Weights & Biases
by CoreWeave

E-Book

A primer on building successful AI agents

Introduction

Key takeaway

AI agents have significant potential to enhance productivity, efficiency, and decision-making provided you implement the right tools and workflows to confidently deploy them in production. Quality, governance, security, and cost are critical concerns. Leveraging agent development tools to evaluate, monitor, and iterate your AI agents accelerates progress toward realizing the full business potential of AI.

Who should read

AI agents are becoming the future of software applications. Any software developer will soon be building agentic applications that use agents trained by AI engineers within or outside their organization. This white paper is relevant for all those AI developers and engineers as well as technical decision makers who need to institute efficient and effective processes for their teams to innovate rapidly and gain competitive advantage through AI.

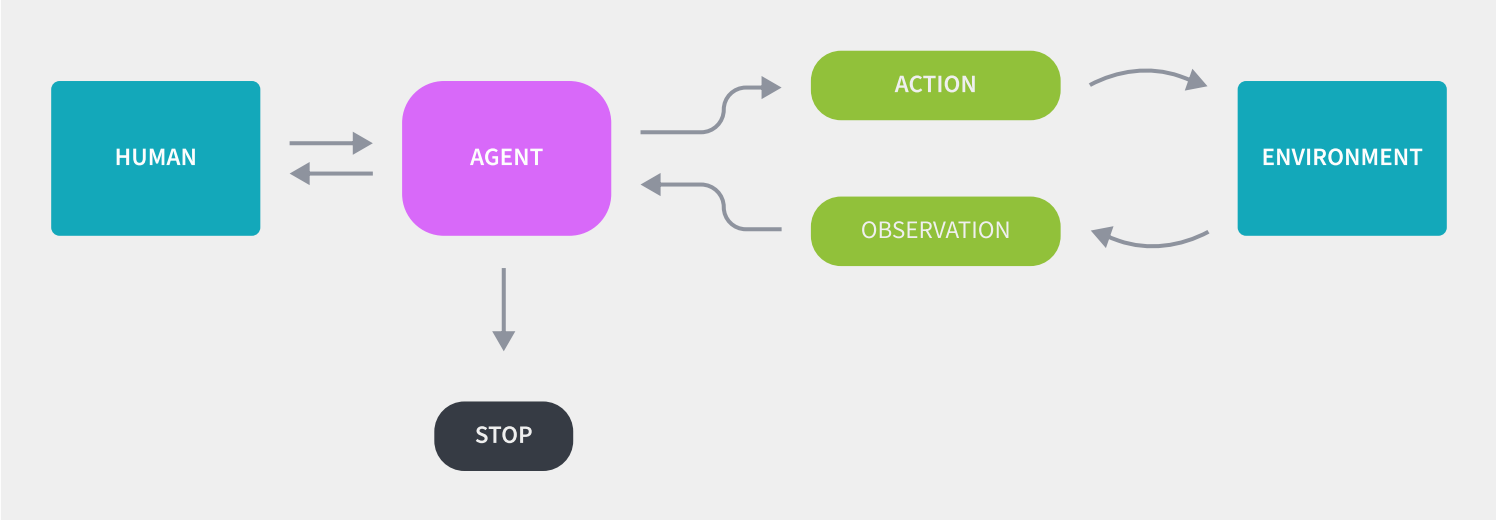
What you'll learn

Gain a good understanding of what constitutes AI agents and agentic applications. Learn the state of the art workflow for building agentic AI applications through an actual example. Get a taste of how pioneering companies are building and deploying agents today. You'll walk away with actionable takeaways and insights to get your organization started on the journey towards the AI era.

What are AI agents?

AI agents are autonomous systems that plan and perform multi-step tasks iteratively until achieving their intended goal to fulfill a user's request. They utilize memory and tools to process data, make decisions, and execute actions independently, with minimal human supervision. Agents consist of a reasoning model wrapped within a harness that manages data pre- and post-processing and coordinates interactions with tools and other agents.

