The Open-Source Stack for AI Agents

I remember sitting down one weekend, convinced I was finally going to build a decent prototype of a research assistant agent. Nothing fancy...

Paolo Perrone

Follow

Data Science Collective a11y-light ~10 min read · April 21, 2025 (Updated: April 21, 2025) · Free: Yes

I remember sitting down one weekend, convinced I was finally going to build a decent prototype of a research assistant agent. Nothing fancy — just something that could read a PDF, extract key info, maybe answer a few follow-up questions. Should've been straightforward, right?

Instead, I spent the better part of two days hopping between half-documented repos, dead GitHub issues, and vague blog posts. One tool looked promising until I realized it hadn't been updated in eight months. Another required spinning up four different services just to parse a single document. By the end of it, my "agent" could barely read the file name, let alone the contents.

But the thing that kept me going wasn't frustration — it was curiosity. I wanted to know: *What are the tools that actual builders use?* Not

three Notion pages to explain.

That search led me to a surprisingly solid set of open-source libraries — tools that are lightweight, reliable, and built with developers in mind.

So if you're in the trenches trying to get agents to actually work, this one's for you.

So, you're ready to build AI agents?

Awesome.

You might be asking:

- What do people use to build voice agents?
- What's the best open-source tool for document parsing?
- How do I give my agent memory without duct-taping a vector DB to everything?

This guide doesn't try to cover everything out there — and that's intentional. It's a curated list of tools I've actually used, kept in my stack, and returned to when building real agent prototypes. Not the ones that looked cool in a demo or showed up in every hype thread, but the ones that helped me move from "idea" to "working thing" without getting lost.

Here's the stack, broken down into categories:

1. **Frameworks for Building and Orchestrating Agents**

handle tools. Think of this as the core brain that turns a raw language model into something more autonomous.

2. Computer and Browser Use

Once your agent can plan, it needs to act. This category includes tools that let your agent click buttons, type into fields, scrape data, and generally control apps or websites like a human would.

3. Voice

If your agent needs to speak or listen, these tools handle the audio side — turning speech into text, and back again. Useful for hands-free use cases or voice-first agents. Some are even good enough for real-time conversations.

4. Document Understanding

Lots of real-world data lives in PDFs, scans, or other messy formats. These tools help your agent actually read and make sense of that content — whether it's invoices, contracts, or image-based files.

5. Memory

To go beyond one-shot tasks, your agent needs memory. These libraries help it remember what just happened, what you've told it before, or even build a long-term profile over time.

6. Testing and Evaluation

checking if the agent's behavior makes sense.

7. Monitoring and Observability

Once your agent is live, you need to know what it's doing and how well it's performing. These tools help you track usage, debug issues, and understand cost or latency impacts.

8. Simulation

Before throwing your agent into the wild, test it in a safe, sandboxed world. Simulated environments let you experiment, refine decision logic, and find edge cases in a controlled setting.

9. Vertical Agents

Not everything needs to be built from zero. These are ready-made agents built for specific jobs — like coding, research, or customer support. You can run them as-is or customize them to fit your workflow.

The Components of the Open-Source Stack for AI Agents

1. Frameworks for Building and Orchestrating Agents

To build agents that actually get things done, you need a solid foundation — something to handle workflows, memory, and tool integration without becoming a mess of scripts. These frameworks

- **CrewAI** — Orchestrates multiple agents working together. Ideal for tasks that need coordination and role-based behavior.
- **Phidata** — Focuses on memory, tool use, and long-term interactions. Great for assistants that need to remember and adapt.
- **Camel** — Designed for multi-agent collaboration, simulation, and task specialization.
- **AutoGPT** — Automates complex workflows with a loop of planning and execution. Best for agents that need to run independently.
- **AutoGen**—Lets agents communicate with each other to solve complex problems.
- **SuperAGI** — Streamlined setup for building and shipping autonomous agents fast.
- **Superagent** — A flexible open-source toolkit to create custom AI assistants.
- **LangChain** & **LlamaIndex** — The go-to tools for managing memory, retrieval, and toolchains.

