Crew AI is an open-source framework designed to help developers orchestrate **multiple AI agents** to work together like a team (or "crew") to solve complex tasks.

Instead of using a single AI agent to perform a job, **Crew AI** lets you create specialized agents, assign them roles and responsibilities, and have them **collaborate** on tasks—much like humans in a company would.

🔗 What is an "Agent" in Crew AI?

An **agent** in Crew AI is an AI persona with:

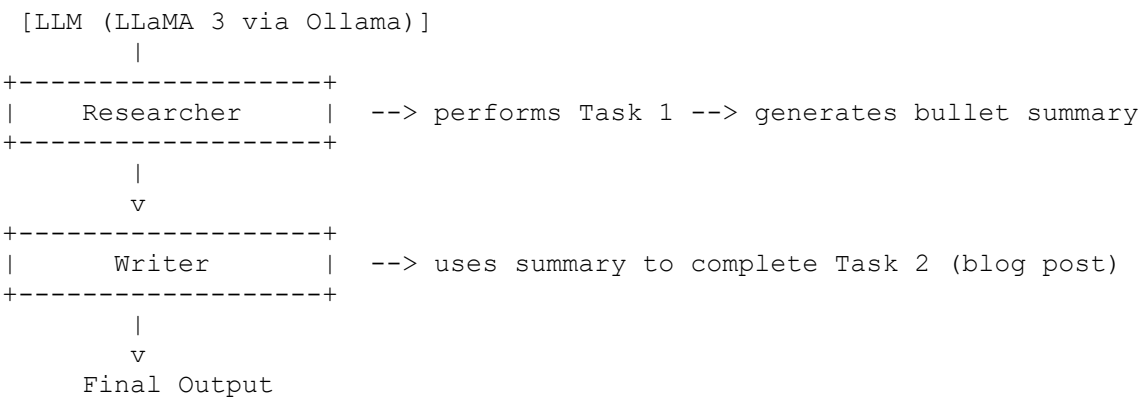
- A **role** (e.g., Writer, Researcher, Developer)
- **Goals** (what it's trying to accomplish)
- **Tools** (optional – for accessing the web, running code, etc.)
- An **LLM** (language model like GPT-4 or open-source models)

✅ Step-by-Step Guide: Creating a Simple Crew AI App

Let's walk through building a simple Crew AI app where two agents collaborate:

- A **Researcher** agent who gathers information.
- A **Writer** agent who writes a summary based on that research.

🧠 Diagram: How It All Flows



---

use **CrewAI with Ollama and LLaMA 3**, you'll need to:

1. **Install Ollama** and pull the `llama3` model.
2. Replace the OpenAI LLM with a **LangChain wrapper** for Ollama.

✅ Step-by-Step Conversion: Use LLaMA 3 (via Ollama) with CrewAI

✅ What Happens Under the Hood:

1. The `Researcher` agent is assigned its task and uses the LLM to complete it.
2. The `Writer` agent gets the output from the `Researcher` and uses that as **context** for writing the blog.
3. The `Crew` manages the flow: first `research_task`, then `writing_task`.

🛠️ Prerequisites

1. **Install Ollama** (if not already):
   https://ollama.com/download
2. **Pull the LLaMA 3 model** (7B or 8B):

```
ollama pull llama3
```

3. **Install required Python packages**:

