installazione llama.cpp su linux con sk nvidia

Ultimo aggiornamento: 2026-10-07

Installare llama.cpp su Linux con supporto NVIDIA CUDA

Passi pratici per compilare llama.cpp da sorgente con CUDA e avviare un server di inferenza. Verificato con CUDA 13.2.

1. Prerequisiti di sistema

```
sudo apt update
sudo apt install -y build-essential cmake git curl libcurl4-openssl-dev
nvcc --version        # verificare che il compilatore CUDA sia presente
```

Se nvcc non è nel PATH ma CUDA è installata:

```
ls -l /usr/local/cuda*/bin/nvcc
```

Atteso (esempio):

```
-rwxr-xr-x 1 root root 33994064 May  8 20:53 /usr/local/cuda-13.2/bin/nvcc
-rwxr-xr-x 1 root root 33994064 May  8 20:53 /usr/local/cuda-13/bin/nvcc
-rwxr-xr-x 1 root root 33994064 May  8 20:53 /usr/local/cuda/bin/nvcc
```

2. Impostare le variabili CUDA

```
export PATH=/usr/local/cuda-13.2/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-13.2/lib64:$LD_LIBRARY_PATH
export CUDACXX=/usr/local/cuda-13.2/bin/nvcc
```

> Adatta `13.2` alla versione effettiva trovata al passo 1.

3. Clonare e compilare

```
git clone https://github.com/ggml-org/llama.cpp.git
cd llama.cpp
cmake -B build -DGGML_CUDA=ON
(aggiungere -DCMAKE_CUDA_ARCHITECTURES="86;120" se vogliam specificare l'architettura)
cmake --build build --config Release -j$(nproc)
```

4. Verificare che il binario funzioni

```
./build/bin/llama-cli --help
```

5. Avvio rapido del server

```
./build/bin/llama-server \
  -m /path/to/modello.gguf \
  -c 65536 \
  --host 0.0.0.0 \
  --port 8080
```

Flag	Cosa fa
`-m`	Percorso del modello GGUF
`-ngl 999`	Prova a offloadare tutti i layer sulla GPU
`-c 65536`	Dimensione del contesto
`--host`	Ascolta anche sulla rete (es. dalla VM)

Flag	Cosa fa
`--port 8080`	Endpoint del server

6. Verifica dell'endpoint OpenAI-compatible

```
curl -s http://llama-host:8080/v1/models | jq .
```

Endpoint base da usare nelle configurazioni (Ollama, Open WebUI, Assistente AI):
`http://llama-host:8080/v1`

7. Configurazione quasi ottimale con vision (Qwen 27B + mmproj)

Esempio di server con proiettore vision su porta 11434:

```
LLSRV="/localIA/llama/llama.cpp/build/bin/llama-server"
PORT=11434
MODEL="/export/nas/<utente>/ia_models/Qwen3.8.27B.ollama/Qwen3.8-27B-UD-IQ4_XS.gguf"
VISION="/export/nas/<utente>/ia_models/Qwen3.8.27B.ollama/mmproj-F16.gguf"

GGML_CUDA_DISABLE_GRAPHS=1 "$LLSRV" \
  -m "$MODEL" \
  --alias qwen3.8-27b \
  --mmproj "$VISION" \
  --no-mmproj-offload \
  --image-max-tokens 4096 \
  -c 65536 \
  -np 1 \
  --kv-unified \
  -ngl 999 \
  --flash-attn on \
  -ctk q4_0 \
  -ctv q4_0 \
  --device CUDA0 \
  --gpu-layers-draft all \
  --spec-draft-device CUDA0 \
  --spec-type draft-mtp \
  --spec-draft-n-max 1 \
  --spec-draft-type-k f16 \
  --spec-draft-type-v f16 \
  --no-spec-draft-backend-sampling \
  -b 512 \
  -ub 64 \
  --threads 7 \
  --fit off \
  --jinja \
  --host 0.0.0.0 \
  --port "$PORT"
```

> Sostituisci <utente> con il nome della tua cartella condivisa NAS e adatta i percorsi al tuo setup.

P.S. Aggiornamento di release:

ls /usr/local/cuda*/bin/nvcc export PATH=/usr/local/cuda-13.2/bin:$PATH cp -rp llama.cpp llama.cpp.old llama.cpp git status git pull rm -rf build cmake -B build -DGGML_CUDA=ON -DCMAKE_CUDA_ARCHITECTURES="86;120" cmake --build build --config Release -j $(nproc) ./build/bin/llama-cli --version