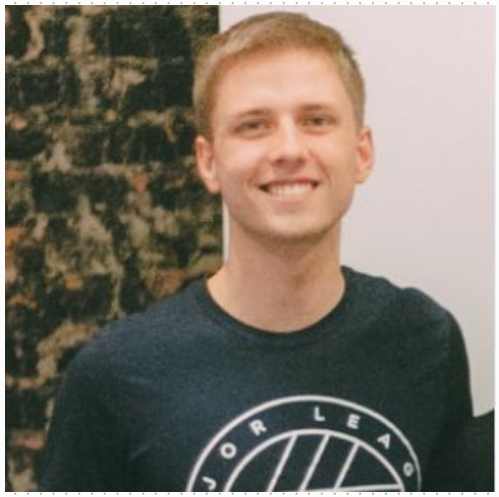
AI Agents with Low-Cost LLMs

GHW May 2025



@MLHacks

@MLHacks

Hey! I'm **Stephen**.

- Coach at Major League Hacking (~5 years).
- Super excited for GHW Open Source!

Overview

- We will learn to build agentic AI applications using CrewAI and Ollama
- Focus is on cost-effective implementation with local LLMs.
- The tutorial is split into two 2-hour sessions
- Some theoretical knowledge…
- Some hands-on experience…

Let’s get started!

Prerequisites

- Python 3.9+
- Git
- Basic understanding of Python programming
- Basic understanding of AI/ML concepts

Quick Start

Clone this repository:

```bash
git clone https://github.com/croppers/crewai
cd crewai
```

Create and activate a virtual environment

```bash
python -m venv venv
source venv/bin/activate  # On Windows:
venv\Scripts\activate
```

Install dependencies

```bash
pip install -r requirements.txt
```

Install Ollama

- Visit ollama.com
- Download and install for your operating system
- Pull a base model:

```bash
ollama pull gemma:2b
```

Part 1: Foundations & Basic Implementation

Part 1: Foundations & Basic Implementation

- Environment Setup
- CrewAI Fundamentals
- Ollama Deep Dive
- Building Your First Agent
- Multi-Agent Basics

Welcome & Overview

Introduction to CrewAI and Ollama

CrewAI: A framework for building agentic AI applications

- Agent-based architecture
- Collaborative AI systems
- Tool integration capabilities
- Process management

Introduction to CrewAI and Ollama

Ollama: Local LLM deployment

- Open-source model hosting
- Cost-effective inference
- Model management
- Performance optimization

Why Local LLMs Matter

- Cost reduction
- Data privacy
- Latency improvement
- Customization potential
- Offline capabilities

Cost Comparison Overview

- Cloud LLM costs (GPT-4, Claude, etc.)
- Local LLM costs (Ollama)
- Infrastructure requirements
- Total cost of ownership

Environment Setup

Installing Ollama

Download and Installation:

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

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

# Windows
# Download from https://ollama.com/download
```

https://ollama.com/download

Basic Model Testing

```bash
# Pull a base model
ollama pull gemma:2b

# Test the model
ollama run gemma:2b "Hello, how are you?"
```

What is gemma:2b?

Model Management

```bash
# List available models
ollama list

# Remove a model
ollama rm gemma:2b

# Pull specific model version
ollama pull gemma:2b
```

Setting up CrewAI

Python Environment Setup

```bash
# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

CrewAI Fundamentals

Agent Architecture

hello_world.py

Crew Concepts

first_crew.py

Ollama Deep Dive

Available Models

- Gemma (2B, 7B)
- Llama2 (7B, 13B, 70B)
- Mistral
- Qwen
- and more…

Model Selection Criteria

- Task requirements
- Hardware constraints
- Performance needs
- Cost considerations

Performance Considerations

- Memory usage
- Inference speed
- Quality of outputs
- Resource utilization

Cost Implications

- Hardware requirements
- Electricity costs
- Maintenance overhead
- Scaling considerations

Building Your First Agent

Single Agent Implementation

custom_agent.py

Multi-Agent Basics

Introduction to Crews

- Crew architecture
- Agent communication
- Task delegation
- Collaboration patterns

Building a Simple Crew

```
from crewai import Agent, Crew, Task
from langchain_community.llms import Ollama
# Initialize agents
researcher = Agent(
    role='Researcher',
    goal='Research topics thoroughly',
    backstory='Expert researcher',
    llm=Ollama(model="ollama/gemma:2b")
)
analyst = Agent(
    role='Analyst',
    goal='Analyze research findings',
    backstory='Data analyst with strong analytical skills',
    llm=Ollama(model="ollama/gemma:2b")
)
writer = Agent(
    role='Writer',
    goal='Create engaging content',
    backstory='Experienced content writer',
    llm=Ollama(model="ollama/gemma:2b")
)
```

Building a Simple Crew

```
# Create tasks
research_task = Task(
    description="Research AI in healthcare",
    agent=researcher
)
analysis_task = Task(
    description="Analyze the research findings",
    agent=analyst
)
writing_task = Task(
    description="Write a comprehensive report",
    agent=writer
)
# Create and run crew
crew = Crew(
    agents=[researcher, analyst, writer],
    tasks=[research_task, analysis_task, writing_task],
    verbose=True
)
result = crew.kickoff()
```

Part 2: Advanced Implementation & Real-World Applications

Basic Configuration

Create **.env** file:

```env
SERPER_API_KEY=YOUR_API_KEY
```

Part 2: Advanced Implementation & Real-World Applications

- Advanced Agent Development
- Building Complex Crews
- Real-World Application Development

Part 1: Advanced Agent Patterns

agent_patterns_hierarchical.py

Part 2: Using a real tool: Google Serper Search

`research_crew_serper.py`

Part 3: Building A Web App for our “Researcher”

```
agent_web_app
```

Challenge: Build our own AI Agent!

We will use codeshare.io to share

Other ways to go…

- More use cases
- More models
- Fun!