1. Ollama介紹

Ollama 是一個開源工具，用於在本地運行大型語言模型 (LLM)。它具有以下主要特點：

1. **本地運行** - 允許在個人電腦或伺服器上部署和運行語言模型，無需雲端服務
2. **支援多種模型** - 從您的代碼中可以看到支援如 Gemma、Llama3、DeepSeek 等模型
3. **API 接口** - 提供簡單的 API 來與模型互動，如您的代碼中使用的 [generate()](vscode-file://vscode-app/c:/Users/clhuang/AppData/Local/Programs/Microsoft%20VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) 和 [chat()](vscode-file://vscode-app/c:/Users/clhuang/AppData/Local/Programs/Microsoft%20VS%20Code%20Insiders/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) 方法
4. **串流回應** - 支援流式回應模式 (streaming mode)，可逐步返回生成內容
5. Ollama in docker
6. 建立ollama容器

無GPU

docker run -d -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama:latest

有GPU

localhost

docker run -d --gpus='"device=0"' -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama:latest

公開IP

docker run -d --gpus='"device=0"' -v ollama:/root/.ollama -p 0.0.0.0:11434:11434 --name ollama ollama/ollama:latest

其他方式供參考For reference:

docker run -d --gpus='"device=0,1"' -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama:latest

docker run -d --gpus=all -v ollama:/root/.ollama -p 11435:11434 --name ollama ollama/ollama

port 被佔用怎麼辦?

docker run -d -e OLLAMA\_KEEP\_ALIVE=-1 --gpus='"device=3"' -v ollama:/root/.ollama -p 0.0.0.0:11435:11434 --name ollama ollama/ollama:latest

-e OLLAMA\_KEEP\_ALIVE=-1 模型被載入GPU後，永遠不卸載。

1. 進入容器執行ollama指令

docker exec -it ollama bash

1. ollama操作

列出模型

ollama list

下載模型

ollama pull gemma3:1b

多語言、多模態(含有vision) Multimodal (Vision)

ollama pull gemma3:4b

與模型對談Terminal方式

ollama run gemma3:4b

移除模型

ollama rm qwen2.5:7b

執行ollama指令，且進入容器:

docker exec -it ollama ollama run gemma3:1b

模型執行ollama指令，但不進入容器:

docker exec ollama ollama pull gemma3:1b

1. API usage
2. Python

|  |
| --- |
| # 設置遠程 Ollama 模型的基礎 URL  REMOTE\_OLLAMA\_URL = "http://163.18.22.32:11435"  model\_name = "gemma3:4b" # 默認模型名稱  #model\_name = "llama3:latest"  #model\_name = "deepseek-r1:7b"  # 創建客戶端實例  client = ollama.Client(host=REMOTE\_OLLAMA\_URL)  try:  # 使用 Ollama API 獲取回應  response = client.generate(model=model\_name, prompt=input\_text, stream=False)  print({'response': response['response']})  except Exception as e:  print(f"Error in Ollama API call: {str(e)}")  return JsonResponse({'error': str(e)}, status=500) |

1. 使用curl

curl -X POST http://localhost:11434/api/generate -d '{

"model": "gemma3:4b",

"prompt": "解釋什麼是深度學習？"

}'

