**Agentic AI**

**1. What is Agentic AI?**

**Agentic AI** means AI systems that:

* **Think** (plan their actions)
* **Act** (interact with the world, tools, or APIs)
* **Learn/Adapt** (improve based on feedback)

Instead of being passive (like a simple chatbot), they work as **autonomous agents**:

* They take a **goal** (“Find me the best flight to Paris”)
* Break it into **steps**
* Execute the steps (search sites, compare prices, book the ticket)
* Report or take action without constant user input.

**2. Key Components of Agentic AI**

1. **LLM Core** – A large language model (like GPT-4, Claude, LLaMA) to handle reasoning & language.
2. **Memory** – Stores context, history, and results to make multi-step reasoning possible.
3. **Planning Engine** – Decides what steps to take next.
4. **Tool Use / API Access** – The agent can:
   * Call APIs (weather, flights, database)
   * Run code
   * Search the web
   * Control apps
5. **Feedback Loop** – Evaluates results and adjusts the plan.

**3. Examples of Agentic AI**

* **Auto-GPT** – Early open-source autonomous AI that can plan & execute tasks.
* **LangChain Agents** – Let you connect LLMs to tools, APIs, and memory.
* **OpenAI GPT-4o + Actions** – Can call external APIs.
* **Microsoft AutoGen** – Framework for multi-agent collaboration.

**4. How It Works (Example Flow)**

Let’s say the goal is:  
*"Summarize the latest AI research papers and email me the summary."*

1. **Understand the goal** (LLM interprets request)
2. **Plan** (Step 1: search arXiv → Step 2: download papers → Step 3: summarize → Step 4: send email)
3. **Execute each step** (using Python code, APIs, etc.)
4. **Check the result** (make sure summaries are correct)
5. **Finish & report** (send email to user)

**5. How to Start Learning Agentic AI**

**Beginner Path:**

1. Learn **Python basics** (needed for automation).
2. Learn **LLM basics** (prompt engineering, context handling).
3. Explore **LangChain** or **LlamaIndex** (agent frameworks).
4. Try **simple agents**:
   * An AI that reads your emails and drafts replies.
   * An AI that fetches stock prices daily and updates a Google Sheet.

**6. Tools & Frameworks for Agentic AI**

* **LangChain** – Popular for building LLM-powered agents.
* **LlamaIndex** – For connecting AI to private data.
* **Microsoft AutoGen** – Multi-agent workflows.
* **OpenAI Functions / Actions** – Let GPT call your APIs.
* **CrewAI** – For orchestrating multiple AI agents.

Agentic AI is an autonomous artificial intelligence system capable of making decisions and performing complex tasks with minimal human intervention. It consists of AI agents that perceive, reason, act, learn, and collaborate to achieve specific goals independently, adapting in real-time to new information and obstacles. Agentic AI leverages advanced techniques such as reinforcement learning, natural language processing, machine learning, and multi-agent collaboration to drive autonomous and intelligent behaviors in various applications like business automation, software development, and cybersecurity.

What is agentic AI?

Agentic AI is an [artificial intelligence](https://www.ibm.com/think/artificial-intelligence) system that can accomplish a specific goal with limited supervision. It consists of AI agents—machine learning models that mimic human decision-making to solve problems in real time. In a multiagent system, each agent performs a specific subtask required to reach the goal and their efforts are coordinated through [AI orchestration](https://www.ibm.com/think/topics/ai-orchestration).

Unlike traditional [AI models](https://www.ibm.com/think/topics/ai-model), which operate within predefined constraints and require human intervention, agentic AI exhibits autonomy, goal-driven behavior and adaptability. The term “agentic” refers to these models’ agency, or, their capacity to act independently and purposefully.

[Agentic AI](https://www.ibm.com/think/insights/agentic-ai) builds on [generative AI](https://www.ibm.com/think/topics/generative-ai) (gen AI) techniques by using [large language models](https://www.ibm.com/think/topics/large-language-models) (LLMs) to function in dynamic environments. While generative models focus on creating content based on learned patterns, agentic AI extends this capability by applying generative outputs toward specific goals. A [generative AI](https://www.ibm.com/think/topics/generative-ai-for-developers) model like OpenAI’s [ChatGPT](https://www.ibm.com/think/topics/gpt" \t "_self) might produce text, images or code, but an agentic AI system can use that generated content to complete complex tasks autonomously by calling external tools. Agents can, for example, not only tell you the best time to climb Mt. Everest given your work schedule, it can also book you a flight and a hotel.

