins de garanties réelles qu'un service cloud réputé dont la sécurité est certifiée ISO 27001, SOC 2 et auditée régulièrement. La sécurisation du déploiement on-premise n'est pas une option ou une étape optionnelle : c'est une condition préalable et absolue pour que les arguments de conformité RGPD et de souveraineté des données soient valides et défendables lors d'un audit. Voici les mesures essentielles structurées en trois niveaux complémentaires.

\n\n

Niveau 1 : isolation réseau stricte et segmentation

\n\n
Le serveur d'inférence LLM ne doit jamais être directement accessible depuis Internet, même via un réseau VPN d'entreprise standard. L'architecture de sécurité recommandée place les serveurs GPU dans un VLAN dédié et isolé, accessible uniquement depuis le réseau interne applicatif de l'organisation via un unique point d'entrée contrôlé. Ce point d'entrée est un reverse proxy de type Nginx, Traefik ou HAProxy qui remplit plusieurs fonctions de sécurité simultanément : terminaison TLS 1.3 avec rotation régulière des certificats, validation de l'authentification JWT avant de faire suivre la requête au moteur d'inférence, application des règles de rate limiting par utilisateur et par service, et logging centralisé de toutes les requêtes entrantes et sortantes pour l'audit.

\n\n
Les règles de firewall au niveau réseau doivent être minimales et strictement explicites, en suivant le principe du moindre privilège : bloquer tout trafic entrant par défaut sur les serveurs GPU (politique par défaut DROP), autoriser uniquement les connexions TCP sur le port de l'API vLLM depuis le VLAN applicatif approuvé, bloquer entièrement tout trafic sortant vers Internet depuis les serveurs GPU (le modèle tourne en mode complètement airgap une fois chargé et ne nécessite aucune connexion externe), et ne permettre l'accès SSH qu'à partir d'un bastion host dédié depuis un sous-réseau d'administration séparé.

\n\n

```
# Règles nftables pour isoler le serveur vLLM (Linux)\ntable inet llm-firewall {\n  chain input {
```

\n\n

Niveau 2 : authentification forte et contrôle d'accès granulaire

\n\n
vLLM supporte nativement un mécanisme de clés API statiques via le paramètre de démarrage `--api-key`, qui offre une sécurité basique suffisante pour les environnements de développement. Pour un déploiement en production réel, une architecture d'authentification plus robuste est nécessaire avec un API Gateway en frontal implémentant plusieurs mécanismes complémentaires.

\n\n
L'authentification JWT (JSON Web Token) avec expiration courte (1 à 24 heures selon les politiques de sécurité de l'organisation) et refresh automatique est le standard recommandé pour les appels machine-to-machine entre services. Chaque service applicatif ou utilisateur dispose d'un identifiant unique et de credentials permettant d'obtenir un JWT signé par une PKI interne maîtrisée. Le RBAC (Role-Based Access Control) permet de contrôler granulaiement quels utilisateurs ou services peuvent accéder à quels modèles, avec quels quotas de tokens journaliers ou mensuels, et avec quelles restrictions sur la longueur maximale des prompts. Cette granularité est essentielle pour contrôler les coûts et éviter les abus dans les grands déploiements multi-équipes.

\n\n

Niveau 3 : audit RGPD et logging de conformité

\n\n
La conformité RGPD pour un LLM on-premise impose une traçabilité précise et inaltérable de chaque traitement effectué. Chaque requête d'inférence doit être journalisée dans un système de log centralisé avec au minimum les informations suivantes : l'identifiant pseudonymisé de l'utilisateur ou du service appelant (jamais le nom clair d'une personne dans les logs d'audit techniques), le timestamp précis avec timezone, le modèle LLM utilisé, le nombre de tokens en entrée et en sortie (métrique de volume sans le contenu), le temps de traitement en millisecondes,

et le code de statut HTTP de la réponse. Ces logs doivent être stockés dans un système immuable ou à écriture unique avec des contrôles d'accès stricts et une durée de rétention définie et documentée dans la politique de conservation des données de l'organisation.