curl -X POST http://localhost:11434/api/generate -d '{

"model": "deepseek-r1",

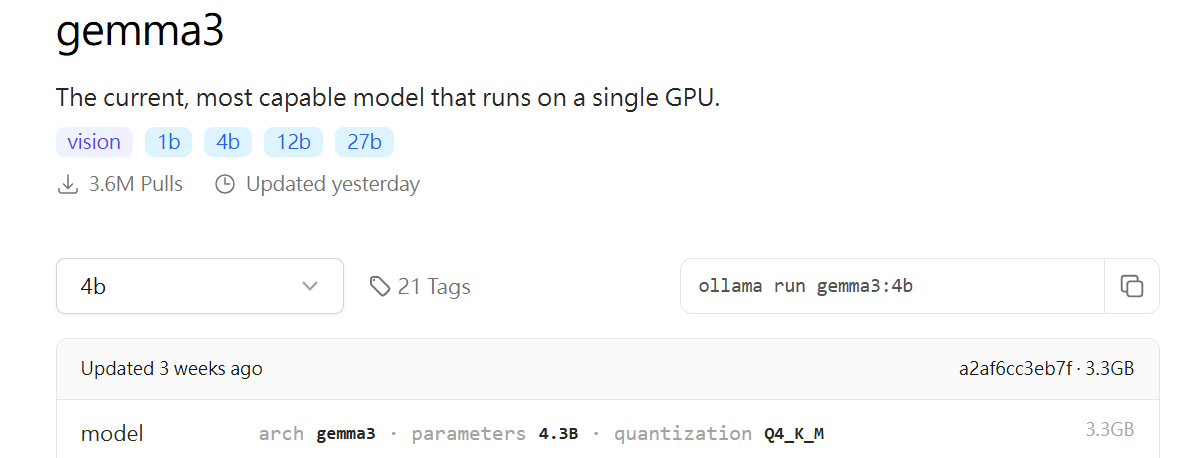
"prompt": "解釋什麼是深度學習？"

}'

1. 有哪些模型可以下載

<https://hub.docker.com/r/ollama/ollama>

1. Gemma 3



<https://ollama.com/library/gemma3:4b>

| **模型參數大小** | **檔案大小** | **執行命令** |
| --- | --- | --- |
| 1B | 約 815 MB | ollama run gemma3:1b |
| 4B | 約 3.3 GB | ollama run gemma3 |
| 12B | 約 8.1 GB | ollama run gemma3:12b |
| 27B | 約 17 GB | ollama run gemma3:27b |

1. 4B IT

在使用 Ollama 執行 Gemma 3 4B IT 模型時，預設下載的是 4-bit 量化版本，檔案大小約為 **3.3 GB**。​此模型格式為 Ollama 專用格式，經過優化以在本地設備上高效運行。

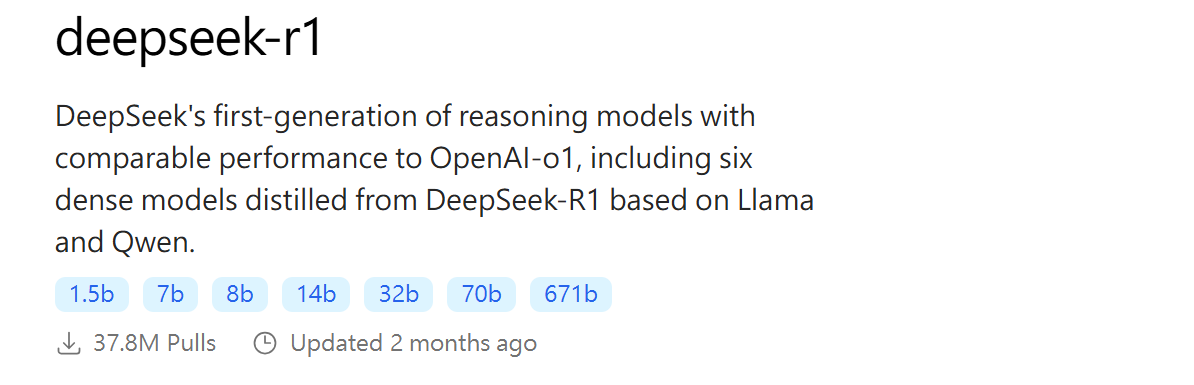
1. 模型格式與版本

* **格式**：​Ollama 專用格式，內部可能基於 GGUF 或其他壓縮格式，但已封裝供 Ollama 使用。
* **預設版本**：​4-bit 量化（Q4\_0）

1. 什麼是「量化」（Quantization）

量化是將模型權重和/或激活從較高位寬的浮點格式（如FP32、FP16）轉換為更低位寬的**整數**格式（如INT8、INT4），以減少模型大小和計算資源需求，並通常伴隨輕微的精度損失

1. Deepseek-r1



Below are the models created via fine-tuning against several dense models widely used in the research community using reasoning data generated by DeepSeek-R1. The evaluation results demonstrate that the distilled smaller dense models perform exceptionally well on benchmarks.