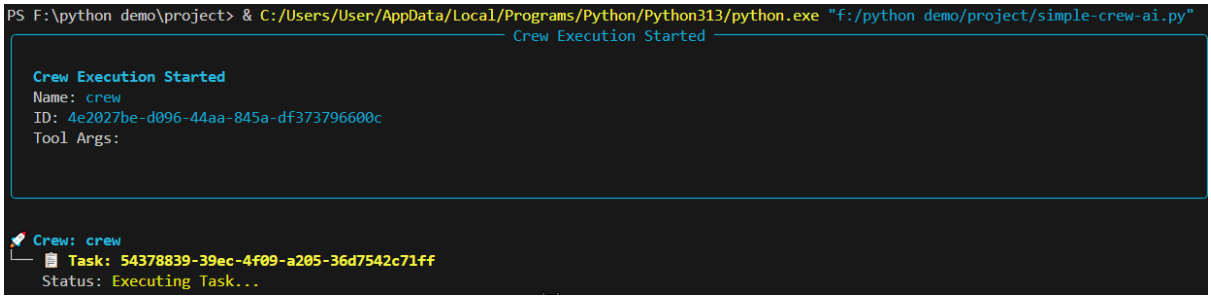
```
pip install crewai langchain
```

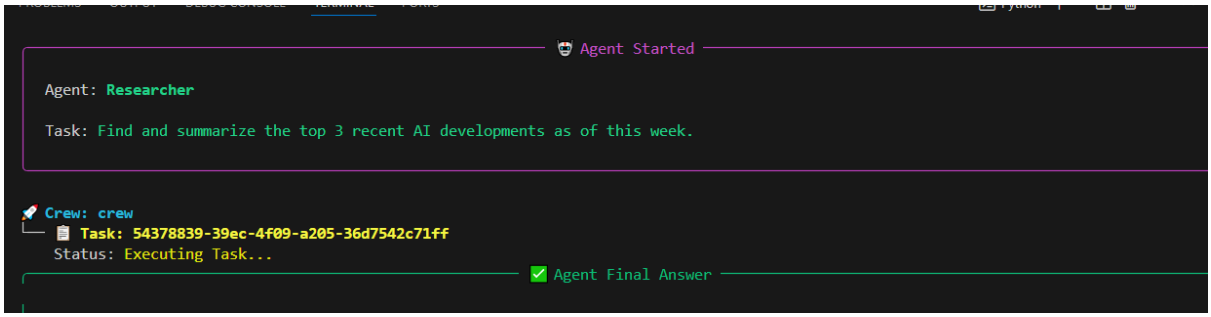
🧠 Full Script Ai Agents Using Ollama LLaMA 3

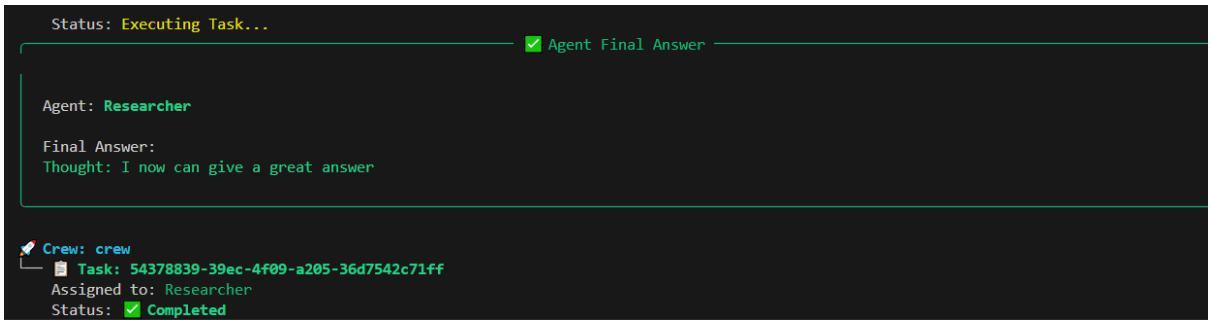
```
from crewai import LLM, Agent, Task, Crew

# Instantiate the local LLaMA-3 model via Ollama
llm_local = LLM(
    model="ollama/llama3",  # Make sure this matches your Ollama model name
    base_url="http://localhost:11434",
    api_key=""  # Not needed for Ollama but required by interface
)

# Define Agents
researcher = Agent(
    role="Researcher",
    goal="Collect recent news about Artificial Intelligence developments",
```