2. Computer and Browser Use

Once your agent can think, the next step is helping it *do*. That means interacting with computers and the web the way a human would — clicking buttons, filling out forms, navigating pages, and running commands. These tools bridge the gap between reasoning and action, letting your agent operate in the real world.

script? Just describe it.

- **Self-Operating Computer** — Gives agents full control of your desktop environment, allowing them to interact with your OS like a person would.
- **Agent-S** — A flexible framework that lets AI agents use apps, tools, and interfaces like a real user.
- **LaVague** — Enables web agents to navigate sites, fill forms, and make decisions in real time — ideal for automating browser tasks.
- **Playwright** — Automates web actions across browsers. Handy for testing or simulating user flows.
- **Puppeteer** — A reliable tool for controlling Chrome or Firefox. Great for scraping and automating front-end behavior.

3. Voice

Voice is one of the most intuitive ways for humans to interact with AI agents. These tools handle speech recognition, voice synthesis and rea-time interactions — making your agent feel a bit more human.

Speech2speech

- **Ultravox** — A top-tier speech-to-speech model that handles real-time voice conversations smoothly. Fast and responsive.
- **Moshi** — Another strong option for speech-to-speech tasks. Reliable for live voice interaction, though Ultravox has the edge on performance.
- **Pipecat** — A full-stack framework for building voice-enabled agents. Includes support for speech-to-text, text-to-speech, and

Speech2text

- **Whisper** — OpenAI's speech-to-text model — great for transcription and speech recognition across multiple languages.
- **Stable-ts** — A more developer-friendly wrapper around Whisper. Adds timestamps and real-time support, making it great for conversational agents.
- **Speaker Diarization 3.1** — Pyannote's model for detecting who's speaking when. Crucial for multi-speaker conversations and meeting-style audio.

Text2speech

- **ChatTTS** — The best model I've found so far. It's fast, stable, and production-ready for most use cases.
- **ElevenLabs** (Commercial)— When quality matters more than open source, this is the go-to. It delivers highly natural-sounding voices and supports multiple styles.
- **Cartesia** (Commercial) — Another strong commercial option if you're looking for expressive, high-fidelity voice synthesis beyond what open models can offer.

Miscellaneous Tools

These don't fit neatly into one category but are very useful when building or refining voice-capable agents.

- **Vocode** — A toolkit for building voice-powered LLM agents. Makes it easy to connect speech input/output with language models.

model setup.

4. Document Understanding

Most useful business data still lives in unstructured formats — PDFs, scans, image-based reports. These tools help your agent read, extract, and make sense of that mess, without needing brittle OCR pipelines.

- **Qwen2-VL** — A powerful vision-language model from Alibaba. Outperforms GPT-4 and Claude 3.5 Sonnet on document tasks that mix images and text — great for handling complex, real-world formats.
- **DocOwl2** — A lightweight multimodal model built for document understanding *without* OCR. Fast, efficient, and surprisingly accurate for extracting structure and meaning from messy inputs.

5. Memory

Without memory, agents are stuck in a loop — treating every interaction like the first. These tools give them the ability to recall past conversations, track preferences, and build continuity. That's what turns a one-shot assistant into something more useful over time.

- **Memo** — A self-improving memory layer that lets your agent adapt to previous interactions. Great for building more personalized and persistent AI experiences.
- **Letta (formerly MemGPT)** — Adds long-term memory and tool use to LLM agents. Think of it as scaffolding for agents that need to remember, reason, and evolve.

building agents that need to stay grounded across multiple turns.

6. Testing and Evaluation

As your agents start doing more than just chatting — navigating web pages, making decisions, speaking out loud — you need to know how they'll handle edge cases. These tools help you test how your agents behave in different situations, catch bugs early, and track where things break down.