## What are the advantages of agentic AI?

Agentic systems have many advantages over their generative predecessors, which are limited by the information contained in the datasets upon which models are trained.

Autonomous

The most important advancement of agentic systems is that they allow for autonomy to perform tasks without constant human oversight. Agentic systems can maintain long-term goals, manage multistep problem-solving tasks and track progress over time.

Proactive

Agentic systems provide the flexibility of LLMs, which can generate responses or actions based on nuanced, context-dependent understanding, with the structured, deterministic and reliable features of traditional programming. This approach allows agents to “think” and “do” in a more human-like fashion.

LLMs by themselves can’t directly interact with external tools or databases or set up systems to monitor and collect data in real time, but agents can. Agents can search the web, call application programming interfaces (APIs) and query databases, then use this information to make decisions and take actions.

Specialized

Agents can specialize in specific tasks. Some agents are simple, performing a single repetitive task reliably. Others can use perception and draw on memory to solve more complex problems. An [agentic architecture](https://www.ibm.com/think/topics/agentic-architecture" \t "_self) might consist of a “conductor” model powered by an LLM that oversees tasks and decisions and supervises other, simpler agents. Such architectures are ideal for sequential workflows but are vulnerable to bottlenecks. Other architectures are more horizontal, with agents working in harmony as equals in a decentralized fashion, but this architecture can be slower than a vertical hierarchy. Different AI applications demand different architectures.

Adaptable

Agents can learn from their experiences, take in feedback and adjust their behavior. With the right guardrails, agentic systems can improve continuously. Multiagent systems possess the scalability to eventually handle broadly scoped initiatives.

Intuitive

Because agentic systems are powered by LLMs, users can engage with them with natural language prompts. This means that entire software interfaces—think of the many tabs, dropdowns, charts, sliders, pop-ups and other UI elements involved in the SaaS platform of one’s choice—can be replaced by simple language or voice commands. Theoretically, any software user experience can now be reduced to “talking” with an agent, who can fetch the information one needs and take action based on that information. This productivity benefit can barely be overstated, when one considers the time it takes for workers to learn and master new interfaces and tools.

## How agentic AI works

Agentic AI tools can take many forms and different [frameworks](https://www.ibm.com/think/insights/top-ai-agent-frameworks) are better suited to different problems, but here are the general steps that agentic systems take to perform their operations.

Perception

Agentic AI begins by collecting data from its environment through sensors, APIs, databases or user interactions. This step ensures that the system has up-to-date information to analyze and act upon.

Reasoning

Once the data is collected, the AI processes it to extract meaningful insights. Using [natural language processing](https://www.ibm.com/think/topics/natural-language-processing) (NLP), computer vision or other AI capabilities, it interprets user queries, detects patterns and understands the broader context. This ability helps the AI determine what actions to take based on the situation.

Goal setting

The AI sets objectives based on predefined goals or user inputs. It then develops a strategy to achieve these goals, often by using [decision trees](https://www.ibm.com/think/topics/decision-trees), [reinforcement learning](https://www.ibm.com/think/topics/reinforcement-learning) or other planning algorithms.

Decision-making

AI evaluates multiple possible actions and chooses the optimal one based on factors such as efficiency, accuracy and predicted outcomes. It might use probabilistic models, utility functions or [machine learning](https://www.ibm.com/think/topics/machine-learning-algorithms)-based reasoning to determine the best course of action.

Execution

After selecting an action, the AI executes it, either by interacting with external systems (APIs, data, robots) or providing responses to users.

Learning and adaptation

After executing an action, the AI evaluates the outcome, gathering feedback to improve future decisions. Through reinforcement learning or [self-supervised learning](https://www.ibm.com/think/topics/self-supervised-learning), the AI refines its strategies over time, making it more effective in handling similar tasks in the future.