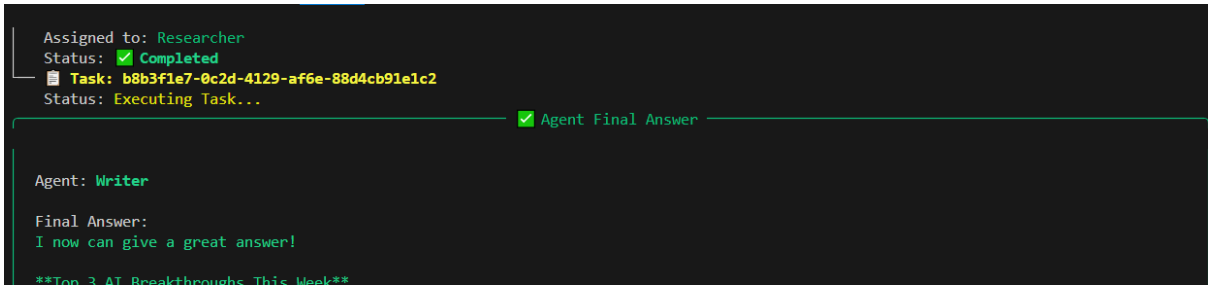
```
    backstory="An expert tech journalist",
    verbose=True,
    allow_delegation=False,
    llm=llm_local
)

writer = Agent(
    role="Writer",
    goal="Write a concise and informative blog post based on research",
    backstory="A skilled content writer",
    verbose=True,
    allow_delegation=False,
    llm=llm_local
)

# Define Tasks
research_task = Task(
    description="Find and summarize the top 3 recent AI developments as of
this week.",
    expected_output="A list of 3 bullet points with summaries and source
URLs.",
    agent=researcher
)

writing_task = Task(
    description="Using the research summary, write a blog post titled 'Top 3
AI Breakthroughs This Week'.",
    expected_output="A 300-500 word well-structured blog post.",
    agent=writer,
    context=[research_task]
)

# Create the Crew and run
crew = Crew(
    agents=[researcher, writer],
    tasks=[research_task, writing_task],
    verbose=True
)

result = crew.kickoff()
print("\n📝 Final Blog Post:\n", result)
```