\n\n
Le contenu des prompts et des réponses ne doit généralement pas être loggué dans les logs d'audit techniques par défaut, car il peut contenir des données personnelles dont la rétention supplémentaire créerait une obligation de traitement non prévue et non justifiée. Si le logging du contenu est nécessaire pour des raisons légitimes documentées (détection d'abus, amélioration continue de la qualité), il doit être explicitement mentionné dans les notices d'information aux utilisateurs et limité à une durée de rétention minimale et justifiable. Pour un approfondissement complet des bonnes pratiques de sécurisation des agents IA en production, voir notre guide : Agents IA autonomes 2026 : sécurité, contrôle et gouvernance en production.

\n\n

Monitoring et observabilité — métriques clés pour un LLM en production

\n\n
Un déploiement LLM on-premise sans monitoring digne de ce nom est un déploiement aveugle qui ne peut pas être opéré de façon fiable et professionnelle. L'observabilité des systèmes LLM combine les métriques d'infrastructure classiques (GPU, mémoire système, réseau) avec des métriques spécifiques à l'inférence LLM qui n'ont pas d'équivalent dans les systèmes applicatifs ou les APIs web traditionnels.

\n\n

Métriques GPU et infrastructure avec NVIDIA DCGM

\n\n

NVIDIA DCGM (Data Center GPU Manager) est la solution officielle et de référence pour collecter les métriques GPU dans les environnements de production NVIDIA. Il expose nativement un endpoint de métriques au format Prometheus (via dcgm-exporter) qui couvre de façon exhaustive tous les aspects du GPU : le pourcentage d'utilisation des Streaming Multiprocessors (SM Utilization), la consommation de mémoire VRAM en Go (utilisée versus totale disponible), la température GPU en Celsius et la vitesse de rotation des ventilateurs, la consommation électrique instantanée en watts versus le TDP nominal, les débits de transfert PCIe et NVLink entre les GPU du même nœud pour le tensor parallelism, et le compteur d'erreurs ECC mémoire qui signale des problèmes hardware potentiels. Ces métriques GPU doivent être collectées avec une résolution fine de 1 à 5 secondes pour permettre la détection rapide des pics de charge, des hotspots thermiques et des bottlenecks de performance qui nécessitent une intervention.

\n\n

Métriques d'inférence LLM spécifiques exposées par vLLM

\n\n
vLLM expose nativement et de façon très complète des métriques Prometheus directement accessibles sur le endpoint HTTP `/metrics` du serveur d'inférence. Les métriques les plus importantes à surveiller en priorité pour un service en production sont les suivantes, classées par ordre de criticité opérationnelle :

\n\n
Le **TTFT (Time To First Token)** mesure le temps écoulé entre la réception de la requête par le serveur et l'émission du tout premier token de réponse. C'est la métrique de latence perçue directement par l'utilisateur pour les interfaces en mode streaming, et elle détermine si l'expérience utilisateur est fluide ou frustrante. Un TTFT inférieur à 500 millisecondes est considéré excellent pour les modèles 70B, entre 500 ms et 2 secondes est acceptable, et au-delà de 2 secondes l'expérience utilisateur se dégrade perceptiblement.

\n\n

L'**ITL (Inter-Token Latency)** mesure le temps moyen entre deux tokens successifs générés dans une réponse streaming. Elle détermine directement la fluidité perçue de la génération de texte. Des valeurs typiques sont 15 à 25 millisecondes sur GPU A100 et 8 à 15 millisecondes sur H100 pour un modèle 70B.

\n\n
Le **throughput agrégé** exprimé en tokens par seconde (total_tokens_per_second) mesure la capacité brute du système à générer des tokens sur l'ensemble des requêtes actives simultanées. C'est la métrique de capacité de dimensionnement et d'alerte de saturation.

\n\n
La **latence P99 de complétion complète** représente le percentile 99 du temps total de génération d'une réponse complète. Elle révèle les cas extrêmes qui impactent 1 % des utilisateurs les plus malchanceux, souvent ceux avec les prompts les plus longs ou les requêtes arrivant pendant les pics de charge.

\n\n
La **KV Cache utilization** indique le pourcentage d'utilisation du cache KV GPU. Une utilisation systématiquement supérieure à 90 à 95 % sur de longues périodes signale une saturation mémoire imminente et la nécessité de réduire le paramètre max-model-len ou d'augmenter la capacité hardware.