Model builders increasingly embed agentic capabilities into reasoning models through test-time compute and reinforcement fine-tuning. Businesses leverage these models using agent frameworks, protocols, and tools to build applications that navigate complex scenarios, collaborate with humans, and dynamically manage workflows. This white paper focuses on these frameworks and tools, demonstrating how to build agentic applications.



Key elements of agentic applications

Goals

The agent needs to recognize when it has fulfilled the user's request and then terminate the loop. Goals are defined within the system prompts. For example, a customer service agent may be instructed to explicitly ask users if their request has been resolved and use their affirmative response as confirmation of goal completion.

Planning

Use chain-of-thought reasoning to plan the steps required to complete tasks. Reasoning models already possess planning capabilities but need specific prompts to plan effectively. For example, a coding agent must identify which files to edit, design appropriate tests, and execute the code to confirm the bug is fixed.

Tool use

The ability to call functions for tasks like running searches, querying data stores, transacting with enterprise applications, and executing code is essential. Tools must be invoked securely. Agent protocols, such as Anthropic's Model Context Protocol (MCP), standardize how agents discover, invoke, and communicate with tools and other agents.

Memory

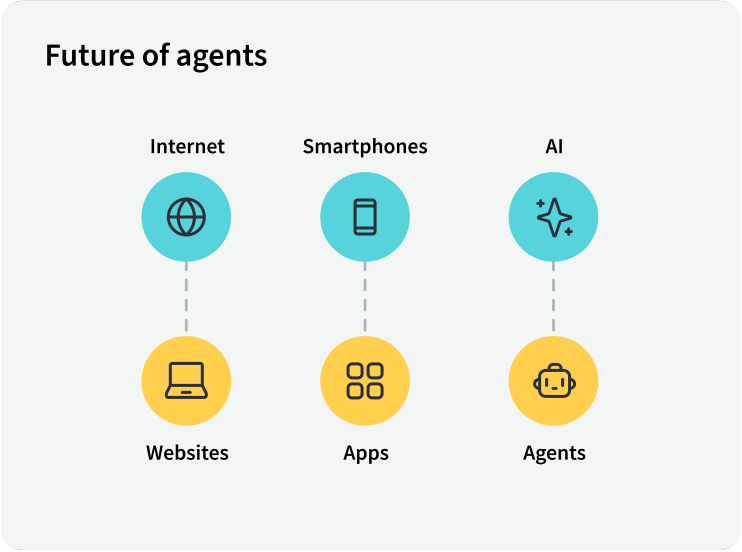
Agents need to retain, apply, and maintain context across interactions. The most common approach is Retrieval Augmented Generation (RAG), which enhances a model provider's built-in memory capabilities. For example, a stock analyst agent can use RAG to retrieve recent news articles for identifying stock picks.

Reflection/verification

One way to improve agent accuracy is to verify its output. Typically, the agent explores multiple trajectories and an LLM acting as a judge evaluates the outputs to select the best. The agent can also pursue one trajectory at a time until it finds an optimal one. This enables the agent to autonomously detect errors and optimize workflows.

The state of agentic AI

Agents will make AI ubiquitous, just as websites did for the internet and apps for smartphones. They offer businesses endless possibilities—from managing legal compliance to innovating product designs and optimizing supply chains. While agents promise to enhance operations, lower costs, and fuel productivity, the technology is in its early stages, with current development focused mainly on agents for coding, research, and customer support.



Today's popular agents in production

Write and execute code

Agentic applications like Cursor and Windsurf are among today's earliest agents because coding is highly valuable in the digital workplace. For instance, a coding agent can handle many company workflows, from addressing customer queries to forecasting sales and processing resumes. They are verifiable: you can execute their code to confirm successful task completion, enabling effective evaluation, iteration, and monitoring.