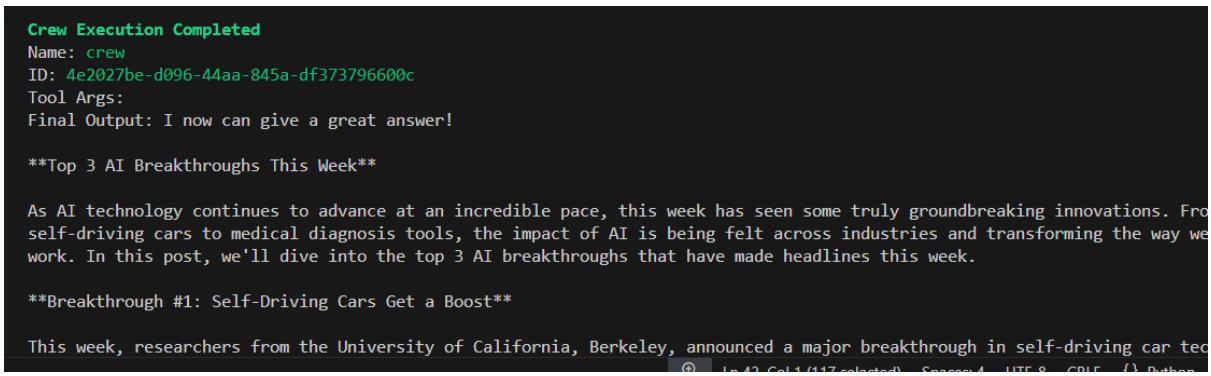
Output:-

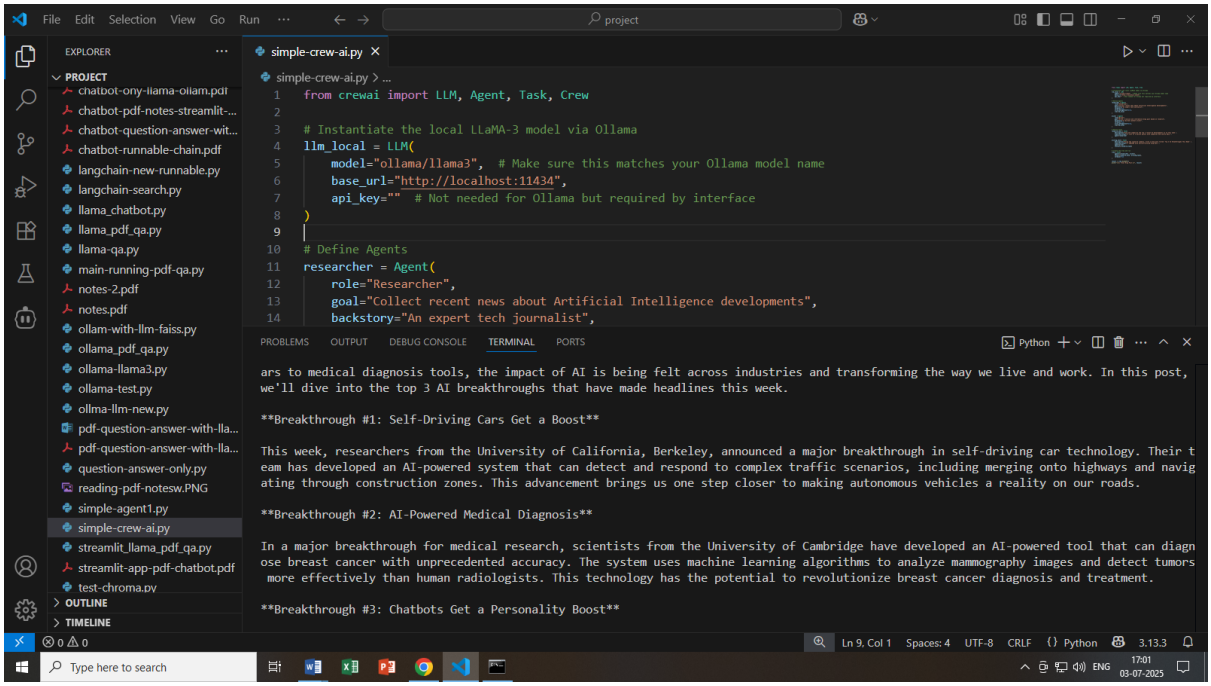

PS F:\python demo\project> & C:/Users/User/AppData/Local/Programs/Python/Python313/python.exe "f:/python demo/project/simple-crew-ai.py"
Crew Execution Started
Crew Execution Started
Name: crew
ID: 4e2027be-d096-44aa-845a-df373796600c
Tool Args:
Crew: crew
Task: 54378839-39ec-4f09-a205-36d7542c71ff
Status: Executing Task...


Agent Started
Agent: Researcher
Task: Find and summarize the top 3 recent AI developments as of this week.
Crew: crew
Task: 54378839-39ec-4f09-a205-36d7542c71ff
Status: Executing Task...
Agent Final Answer


Status: Executing Task...
Agent Final Answer
Agent: Researcher
Final Answer:
Thought: I now can give a great answer
Crew: crew
Task: 54378839-39ec-4f09-a205-36d7542c71ff
Assigned to: Researcher
Status: Completed


Assigned to: Researcher
Status: Completed
Task: b8b3f1e7-0c2d-4129-af6e-88d4cb91e1c2
Status: Executing Task...
Agent Final Answer
Agent: Writer
Final Answer:
I now can give a great answer!


Crew Execution Completed
Name: crew
ID: 4e2027be-d096-44aa-845a-df373796600c
Tool Args:
Final Output: I now can give a great answer!
Top 3 AI Breakthroughs This Week
As AI technology continues to advance at an incredible pace, this week has seen some truly groundbreaking innovations. Fro
self-driving cars to medical diagnosis tools, the impact of AI is being felt across industries and transforming the way we
work. In this post, we'll dive into the top 3 AI breakthroughs that have made headlines this week.
Breakthrough #1: Self-Driving Cars Get a Boost
This week, researchers from the University of California, Berkeley, announced a major breakthrough in self-driving car tec



⚠ Make Sure:

- **Ollama is running**:

  `ollama run llama3`

- `model="ollama/llama3"` — or match the exact name listed from:

  `ollama list`

- The base URL `http://localhost:11434` is accessible (Ollama's default).

Here's a **step-by-step explanation** of this above code , broken down clearly for beginners and intermediate users who want to understand how **CrewAI + Ollama (LLaMA 3)** works:-

✅ Step-by-Step Breakdown

✅ 1. Import Core Classes

```
from crewai import LLM, Agent, Task, Crew
```

These are the core building blocks of **CrewAI**:

- `LLM`: Lets you connect to a local or cloud-based language model (like LLaMA 3, GPT-4, Claude, etc.).
- `Agent`: An AI "persona" with a role, goal, and access to an LLM.
- `Task`: A specific piece of work assigned to an Agent.
- `Crew`: A team (group of Agents + Tasks) that collaborates to solve a problem.