- **eeVoice Lab** — A comprehensive framework for testing voice agents, ensuring your agent's speech recognition and responses are accurate and natural.
- **AgentOps** — A set of tools for tracking and benchmarking AI agents, helping you spot any issues and optimize performance before they impact users.
- **AgentBench** — A benchmark tool for evaluating LLM agents across various tasks and environments, from web browsing to gaming, ensuring versatility and effectiveness.

7. Monitoring and Observability

To ensure your AI agents run smoothly and efficiently at scale, you need visibility into their performance and resource usage. These tools provide the necessary insights, allowing you to monitor agent behavior, optimize resources, and catch issues before they impact users.

- **openllmetry** — Provides end-to-end observability for LLM applications using OpenTelemetry, giving you a clear view of

- **AgentOps** — A comprehensive monitoring tool that tracks agent performance, cost, and benchmarking, helping you ensure your agents are efficient and within budget.

8. Simulation

Simulating real-world environments before deployment is a game-changer. These tools let you create controlled, virtual spaces where your agents can interact, learn, and make decisions without the risk of unintended consequences in live environments.

- **AgentVerse** — Supports deploying multiple LLM-based agents across diverse applications and simulations, ensuring effective functioning in various environments.
- **Tau-Bench** — A benchmarking tool that evaluates agent-user interactions in specific industries like retail or airlines, ensuring smooth handling of domain-specific tasks.
- **ChatArena** — A multi-agent language game environment where agents interact, ideal for studying agent behavior and refining communication patterns in a safe, controlled space.
- **AI Town** — A virtual environment where AI characters interact socially, test decision-making, and simulate real-world scenarios, helping to fine-tune agent behavior.
- **Generative Agents** — A Stanford project focused on creating human-like agents that simulate complex behaviors, perfect for testing memory and decision-making in social contexts.

9. Vertical Agents

growing ecosystem of these, here are a few that I've personally used and found particularly useful:

Coding:

- **OpenHands** — A platform for software development agents powered by AI, designed to automate coding tasks and speed up the development process.
- **aider**— A pair programming tool that integrates directly with your terminal, offering an AI co-pilot to assist right in your coding environment.
- **GPT Engineer**— Build applications using natural language; simply describe what you want, and the AI will clarify and generate the necessary code.
- **screenshot-to-code** — Converts screenshots into fully functional websites with HTML, Tailwind, React, or Vue, great for turning design ideas into live code quickly.

Research:

- **GPT Researcher**—An autonomous agent that conducts comprehensive research, analyzes data, and writes reports, streamlining the research process.

SQL:

- **Vanna** — Interact with your SQL database using natural language queries; no more complicated SQL commands, just ask questions, and Vanna retrieves the data.

Conclusion

Reflecting on my early attempts to build a research assistant, I can see I was overcomplicating things. The project turned out to be a mess — outdated code, half-baked tools, and a system that struggled with something as simple as a PDF.

But, paradoxically, that's where I learned the most.

It wasn't about finding the perfect tool; it was about sticking to what works and keeping it simple. That failure taught me that the most reliable agents are built with a pragmatic, straightforward stack — not by chasing every shiny new tool.

Successful agent development doesn't require reinventing the wheel.

It's about choosing the right tools for the job, integrating them thoughtfully, and refining your prototypes. Whether you're automating workflows, building voice agents, or parsing documents, a well-chosen stack can make the process smoother and more efficient.

So, get started, experiment, and let curiosity guide you. The ecosystem is evolving, and the possibilities are endless.

Want to hear from me more often?

👉 **Connect with me on LinkedIn!**

Are you a tech professional looking to grow your audience through writing?

👉 **Don't miss my newsletter!**

My **Tech Audience Accelerator** is packed with actionable copywriting and audience building strategies that have helped hundreds of professionals stand out and accelerate their growth. Subscribe now to stay in the loop:

#ai #ai-agent #open-source #machine-learning #machine-learning-ai