**DeepSeek-R1-Distill-Qwen-1.5B**

ollama run deepseek-r1:1.5b

**DeepSeek-R1-Distill-Qwen-7B**

ollama run deepseek-r1:7b

**DeepSeek-R1-Distill-Llama-8B**

ollama run deepseek-r1:8b

**DeepSeek-R1-Distill-Qwen-14B**

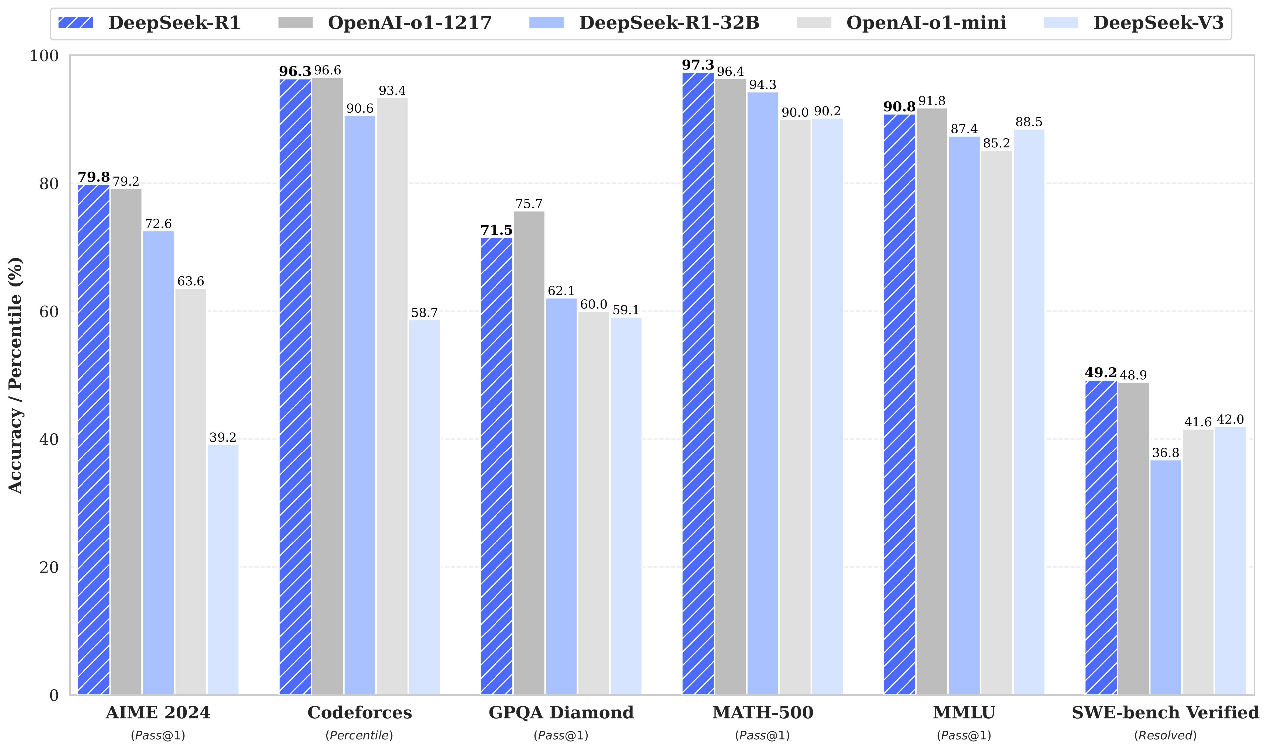
ollama run deepseek-r1:14b

**DeepSeek-R1-Distill-Qwen-32B**

ollama run deepseek-r1:32b

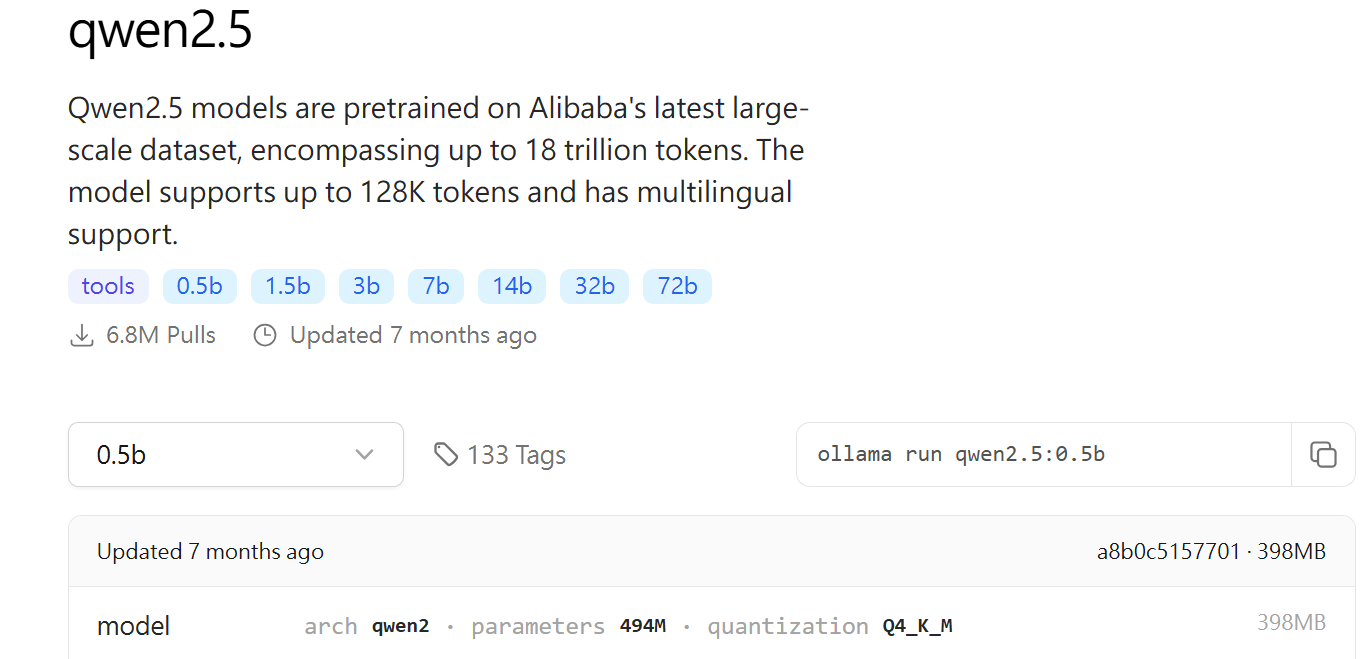
**DeepSeek-R1-Distill-Llama-70B**

ollama run deepseek-r1:70b



<https://ollama.com/library/deepseek-r1>

1. qwen2.5



<https://ollama.com/library/qwen2.5>

1. docker command

# 或使用 docker 命令

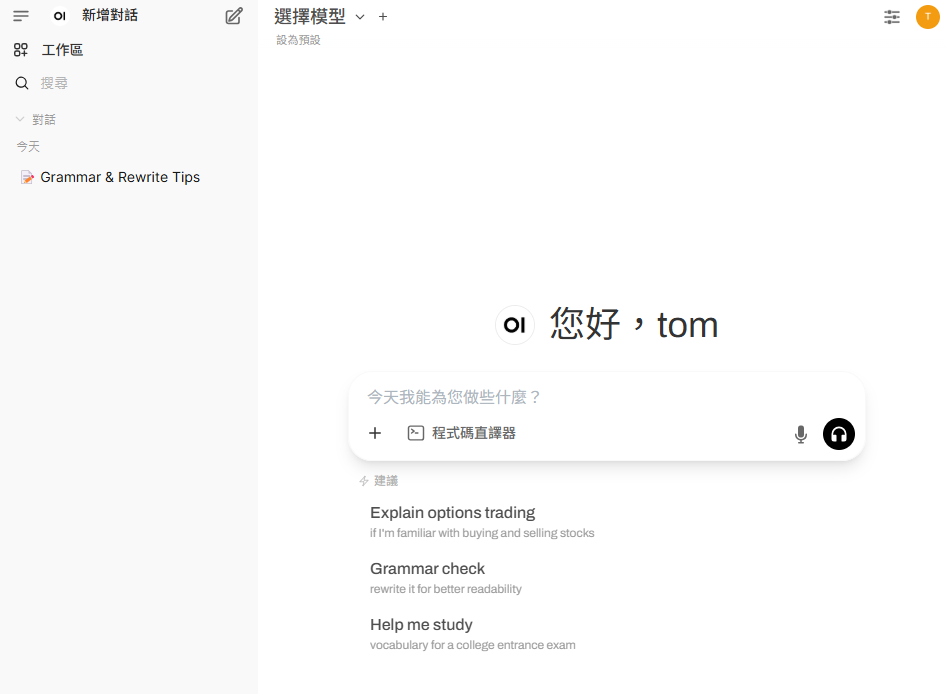
docker stop ollama

docker rm ollama

# 移除鏡像

docker rmi ollama/ollama:latest

1. 使用docker compose建立容器+使用者webui介面