Find and collate information

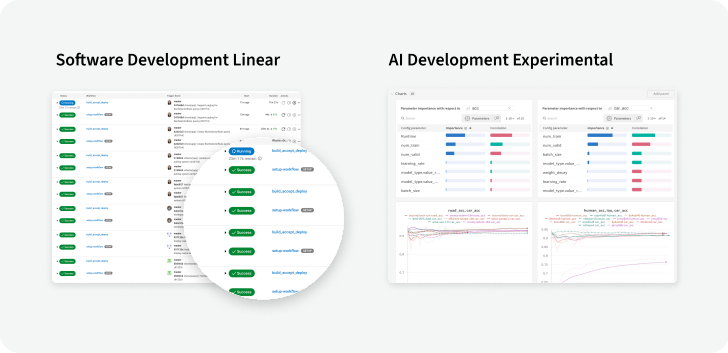
Gathering and synthesizing information from diverse sources to answer business questions is another common task performed by digital workers in enterprises. Research agents, such as those developed by companies like Perplexity have emerged as the next most popular category. While their outputs aren't as easily verifiable, traditional search-scoring techniques, such as TF-IDF, can still be used for validation.

Handle customer requests

Customer agents have evolved from simple assistants and chatbots into advanced systems capable of executing transactions and retrieving customer data from internal sources by coordinating with other agents. Their popularity stems from leveraging the capabilities of coding and research agents. Today, nearly all enterprise agent applications deployed externally to end users are customer service and support agents.

Why aren't agents ubiquitous

AI agents are likely among the most transformative technological advancements in human history, but they aren't yet widespread because developers lack suitable tools. Why?



Observability, the missing link to ubiquity of agents

Today's popular AI agents, like coding and research agents, are typically developed by AI engineers, who deeply understand AI fundamentals and the necessary experimentation tools. However, AI engineers are limited in number. To achieve broader adoption of AI, we need to equip the wider software developer community, most of whom are not yet familiar with AI-specific dev tools and processes. Unlike traditional software development, where outcomes are predictable and linear, building AI agents involves significant uncertainty.

Standard software development is linear and deterministic, characterized by clear cause-and-effect relationships between code changes and predictable outcomes. For instance, fixing a software bug usually means changing some logic in your code and seeing exactly the result you expected. In contrast, AI development is intrinsically non-deterministic. It's difficult to predict precisely how changes, such as tweaking model parameters and the architecture, will affect the response of an AI agent. Therefore, experimentation and empirical analysis are required to develop agents. For instance, improving an AI agent's accuracy cannot rely on theoretical predictions; it necessitates systematic experimentation with various hyperparameters, analyzing empirical results, and identifying optimal configurations. Furthermore, enhancements are not binary successes or failures. Instead, improvements are multi-dimensional, involving trade-offs such as achieving higher accuracy potentially at the expense of some performance.

In summary, successful AI agent development requires a new set of tools for evaluation, tracing, and monitoring, collectively referred to as observability tools. These observability tools complement conventional programming tools including debuggers, frameworks, and protocols. They are essential not just for production deployment but are equally critical during the development phase of AI agents.

Weights & Biases platform

Weights & Biases provides W&B Weave, a developer-friendly observability platform designed to support experimentation, evaluation, and analytics seamlessly throughout development and into production. Weave aims to help developers easily incorporate experimentation and analytics into their regular workflow. Weave integrates effortlessly with popular agent frameworks and protocols, offering developers the flexibility to use their preferred libraries.

Agent development tools

Frameworks

Agent frameworks are software libraries that simplify the creation, orchestration, and execution of AI agents. They provide abstractions for planning, memory, tool use, and multi-agent coordination, enabling developers to focus on logic rather than low-level details. The benefit is faster iteration and experimentation with complex agent behaviors, allowing teams to build more capable and reliable systems with less effort. Popular agent frameworks: OpenAI Agents SDK, CrewAI, and Google ADK.

Protocols

Agent protocols are standardized interfaces and communication conventions that define how AI agents interact with tools, systems, or other agents. They provide structured formats and rules for requests, responses, and data exchange, ensuring consistency and interoperability. The benefit is streamlined collaboration and integration, allowing diverse agents and systems to communicate seamlessly without custom implementations. Popular agent protocols: Anthropic's Model Context Protocol (MCP) and Google's Agent2Agent (A2A).

