LLM en local amb Ollama

Índex

Entorn: Ubuntu 26.04 Desktop

1. Introducció

Ollama és una eina de codi obert que permet executar models de llenguatge gran (LLM) directament al teu ordinador, sense necessitat de connexió a internet ni de pagar per API externes. Proporciona una interfície senzilla per descarregar, gestionar i interactuar amb models com **Llama 3**, **Mistral**, **Gemma**, **Phi** i molts d'altres.

Figura 1: Ollama logo

Avantatges principals

- **Privacitat total**: les converses no surten del teu sistema.
- **Sense cost per ús**: una vegada descarregat el model, funciona sense límits.
- **Fàcil gestió**: interfície similar a Docker per als models.
- **API compatible**: exposa una API REST compatible amb OpenAI.

2. Requisits previs

Maquinari mínim

Component	Mínim	Recomanat
RAM	8 GB	16 GB o més
Disc lliure	10 GB	50 GB+ (segons models)
CPU	x86_64 moderna	AVX2 suportat
GPU (opcional)	---	NVIDIA amb 6 GB VRAM+

> **Nota:** Sense GPU, els models funcionen per CPU. Són funcionals però més lents. Per a ús quotidià, un model de 7B paràmetres en CPU és perfectament usable.

Verificació del sistema

```
# Comprova l'arquitectura i la RAM disponible
uname -m
free -h

# Comprova si la CPU suporta AVX2 (recomanat)
grep -o 'avx2' /proc/cpuinfo | head -1

# Espai en disc disponible
df -h ~
```

3. Instal·lació d'Ollama

Mètode ràpid (script oficial)

```
curl -fsSL https://ollama.com/install.sh | sh
```

L'script detecta automàticament el sistema, instal·la els binaris i configura un servei `systemd`.

Verificació de la instal·lació

```
# Comprova la versió instal·lada
ollama --version

# Comprova l'estat del servei
systemctl status ollama
```

La sortida hauria de mostrar `active (running)`.

Inici manual del servei (si cal)

```
# Iniciar el servei
sudo systemctl start ollama

# Activar-lo a l'inici del sistema
sudo systemctl enable ollama

# Consultar logs en temps real
journalctl -u ollama -f
```

Desinstal·lació (si cal en el futur)

```
sudo systemctl stop ollama
sudo systemctl disable ollama
sudo rm /etc/systemd/system/ollama.service
sudo rm $(which ollama)
sudo rm -rf ~/.ollama
```

4. Primers passos: descarregar i executar models

Descarregar i iniciar un model

```
# Descarrega i inicia una conversa amb Llama 3.2 (3B, ~2 GB)
ollama run llama3.2

# Versió més lleugera per a equips amb poca RAM (1B, ~800 MB)
ollama run llama3.2:1b

# Model Mistral 7B (~4 GB, molt recomanat per equilibri
 ↪  qualitat/velocitat)
ollama run mistral

# Model de Google, molt eficient (2B, ~1.5 GB)
ollama run gemma2:2b
```

La primera vegada es descarrega el model. Les vegades següents, arrenca directament.

Sessió interactiva

Un cop dins la sessió interactiva:

```
>>> Explica'm què és una xarxa VLAN en termes senzills.

[resposta del model...]

>>> /bye
```

Ordres especials dins la sessió:

Ordre	Acció
`/bye`	Surt de la sessió
`/clear`	Neteja el context de la conversa
`/set system <text>`	Defineix el missatge de sistema
`Ctrl+D`	Surt de la sessió

5. Ús des de la línia d'ordres

Mode no interactiu (una sola pregunta)

```
# Resposta directa sense obrir sessió interactiva
ollama run mistral "Quina diferència hi ha entre TCP i UDP?"

# Útil per a scripts
RESPOSTA=$(ollama run llama3.2 "Genera 5 preguntes tipus test sobre
 ↪  subnetting")
echo "$RESPOSTA"
```

Ús amb pipe (entrada estàndard)

```
# Analitza el contingut d'un fitxer
cat error.log | ollama run mistral "Analitza aquest log i indica els
 ↪  errors principals:"

# Resumeix un document
cat apunts.md | ollama run llama3.2 "Resumeix aquest text en 5 punts
 ↪  clau:"

# Revisa codi
cat script.sh | ollama run codellama "Revisa aquest script Bash i
 ↪  suggereix millores:"
```

Model especialitzat en codi

```
# CodeLlama: model específic per a programació
ollama run codellama

# Starcoder2 per a tasques de codi
ollama run starcoder2:3b
```

6. API REST d'Ollama

Ollama exposa una API REST al port **11434** que pot ser consumida per qualsevol aplicació.

