What is BeeAI?

BeeAI is an open source platform that provides a centralized place to discover, run and share AI agents across frameworks. Developed by IBM, BeeAI is built on the Agent Communication Protocol (ACP) and hosted on the Linux Foundation. Teams can use the BeeAI framework to deploy agents outside of their respective siloed ecosystems.

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe here. Refer to our IBM Privacy Statement for more information.

What does BeeAI do?

BeeAI provides both individual developers and teams with a framework-agnostic platform in which to find, deploy and share artificial intelligence (AI) agents. The platform was designed to address the three primary challenges when working with AI agents:

- **Siloed ecosystems:** Each agent exists within its own framework. BeeAI pulls agents together into a single centralized workspace for streamlined AI agent orchestration.
- **Limited scalability:** BeeAI allows users to deploy agents without having to deal with complicated, individualized setup procedures.
- **Fragmented discovery:** Bee agents are located in a centralized discovery hub, making it easy to find and experiment with agentic AI.

Individual developers can use Bee to streamline the process of exploring and deploying agents for use in agentic automation and other contexts. Meanwhile, teams can share the same BeeAI workspace through a centralized instance for shared work in real time while centrally managing large language model (LLM) connections and APIs.

The community catalog hosts all the BeeAI agents available on the platform, from which they can be deployed without complicated setup. Standardized user interfaces make for consistent user

experiences, and standard containers enable developers to package agents from any framework while bypassing compatibility issues.

Using agents in BeeAI

The IBM research team built BeeAI around a suite of core features that enable its functionality as an agentic workspace. These include:

Agent catalog

BeeAI's repository for agentic AI brings the team together in one centralized workspace for smoother multiagent workflows. The BeeAI agent catalog is one of its key features: it is searchable and features capability details for each agent on offer. Developers can identify usage patterns and choose agents accordingly.

Agents are sorted by type. BeeAI presents conversational chat agents through a chatbot interface. Meanwhile, hands-off agents form the backbone of many agentic workflows, because they are designed to work autonomously after receiving a single instruction.

The community catalog hosts user-created agents, and users can also push agents they've built to GitHub through the BeeAI interface.

Framework-agnostic environments

BeeAI uses ACP to standardize agent use regardless of individual frameworks. Developers use the tools they prefer with the agents they want. The interactive setup wizard streamlines the process of environment creation to get teams up and running with shared AI agent workspaces.

Setup

The setup process includes API key entry, recommendations for model selection, connection testing and provider-specific options, such as Ollama's context window. Available LLM providers include Anthropic's Claude, OpenAI's GPT, DeepSeek and IBM's watsonx. Meta's Llama3 is available through a local Ollama connection.

Users can import agents locally or from GitHub repositories, other frameworks such as LangChain and even build their own agents for use in BeeAI.

Running agents

BeeAI runs each agent in its own container with defined resource limits, enabling modular multi agent system construction. Input options include an interactive mode for communication with the agent and multi-line input for sharing snippets of Python code and other languages. Standardized user interfaces mean that agent interactions within agentic workflows are predictable.

Observability is built into the platform through the streaming of real-time logs from any running agent. BeeAI collects telemetry data with OpenTelemetry and sends it to a designated Arize Phoenix instance.

How does BeeAI work?

BeeAI is designed around a local-first experience, hosting agents on individual devices or onsite to grant users full control over their data. The core components include:

- **Agents:** Agents in BeeAI are containerized and communicate through ACP. One of the defining characteristics of AI agents is the ability to call tools as needed to expand their capabilities.
- **BeeAI server:** The server orchestrates between agents, managing lifecycles and configurations, routing communication between agents and clients and collecting telemetry data.
- **BeeAI CLI and UI:** Users interact with BeeAI through two modes. The command line interface (CLI) facilitates scripting and command control, while the graphical user interface (UI) handles more intuitive interactions such as conversational chats.
- **Python integration:** The ACP SDK (software development kit) enables developers to integrate BeeAI into their Python-based applications. BeeAI can handle agent workflows within the context of Python apps, such as those designed for task automation.
- **Arise Phoenix for monitoring:** Available in BeeAI, Phoenix is an open source tool for tracing and monitoring agent behavior.