✅ 2. Instantiate the LLM (LLaMA 3 via Ollama)

```
llm_local = LLM(
    model="ollama/llama3",        # Must match your local Ollama model name
    base_url="http://localhost:11434",   # Default API endpoint for Ollama
    api_key=""                    # Blank because Ollama doesn't need a key
)
```

- This connects your **local LLaMA 3 model** running via **Ollama** to CrewAI.
- Ollama must be running:
  ➤ `ollama run llama3`

✅ 3. Define the Agents

```
researcher = Agent(
    role="Researcher",
    goal="Collect recent news about Artificial Intelligence developments",
    backstory="An expert tech journalist",
    verbose=True,
    allow_delegation=False,
    llm=llm_local
```

```
)

writer = Agent(
    role="Writer",
    goal="Write a concise and informative blog post based on research",
    backstory="A skilled content writer",
    verbose=True,
    allow_delegation=False,
    llm=llm_local
)
```

- **Agents** are personas with roles and specific goals.
- `role` and `goal` define their purpose.
- `backstory` adds personality and tone.
- `verbose=True` prints more logs for transparency.
- `llm=llm_local` tells the agent which LLM to use (in this case, local LLaMA 3).

✅ 4. Define the Tasks

```
research_task = Task(
    description="Find and summarize the top 3 recent AI developments as of
this week.",
    expected_output="A list of 3 bullet points with summaries and source
URLs.",
    agent=researcher
)
```

- Assigns the **Researcher** agent a task to find 3 AI news items.
- The task must return a **summary + URLs** (in bullet point form).

```
writing_task = Task(
    description="Using the research summary, write a blog post titled 'Top
3 AI Breakthroughs This Week'.",
    expected_output="A 300-500 word well-structured blog post.",
    agent=writer,
    context=[research_task]
)
```

- Assigns the **Writer** agent a task to write the blog post.
- `context=[research_task]` means the Writer **waits for and uses the output** of the Researcher task.

✅ 5. Create the Crew and Execute

```
crew = Crew(
    agents=[researcher, writer],
    tasks=[research_task, writing_task],
    verbose=True
)
```

- The **Crew** coordinates the execution order and communication between agents.
- `agents` defines the team members.
- `tasks` defines the workflow.
- `verbose=True` logs the progress.

✅ 6. Run the Crew

```
result = crew.kickoff()
```

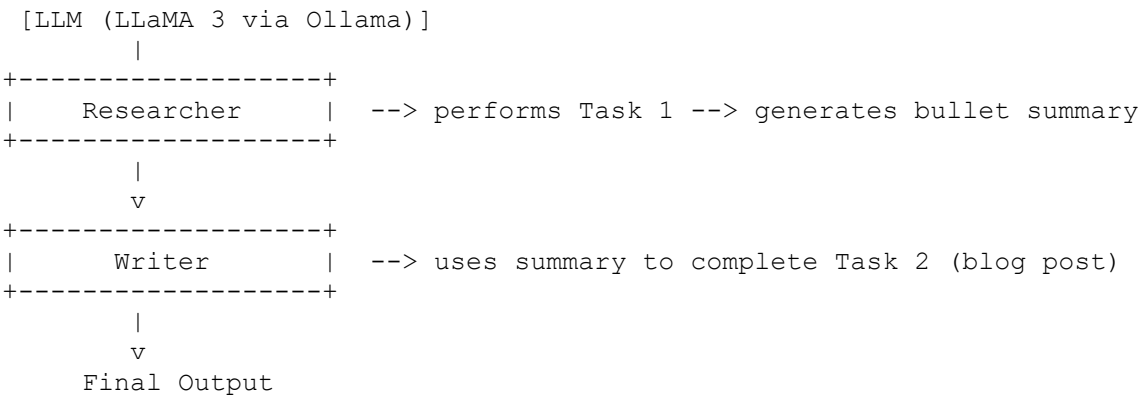
- This **starts the full workflow**:
    1. The **Researcher** does their task and outputs the 3 AI updates.
    2. The **Writer** receives that output and writes a blog post.

✅ 7. Print the Final Result

```
print("\n📝 Final Blog Post:\n", result)
```

- Prints the result of the **Writer's** task (the blog post).