Orchestration

AI orchestration is the coordination and management of systems and agents. Orchestration platforms [automate](https://www.ibm.com/think/topics/automation) [AI workflows](https://www.ibm.com/think/topics/ai-workflow), track progress toward task completion, manage resource usage, monitor data flow and memory and handle failure events. With the right architecture, dozens, hundreds or even thousands of agents could theoretically work together in harmonious productivity.

## Examples of agentic AI

Agentic AI solutions can be deployed across virtually any AI use case in any real-world ecosystem. Agents can integrate within complex workflows to perform business processes autonomously.

An AI-powered trading bot can analyze live stock prices and economic indicators to perform predictive analytics and execute trades.

In autonomous vehicles, real-time data sources such as GPS and sensor data can improve navigation and safety.

In healthcare, agents can monitor patient data, adjust treatment recommendations based on new test results and provide real-time feedback to clinicians through [chatbots](https://www.ibm.com/think/topics/chatbots" \t "_self).

In cybersecurity, agents can continuously monitor network traffic, system logs and user behavior for anomalies that might indicate vulnerabilities to malware, phishing attacks or unauthorized access attempts.

AI can streamline supply chain management through [process automation](https://www.ibm.com/think/topics/business-process-automation) and optimization, autonomously placing orders with suppliers or adjusting production schedules to maintain optimal inventory levels.

Challenges for agentic AI systems

Agentic AI systems have massive potential for the enterprise. Their autonomy is their primary benefit, but this autonomous nature can bring serious consequences if agentic systems go “off the rails.” The usual [AI risks](https://www.ibm.com/think/topics/ai-safety) apply, but can be magnified in agentic systems.

Many agentic AI systems use reinforcement learning, which involves maximizing a reward function. If the reward system is poorly designed, the AI might exploit loopholes to achieve “high scores” in unintended ways.

Consider a few examples:

An agent tasked with maximizing social media engagement that prioritizes sensational or misleading content, inadvertently spreading misinformation

A warehouse robot optimizing for speed that damages products to move faster.

A financial trading AI meant to maximize profits that engages in risky or unethical trading practices, triggering market instability.

A content moderation AI designed to reduce harmful speech overcensors legitimate discussions.

Some agentic AI systems can become self-reinforcing, escalating behaviors in an unintended direction. This issue happens when the AI optimizes too aggressively for a particular metric without safeguards. And because agentic systems are often composed of multiple autonomous agents working together, there are opportunities for failure. Traffic jams, bottlenecks, resource conflicts—all of these errors have the potential to cascade. It’s important for models to have clearly-defined goals that can be measured, with feedback loops in place so models can move ever closer to the organization’s intention over time.

**LangGraph**

LangGraph is an **open-source framework for building stateful, multi-agent applications on top of large language models (LLMs)**.

It’s developed by **LangChain** and is designed for cases where you want more than just a single prompt/response interaction. Instead, you can create **graphs of reasoning steps, tools, and agents** that interact in structured workflows.

### Key ideas of LangGraph:

* **Graphs of computation**:  
  Instead of a simple chain of prompts, you define a directed graph where nodes are steps (like invoking an LLM, calling a tool, or running custom logic) and edges decide how the workflow progresses.
* **State management**:  
  Keeps track of intermediate results (conversation history, variables, agent states) across multiple steps and iterations.
* **Multi-agent support**:  
  Lets you model interactions between different agents (e.g., a "research agent" + "writing agent") in a controlled loop.
* **Control flow**:  
  Unlike plain LangChain, LangGraph provides loops, conditionals, and branching logic so agents can retry, refine answers, or collaborate.
* **Persistence**:  
  Built-in support for saving state (using Redis, SQL, MongoDB, etc.), so agents can be paused/resumed — useful for long-running apps like copilots, customer support bots, or workflow orchestrators.

### Example use cases:

* Multi-step workflows (research → draft → review → finalize).
* Agentic apps (multiple agents collaborating on a task).
* Conversational bots that need memory and tool use.
* Complex LLM pipelines that require retries and branching.

👉 In short: **LangGraph = LangChain + graphs + memory + multi-agent orchestration.**