

Formation Déployer un LLM en local : inférence haute performance et sécurité

Réf. SF34745

Faites tourner des modèles de langage open source sur votre propre infrastructure, du prototype sur poste au serveur d'inférence de production, rapide et sécurisé.

2 j
Durée

1 800 € HT
Inter / participant

0
Sessions à venir

Objectifs pédagogiques

- ✓ Choisir un modèle à poids ouverts, un format de quantification et une configuration matérielle adaptés à un cas d'usage, à sa licence et à son budget
- ✓ Déployer et configurer un modèle local avec Ollama et llama.cpp pour le prototype
- ✓ Mettre en service un modèle avec vLLM et optimiser son débit et sa latence à partir de mesures de performance
- ✓ Exposer un modèle via une API compatible OpenAI et l'intégrer dans une application existante
- ✓ Sécuriser un service d'inférence local : contrôle d'accès, isolation, filtrage des entrées et sorties, intégrité des modèles
- ✓ Superviser et exploiter un service d'inférence dans la durée (métriques, journalisation, mises à jour des modèles)

Public visé & prérequis

Public visé Développeurs, Ingénieurs Big Data, Ingénieurs DevOps

Prérequis - Pratiquer Python et consommer une API REST - Être à l'aise avec Linux, la ligne de commande et Docker - Connaître le fonctionnement général d'un LLM (prompt, contexte, jetons)

Programme détaillé

Enjeux et dimensionnement d'un déploiement local

- Pourquoi héberger soi-même un LLM : confidentialité des données, souveraineté, maîtrise des coûts et de la latence
- Auto-hébergement ou API : critères de choix et ordre de grandeur du volume à partir duquel l'hébergement local devient rentable
- Modèles à poids ouverts : panorama des familles du marché (Mistral, Llama, Qwen, Gemma...), tailles et licences d'usage en entreprise
- Empreinte mémoire : estimer la VRAM nécessaire à partir du nombre de paramètres, de la précision et de la taille du contexte (cache KV)
- Choix du matériel : GPU, CPU, mémoire unifiée, poste de développement ou serveur dédié selon l'usage visé

Activités pratiques :

- Calcul de la VRAM nécessaire pour plusieurs modèles, précisions et longueurs de contexte
- Comparaison d'un modèle local et d'une API cloud sur un même jeu de prompts (qualité, temps de réponse, coût)

Formats de modèles et quantification

- Formats de poids : safetensors, GGUF et correspondance entre formats et moteurs d'inférence
- Principe de la quantification : réduire la précision des poids pour gagner en mémoire et en vitesse
- Techniques du marché : niveaux de quantification GGUF, AWQ, GPTQ, FP8, et leurs cas d'usage
- Arbitrage taille / vitesse / qualité : mesurer la perte de qualité plutôt que la supposer
- Modèles ou adaptateurs déjà fine-tunés : convertir et charger des poids issus d'un entraînement existant, sans le refaire

Activités pratiques :

- Chargement de plusieurs variantes quantifiées d'un même modèle et mesure de l'empreinte mémoire et de la vitesse
- Évaluation rapide de la qualité des réponses selon le niveau de quantification

Prototyper avec Ollama et llama.cpp

- Ollama : installation, accélération GPU, commandes essentielles et cycle de vie des modèles (téléchargement, mise à jour, suppression)
- Modelfiles : paramètres d'inférence, instructions système et import d'un modèle GGUF ou d'un adaptateur existant
- llama.cpp : serveur intégré, réglage du contexte et répartition des couches entre CPU et GPU
- Interface web de prototypage : tester et comparer des modèles avec les équipes métier
- Limites des moteurs de poste : concurrence, débit multi-utilisateurs et quand passer à un serveur de production

Activités pratiques :

- Lancement du modèle fil rouge sous Ollama puis sous llama.cpp, et comparaison des performances
- Création d'un Modelfile spécialisé sur une tâche métier (instructions système, paramètres)

