Local AI Models

Running AI models locally with Ollama and LM Studio

Why Run Models Locally?

- **Privacy & Data Control**: Your data never leaves your machine
- **Offline Capabilities**: Work without internet connection
- **Cost Control**: No per-token charges for high-volume usage
- **Reduced Latency**: No network round trips
- **Experimentation**: Try different open-source models freely
- **Customization**: Fine-tune models for your specific needs

Tradeoffs: Local vs Cloud

Local Models ✅

- Complete privacy
- No usage costs
- Offline capability
- Full control

Cloud APIs ✅

- Faster inference
- No hardware limits
- Zero setup
- Always updated

Key Considerations

- **Hardware requirements**: Local models need significant RAM/VRAM
- **Model quality**: Top cloud models often outperform local ones
- **Maintenance**: Local setup requires more technical management

Ollama: Command-Line Model Runner

What is Ollama?

- CLI tool for running LLMs locally
- Simple model management
- OpenAI-compatible API
- Cross-platform support

Key Features

- One-command model installation
- Built-in API server
- Memory-efficient quantization
- Concurrent model loading

Installation

```
# macOS/Linux
curl -fsSL https://ollama.ai/install.sh | sh

# Or download from ollama.ai
```

Basic Usage

```
# Start API server
ollama serve

# Pull a model
ollama pull gemma3:1b

# Run interactively
ollama run gemma3:1b
```

Popular Ollama Models (2025)

Latest & Greatest

- **devstral** (24B) - Best open-source coding agent model
- **qwen3** (0.6B - 235B) - Latest reasoning powerhouse
- **deepseek-r1** (1.5B - 671B) - Reasoning specialist
- **phi4** (14B) - Microsoft's state-of-the-art model
- **gemma3** (1B, 4B, 9B, 27B) - Google's newest efficient model

Usage Examples

```
# Latest coding model
ollama pull devstral

# Reasoning model
ollama pull deepseek-r1:14b

# Efficient general model
ollama pull gemma3:9b
```

Vision Models with Ollama

Multimodal Capabilities

- **Image analysis** and understanding
- **Document scanning** with OCR
- **Translation** of visual text
- **Contextual reasoning** about images
- **Multiple image** processing

Supported Vision Models

- **Llama 4** - Scout & Maverick variants
- **Gemma 3** - Multimodal support
- **Qwen 2.5 VL** - Vision-language model

Usage Examples

```
# Pull a vision model
ollama pull llama4:scout

# Interactive vision chat
ollama run llama4:scout
>>> What do you see in this image? [path to image]
```

Ollama API Usage

Generate Completion

```
curl http://localhost:11434/api/generate \
  -d '{
    "model": "gemma3:1b",
    "prompt": "Write a Python function to reverse a string",
    "stream": false
  }'
```

Chat Interface

```
curl http://localhost:11434/api/chat \
  -d '{
    "model": "gemma3:1b",
    "messages": [
      {
        "role": "user",
        "content": "Explain recursion"
      }
    ]
  }'
```

Python Integration

```python
import ollama

# Simple generation
response = ollama.generate(
    model='gemma3:1b',
    prompt='Write a hello world program'
)
print(response['response'])

# Chat conversation
response = ollama.chat(
    model='gemma3:1b',
    messages=[{
        'role': 'user',
        'content': 'How do I use decorators in Python?'
    }]
)
print(response['message']['content'])
```

LM Studio: GUI Model Manager

What is LM Studio?

- Desktop GUI for local AI models
- Cross-platform (Windows, Mac, Linux)
- Model discovery and download
- Built-in chat interface
- Local API server

Interface Highlights

- **Model Library**: Browse and download models
- **Chat Interface**: Test models interactively
- **Server Tab**: Configure local API
- **Settings**: Hardware optimization

Gollama: Bridging Ollama & LM Studio

CW JV

What is Gollama?

- TUI (Text User Interface) for Ollama
- Enhanced model management
- Links Ollama models to LM Studio
- macOS/Linux tool
- Written in Go

Key Features

- Interactive model browser
- vRAM usage estimation
- Model copying between hosts
- Theme customization
- Advanced filtering

Gollama

Installation

```
# Go install (recommended)
go install github.com/sammcj/gollama@HEAD

# Or curl
curl -sSL https://raw.githubusercontent.com/sammcj/gollama/main/install.sh | bash
```

Usage

```
# Launch TUI
gollama

# Command line mode
gollama list
gollama run llama3.2
gollama link  # Link to LM Studio
```

Integration Workflows

Development Workflow

- **Explore** models in LM Studio
- **Download** via Ollama for scripting
- **Manage** with Gollama TUI
- **Integrate** via APIs

Hybrid Approach

- **Local** for development/testing
- **Cloud** for production/scale
- **Ollama** for automation
- **LM Studio** for experimentation

Tool Comparison

Feature	Ollama	LM Studio	Gollama
CLI	✅	❌	✅
GUI	❌	✅	TUI
API	✅	✅	❌
Model Discovery	Basic	✅	✅
vRAM Estimation	❌	✅	✅
Cross-platform	✅	✅	Mac/Linux

Hardware Considerations

Memory Requirements

- **7B models**: 8-16GB RAM
- **13B models**: 16-32GB RAM
- **34B+ models**: 32GB+ RAM
- **Quantization helps**: Q4 ≈ 50% size reduction

GPU Acceleration

- **NVIDIA**: CUDA support
- **Apple**: Metal acceleration
- **AMD**: ROCm support (Linux)
- **CPU-only**: Slower but works

Performance Tips

- **Quantization**: Use Q4 or Q8 for speed
- **Context length**: Shorter = faster
- **Batch size**: Optimize for your hardware
- **Multiple models**: Load on demand

Monitoring

```
# Check model size
ollama show llama3.2

# Monitor resources
htop
nvidia-smi  # For NVIDIA GPUs
```

Getting Started Hands-On

Step 1: Install Ollama

```
# macOS/Linux
curl -fsSL https://ollama.ai/install.sh | sh

# Test installation
ollama --version
```

Step 2: Get Your First Model

```
# Start with a smaller model
ollama pull gemma3:1b
```

Step 3: Test It

```
ollama run gemma3:1b
# Chat interactively, Ctrl+D to exit
```

Step 4: Install LM Studio

- Download from lmstudio.ai
- Install and launch
- Browse model library
- Download a model to try

Step 5: Try Gollama (Optional)

```
go install github.com/sammcj/gollama@HEAD
gollama
# Explore the TUI interface
# L to link all ollama models to lmstudio
```

Best Practices

- **Start small**: Begin with 3B-7B models to test your setup
- **Monitor resources**: Watch RAM/VRAM usage carefully
- **Hybrid approach**: Local for dev, cloud for production
- **Security/Privacy**: Local models eliminate data transmission risks

When to Use What

- **Ollama**: Automation, CI/CD, scripting
- **LM Studio**: Experimentation, non-technical users
- **Cloud APIs**: Latest models, production scale
- **Local models**: Privacy, offline work, cost control

Resources

- Ollama Documentation
- LM Studio
- Gollama GitHub

Have Fun!