Verificació que l'API funciona

```
curl http://localhost:11434/
# Ha de retornar: Ollama is running
```

Petició de generació (streaming)

```
curl http://localhost:11434/api/generate \
  -d '{
    "model": "llama3.2",
    "prompt": "Explica el model OSI en 3 línies",
    "stream": false
  }'
```

Petició en format compatible OpenAI

```
curl http://localhost:11434/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "llama3.2",
    "messages": [
      {"role": "system", "content": "Ets un professor d'\''informàtica
      ↪  en català."},
      {"role": "user", "content": "Explica el protocol DHCP"}
    ]
  }'
```

Exemple amb Python

```
pip install ollama
```

```
import ollama

# Conversa senzilla
resposta = ollama.chat(
    model='llama3.2',
    messages=[
        {'role': 'user', 'content': 'Explica el protocol SSH en 3
        ↪  punts'}
    ]
)
print(resposta['message']['content'])
```

Exemple amb Python (streaming)

```
import ollama

for chunk in ollama.chat(
    model='llama3.2',
    messages=[{'role': 'user', 'content': 'Explica iptables'}],
    stream=True
):
    print(chunk['message']['content'], end='', flush=True)
```

7. Gestió de models

Llistar models disponibles en local

```
ollama list
```

Exemple de sortida:

```
NAME                    ID              SIZE    MODIFIED
llama3.2:latest         a80c4f17acd5    2.0 GB  2 hours ago
mistral:latest          f974a74358d6    4.1 GB  1 day ago
gemma2:2b               8ccf136fdd52    1.6 GB  3 days ago
```

Descarregar un model sense executar-lo

```
ollama pull phi3:mini
ollama pull nomic-embed-text    # model per a embeddings
```

Eliminar un model

```
ollama rm mistral
```

Informació detallada d'un model

```
ollama show llama3.2
```

Models recomanats per a ús docent

Model	Mida	RAM mínima	Ús recomanat
`llama3.2:1b`	800 MB	4 GB	Proves ràpides, equips vells
`llama3.2`	2 GB	8 GB	Ús general, molt equilibrat
`mistral`	4.1 GB	8 GB	Tasques complexes, codi
`gemma2:2b`	1.6 GB	6 GB	Ràpid i eficient
`codellama`	3.8 GB	8 GB	Generació i anàlisi de codi
`phi3:mini`	2.3 GB	8 GB	Excel·lent relació qualitat/mida

Explorar el catàleg de models

Visita ollama.com/library per veure tots els models disponibles.

8. Interfície web amb Open WebUI

Open WebUI proporciona una interfície gràfica similar a ChatGPT que es connecta a Ollama.

Instal·lació amb Docker

```
# Instal·la Docker si no el tens
sudo apt install -y docker.io
sudo usermod -aG docker $USER
newgrp docker

# Executa Open WebUI connectat a Ollama local
docker run -d \
  -p 3000:8080 \
  --add-host=host.docker.internal:host-gateway \
  -v open-webui:/app/backend/data \
  --name open-webui \
  --restart always \
  ghcr.io/open-webui/open-webui:main
```

Accedeix a `http://localhost:3000` al navegador.

> La primera vegada et demanarà crear un compte d'administrador local (sense connexió externa).

Sense Docker (pip)

```
pip install open-webui
open-webui serve
```

Accedeix a `http://localhost:8080`.

9. Acceleració GPU (opcional)

NVIDIA (CUDA)

L'script d'instal·lació d'Ollama detecta automàticament les GPU NVIDIA si els drivers estan instal·lats.

```
# Comprova si CUDA és detectat
nvidia-smi

# Comprova que Ollama usa la GPU
ollama run llama3.2 "test"
# Observa a nvidia-smi si la VRAM augmenta
```

AMD (ROCm)

```
# Instal·la els drivers ROCm primer
# https://rocm.docs.amd.com/en/latest/deploy/linux/quick_start.html

# L'script d'Ollama detecta ROCm automàticament
curl -fsSL https://ollama.com/install.sh | sh
```

Verificar quin dispositiu s'usa

```
# Comprova els logs d'Ollama
journalctl -u ollama --since "1 minute ago"
# Cerca línies com: "llm runner started ... gpu=cuda"
```

10. Resolució de problemes comuns

El servei no arrenca

```
# Comprova l'estat
systemctl status ollama

# Mira els logs complets
journalctl -u ollama -n 50
```

Error “out of memory”

El model no cap a la RAM disponible. Solucions:

```
# Usa un model més petit
ollama run llama3.2:1b    # en lloc de llama3.2

# O redueix el context màxim
ollama run mistral --num-ctx 2048
```

La descàrrega s'interromp

```
# Torna a executar la mateixa ordre; Ollama reprèn la descàrrega
ollama pull llama3.2
```

Canviar el directori on es guarden els models

Per defecte els models es guarden a `~/.ollama/models`. Per canviar-ho:

```
# Edita la configuració del servei
sudo systemctl edit ollama
```

Afegeix dins el fitxer:

```
[Service]
Environment="OLLAMA_MODELS=/ruta/alternativa/models"
```

```
sudo systemctl daemon-reload
sudo systemctl restart ollama
```

Permetre accés des d'altres màquines de la xarxa

Per defecte Ollama escolta només a `localhost`. Per obrir-ho a la xarxa local:

```
sudo systemctl edit ollama
```

```
[Service]
Environment="OLLAMA_HOST=0.0.0.0:11434"
```

```
sudo systemctl daemon-reload
sudo systemctl restart ollama
```

AVÍS

No exposis Ollama a Internet sense autenticació addicional.

Resum d'ordres essencials

```
ollama run <model>          # Descarrega (si cal) i inicia conversa
ollama pull <model>         # Descarrega un model sense executar-lo
ollama list                 # Llista models en local
ollama rm <model>           # Elimina un model
ollama show <model>         # Informació del model
ollama serve                # Inicia el servidor manualment (sense
 ↪  systemd)
systemctl status ollama     # Estat del servei
```

Versions d'aquest document

- HTML - ollama.html
- PDF - ollama.pdf
- ODT - ollama.odt
- MD - ollama.md