Inférence haute performance avec vLLM

- Serveur d'inférence de production : rôle de vLLM et alternatives du marché (TGI)
- Continuous batching : servir de nombreuses requêtes simultanées sans dégrader le débit
- PagedAttention et cache KV : gestion de la mémoire GPU, cache de préfixe et longueur de contexte
- Indicateurs de performance : latence du premier jeton, débit en jetons par seconde, concurrence et point de rupture
- Service de plusieurs adaptateurs LoRA sur un même modèle de base : mutualiser le GPU entre plusieurs usages
- Répartition sur plusieurs GPU : principes du parallélisme de tenseurs et de pipeline, et quand y recourir

Activités pratiques :

- Mise en service du modèle fil rouge avec vLLM et réglage des principaux paramètres mémoire
- Benchmark de charge (débit, latence, concurrence) et comparaison avec Ollama
- Rédaction d'une fiche de dimensionnement matériel à partir des mesures

Exposer, intégrer et exploiter le service d'inférence

- API compatible OpenAI : endpoints, streaming, sorties structurées et appel d'outils exposés par les moteurs locaux
- Intégration applicative : brancher une application ou un framework existant (LangChain, LangGraph, MCP) sur le modèle local sans réécrire le code
- Passerelle de modèles (LiteLLM) : point d'accès unique, routage entre modèles, clés et quotas par équipe
- Conteneurisation : packager le service d'inférence avec Docker et Docker Compose
- Supervision : métriques exposées par le serveur, mémoire GPU, latence et taux d'erreur dans un tableau de bord
- Cycle de vie : mise à jour ou changement de modèle, tests de non-régression et suivi des coûts

Activités pratiques :

- Connexion d'une application existante à l'API locale en remplacement d'une API cloud
- Mise en place d'une passerelle LiteLLM devant Ollama et vLLM
- Tableau de bord de supervision du service d'inférence

Sécuriser un déploiement LLM local

- Risques spécifiques aux LLM : panorama OWASP Top 10 LLM (injection de prompt, fuite de données, consommation excessive de ressources...)
- Contrôle d'accès : authentification devant le serveur d'inférence, clés API, RBAC et limitation de débit
- Isolation et chiffrement : cloisonnement réseau, TLS, conteneurs durcis et environnements déconnectés (air-gapped)
- Filtrage des entrées et sorties : garde-fous avec Llama Guard ou NeMo Guardrails face aux injections et aux jailbreaks
- Protection des données sensibles : détection et anonymisation des données personnelles avant traitement
- Chaîne d'approvisionnement des modèles : provenance, formats sûrs, vérification d'intégrité et respect des licences
- Journalisation et traçabilité : conserver les usages utiles à l'audit sans exposer de données sensibles

Pourquoi choisir Sparks Formation ?

20 ans d'expertise IT, des formateurs experts et des sessions garanties contre l'annulation.

20 ans
d'expertise

1 068+
formations

11 178
apprenants formés/an

+67
Net Promoter Score

Avis des participants

4,9/5 — 190 avis

Anne-Lise C. — Très bonne adaptation du formateur au public et au niveau d'attente.

Marie G. — La formatrice était très pédagogue et bienveillante ! Elle a su nous partager sa passion et donner envie de pratiquer à la suite Voilà bien longtemps que je...

Ils nous font confiance

Cette année notre filière Intelligence artificielle & Data Science a accompagné plus de 544 entreprises dont :

SOPRA STERIA THALES GLOBAL SERVICES ARTELIA AXA ASSISTANCE FRANCE GRANT THORNTON FRANCE AKKODIS DIGITAL SAS
MEDIABRANDS CEGEDIM COVAGE INFRA SNCF - E.SNCF SOLUTIONS +534 autres

Financement & certifications

Cette formation est éligible aux financements OPCO et FNE-Formation. Nos équipes vous accompagnent dans le montage de votre dossier.



Réservez votre place dès maintenant

20 ans d'expertise · 1 000+ formations · NPS +67 · réponse sous 24h ouvrées

0 805 950 800

demande@sparks-formation.com