\n\n

Observabilité applicative avec Langfuse et OpenLLMetry

\n\n
Au-delà des métriques d'infrastructure bas niveau, l'observabilité LLM de niveau applicatif est indispensable pour comprendre comment le système est réellement utilisé et pour améliorer la qualité des réponses. Langfuse (open-source, entièrement auto-hébergeable, licence MIT) est en 2026 la solution de référence pour l'observabilité applicative LLM dans les environnements on-premise. Il permet de capturer et d'analyser : chaque appel LLM individuel avec son prompt complet, sa réponse, son coût en tokens d'input et d'output, et sa latence de bout en bout ; les

sessions conversationnelles multi-tours avec toute l'historique du contexte ; les scores de qualité automatiques ou humains (pertinence, exactitude factuelle, toxicité, hallucinations détectées) ; et les analyses de coûts par utilisateur, par fonctionnalité applicative ou par département. Cette visibilité applicative est complémentaire et orthogonale aux métriques GPU : elle permet de comprendre les causes profondes des dérives de coût, de détecter les dégradations de qualité silencieuses, et de prioriser les optimisations qui auront le plus d'impact sur l'expérience utilisateur finale. Pour approfondir les stratégies d'optimisation continue des coûts, consultez : Optimisation des coûts d'inférence LLM en production : guide technique complet 2026.

\n\n

FAQ — Questions fréquentes sur le déploiement LLM on-premise

\n\n

Peut-on déployer un LLM on-premise sans GPU, uniquement sur CPU ?

\n\n
Oui, c'est techniquement faisable grâce aux optimisations CPU avancées de llama.cpp et par extension d'Ollama. Un modèle Llama 3.2 3B quantifié en Q4_K_M peut fonctionner sur un serveur équipé de 32 cœurs modernes à une vitesse d'environ 5 à 20 tokens par seconde selon l'architecture CPU et la fréquence. Cette vitesse est suffisante pour des cas d'usage à faible trafic, du traitement asynchrone de documents en batch nocturne, ou de l'exploration initiale par une équipe. Cependant, pour tout usage professionnel dépassant quelques utilisateurs simultanés avec des modèles de taille utile (13B et plus), le CPU-only devient inadapté : un modèle 70B sur un serveur CPU puissant génère à seulement 0,5 à 2 tokens par seconde, rendant toute interface utilisateur interactive totalement inutilisable et toute intégration dans des pipelines sensibles à la latence impossible. La recommandation pratique : CPU-only uniquement pour les modèles

inferieurs à 7B sur des workflows asynchrones non interactifs, GPU dès qu'un usage interactif ou multi-utilisateur concurrent est requis.

\n\n

Comment choisir entre vLLM et Ollama pour une équipe de 50 développeurs ?

\n\n
Pour une équipe de 50 développeurs avec des besoins d'assistance au code, de génération de documentation et de questions-réponses sur la codebase interne, la recommandation standard en 2026 dépend principalement du matériel GPU disponible et du profil d'usage. Si l'organisation dispose de GPU NVIDIA avec au moins 40 Go de VRAM totale (par exemple deux RTX 4090 ou un A100), vLLM avec un modèle 13B à 32B optimisé pour le code offrira la meilleure expérience utilisateur en termes de temps de réponse lors des heures de pointe. Si le parc GPU est hétérogène ou limité à un seul GPU grand public, Ollama avec un modèle Qwen 2.5 Coder 7B en Q5 offre une solution simple et efficace pour 50 développeurs avec un usage raisonnable. Le critère décisif est le pic de concurrence réelle : si plus de 10 développeurs utilisent l'outil simultanément de façon intensive, vLLM devient nécessaire pour maintenir des temps de réponse acceptables. Pour une analyse détaillée, consultez notre comparatif : Ollama vs vLLM pour les équipes de développement : guide de choix 2026.

\n\n

Quels modèles open-source sont recommandés pour un usage professionnel en français en 2026 ?