🧠 Diagram: How It All Flows



✅ Summary of Core Concepts

Concept	Description
LLM	Connects to local LLaMA 3 model using Ollama
Agent	A persona that uses an LLM to perform a goal
Task	A single job or prompt given to an agent
Crew	Orchestrates the tasks and agents to solve a bigger objective

Language model (like OpenAI’s GPT)

□ Step 1: Install Crew AI

```
pip install crewai
```

You’ll also need `langchain` and a language model (like OpenAI’s GPT or Llama3):

```
pip install openai langchain
```

Set your OpenAI API key:

```
export OPENAI_API_KEY='your-api-key-here'
```

👦💻 Step 2: Create the Agents

```
from crewai import Agent
from langchain.chat_models import ChatOpenAI

# Define the LLM
llm = ChatOpenAI(model_name='gpt-4', temperature=0.5)
```

```
# Agent 1: Researcher
researcher = Agent(
    role='Researcher',
    goal='Collect recent news about Artificial Intelligence developments',
    backstory="An expert tech journalist, great at finding accurate and up-
to-date news.",
    verbose=True,
    allow_delegation=False,
    llm=llm
)

# Agent 2: Writer
writer = Agent(
    role='Writer',
    goal='Write a concise and informative blog post based on research',
    backstory="A skilled content writer who crafts compelling blog posts
from technical data.",
    verbose=True,
    allow_delegation=False,
    llm=llm
)
```

📋 Step 3: Define a Task for Each Agent

```
from crewai import Task

# Task for Researcher
research_task = Task(
    description="Find and summarize the top 3 recent AI developments as of
this week.",
    expected_output="A list of 3 bullet points with summaries and source
URLs.",
    agent=researcher
)

# Task for Writer
writing_task = Task(
    description="Using the research summary, write a blog post titled 'Top
3 AI Breakthroughs This Week'.",
    expected_output="A well-structured blog post around 300-500 words.",
    agent=writer,
    context=[research_task]  # Writer uses output from Researcher
)
```

👥 Step 4: Create the Crew

```
from crewai import Crew

crew = Crew(
    agents=[researcher, writer],
```

```
    tasks=[research_task, writing_task],
    verbose=True
)
```

▶️ Step 5: Run the Crew

```
result = crew.kickoff()
print("📄 Final Blog Post:\n")
print(result)
```

✅ What Happens Under the Hood:

1. The `Researcher` agent is assigned its task and uses the LLM to complete it.
2. The `Writer` agent gets the output from the `Researcher` and uses that as **context** for writing the blog.
3. The `Crew` manages the flow: first `research_task`, then `writing_task`.

Full working code together:-

crew_ai_blog_creator.py

```
import os
from crewai import Agent, Task, Crew
from langchain.chat_models import ChatOpenAI

# Set your API key
os.environ["OPENAI_API_KEY"] = "your-api-key"  # Replace with your key

# Step 1: Define the LLM
llm = ChatOpenAI(model_name='gpt-4', temperature=0.5)

# Step 2: Define Agents
researcher = Agent(
    role='Researcher',
    goal='Collect recent news about Artificial Intelligence developments',
    backstory="An expert tech journalist, great at finding accurate and up-to-
date news.",
    verbose=True,
    allow_delegation=False,
    llm=llm
)

writer = Agent(
```

```
    role='Writer',
    goal='Write a concise and informative blog post based on research',
    backstory="A skilled content writer who crafts compelling blog posts from technical data.",
    verbose=True,
    allow_delegation=False,
    llm=llm
)

# Step 3: Define Tasks
research_task = Task(
    description="Find and summarize the top 3 recent AI developments as of this week.",
    expected_output="A list of 3 bullet points with summaries and source URLs.",
    agent=researcher
)

writing_task = Task(
    description="Using the research summary, write a blog post titled 'Top 3 AI Breakthroughs This Week'.",
    expected_output="A well-structured blog post around 300-500 words.",
    agent=writer,
    context=[research_task]  # Gets input from the research task
)

# Step 4: Create the Crew
crew = Crew(
    agents=[researcher, writer],
    tasks=[research_task, writing_task],
    verbose=True
)

# Step 5: Run the Crew
result = crew.kickoff()

# Print Final Output
print("\n📝 Final Blog Post:\n")
print(result)
```