建立容器，並在背景中執行

docker compose up -d

停止並移除容器

docker-compose down

docker-compose.yml

|  |
| --- |
| services:  ollama:  image: ollama/ollama:latest # 使用 Ollama 的最新版本映像檔  container\_name: ollama # 指定容器名稱為 ollama  ports:  - "0.0.0.0:11435:11434" # 對外開放本機的 11435 端口對應容器內部的 11434 端口（Ollama 預設 API 端口）  volumes:  - ollama-data:/root/.ollama # 將容器內部的模型資料夾掛載到本地持久化卷中  restart: unless-stopped # 若容器意外停止，自動重啟（除非手動停止）  tty: true # 分配虛擬終端，對某些工具及除錯方便  environment:  - OLLAMA\_HOST=0.0.0.0:11434 # 指定 Ollama 在容器內綁定的主機與端口  - OLLAMA\_NUM\_PARALLEL=2 # 同時允許 2 個請求並行處理  - OLLAMA\_MAX\_QUEUE=512 # 請求等待佇列的最大長度  - OLLAMA\_MAX\_LOADED\_MODELS=3 # 同時可在記憶體中保留的模型數量上限  - OLLAMA\_KEEP\_ALIVE=-1 # 保持模型常駐記憶體，不自動卸載  deploy:  resources:  reservations:  devices:  - driver: nvidia # 使用 NVIDIA 驅動  device\_ids: ["6"] # 指定使用第 6 與第 7 顆 GPU（從 0 開始計算）  capabilities: [gpu] # 宣告此裝置具有 GPU 能力  open-webui:  image: ghcr.io/open-webui/open-webui:main # 使用 Open WebUI 的主要映像版本  container\_name: open-webui # 容器名稱為 open-webui  volumes:  - open-webui:/app/backend/data # 持久化 WebUI 的後端資料  depends\_on:  - ollama # 此容器需在 ollama 啟動之後才啟動  ports:  - 0.0.0.0:3000:8080 # 對外開放本機的 3000 端口對應容器內部的 8080 端口（Web UI 用戶介面）  environment:  - "OLLAMA\_BASE\_URL=http://ollama:11434" # 指定 WebUI 要連接的 Ollama API 位置    extra\_hosts:  - host.docker.internal:host-gateway # 允許容器內部訪問主機（常用於 Windows/Mac）  restart: unless-stopped # 若容器意外停止，自動重啟（除非手動停止）  volumes:  ollama-data: # 定義名為 ollama-data 的卷，儲存模型資料 內定的路徑不必指定  open-webui: # 定義名為 open-webui 的卷，儲存 UI 相關資料 |