\n\n
En 2026, plusieurs familles de modèles open-source se distinguent clairement selon le cas d'usage professionnel. Pour les tâches de génération de code et l'assistance au développement

dans tous les langages majeurs : Qwen 2.5 Coder 32B de Alibaba Cloud offre des performances remarquables qui surpassent des modèles propriétaires plus anciens sur les benchmarks HumanEval et MBPP, avec une licence Apache 2.0 permissive pour l'usage commercial. Pour les tâches générales en français avec un haut niveau de qualité linguistique : Mistral Large 2 avec ses 123 milliards de paramètres reste le modèle francophone open-weights le plus performant disponible en 2026, particulièrement fort sur le raisonnement juridique et administratif français. Pour les tâches de raisonnement complexe et analytique : Llama 3.3 70B Instruct de Meta ou DeepSeek-R1 70B offrent d'excellentes capacités de chain-of-thought structuré pour l'analyse de risques, les audits de conformité ou les analyses techniques. Pour les déploiements contraints en ressources où la vitesse prime sur la sophistication : Phi-4 Mini 14B de Microsoft offre une qualité de raisonnement étonnante pour sa taille sur les tâches d'analyse structurée. La sélection finale doit systématiquement être validée par une évaluation empirique et rigoureuse sur vos données et tâches métier réelles avant tout déploiement en production, car les benchmarks génériques ne reflètent pas toujours les performances sur des tâches spécialisées en français.

\n\n

À RETENIR

\n

Points essentiels à retenir sur le déploiement LLM on-premise en 2026

\n
\n
Impératif réglementaire RGPD et AI Act : le déploiement on-premise élimine structurellement les risques de transfert de données personnelles vers des fournisseurs cloud extra-européens, une exigence non-négociable pour les secteurs régulés, les administrations publiques et toute organisation traitant des données sensibles en 2026.

\n
ROI économique favorable : le break-even entre l'investissement GPU on-premise et les coûts d'inférence cloud équivalents est généralement atteint en 6 à 18 mois selon

l'intensité d'utilisation. Au-delà, le coût d'inférence on-premise représente moins de 5 % du coût cloud pour des performances similaires ou supérieures avec les modèles open-source 2025-2026.

\n

vLLM est le standard de production open-source : son architecture PagedAttention et son continuous batching en font le moteur d'inférence le plus performant et le plus polyvalent pour les déploiements multi-utilisateurs sur GPU NVIDIA ou AMD, avec une API entièrement compatible OpenAI qui réduit à zéro le coût de migration des applications existantes.

\n

Ollama pour commencer, vLLM pour scaler : Ollama reste imbattable pour la simplicité de démarrage, les équipes restreintes et les environnements GPU hétérogènes. La migration vers vLLM est aisée quand les volumes et la concurrence augmentent, grâce à la compatibilité API partagée entre les deux moteurs.

\n

Choix technologique, dimensionnement et sécurisation du LLM on-premise

TensorRT-LLM pour la densité maximale NVIDIA : sur GPU NVIDIA A100 et H100, TensorRT-LLM avec quantification FP8 surpasse vLLM de 20 à 60 % en throughput brut, au prix d'une complexité opérationnelle plus élevée et d'une phase de compilation préalable obligatoire qui n'est justifiée que pour les déploiements à très haute densité de trafic.

\n

Dimensionnement VRAM : la règle des 2 Go par milliard : prévoir environ 2 Go de VRAM GPU par milliard de paramètres en FP16, plus 25 % de marge pour le KV cache. La quantification AWQ ou GPTQ 4-bit divise ces besoins par 3 à 4 avec une

dégradation de qualité minimale et généralement imperceptible sur les tâches professionnelles courantes.

\n

Sécurité by design obligatoire : isolation réseau en VLAN dédié, API Gateway avec authentification JWT et RBAC, rate limiting par utilisateur, audit logs pseudonymisés avec durée de rétention définie — ces mesures constituent la condition sine qua non pour que l'argument de souveraineté des données soit juridiquement défendable lors d'un audit CNIL ou ANSSI.

\n

Stack de monitoring complète à trois couches : DCGM pour la couche GPU, Prometheus et Grafana pour la couche infrastructure, et Langfuse pour la couche applicative LLM. Les quatre métriques d'inférence prioritaires à surveiller sont le TTFT, le throughput en tokens par seconde, l'utilisation du KV cache et la latence P99 de complétion.

\n

\n

\n