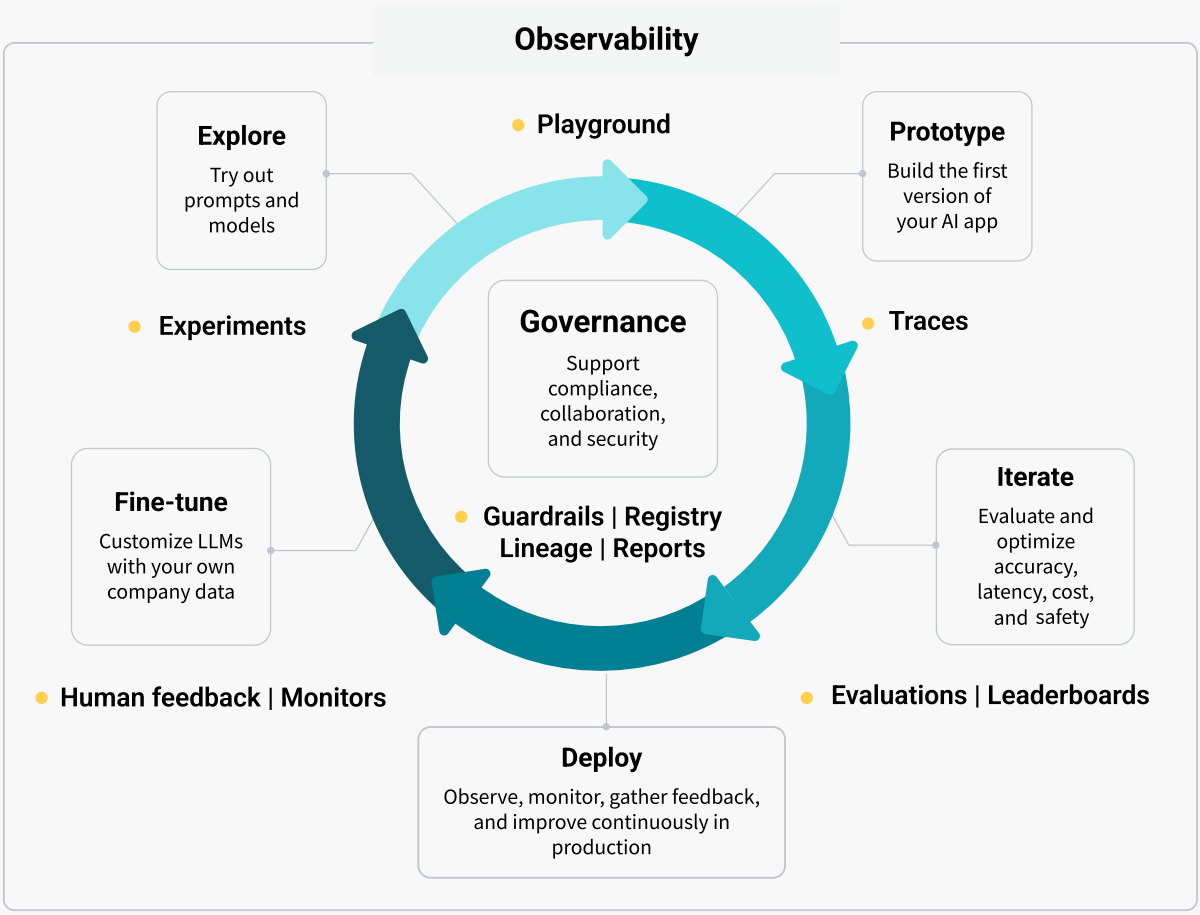
Observability

Observability tools provide tracing, evaluation, monitoring, and governance capabilities for the end-to-end AI agent workflow both during the development stage and in post-production. These tools help developers debug agents, iterate on them, deploy in production, collect end user and expert feedback, and build datasets to fine-tune the underlying large language models (LLMs). Observability tools also include guardrails, registry, lineage, and reporting tools for governance throughout the lifecycle. Weights & Biases offers W&B Weave, W&B Models, and W&B Inference for AI agent observability.

Agent development workflow

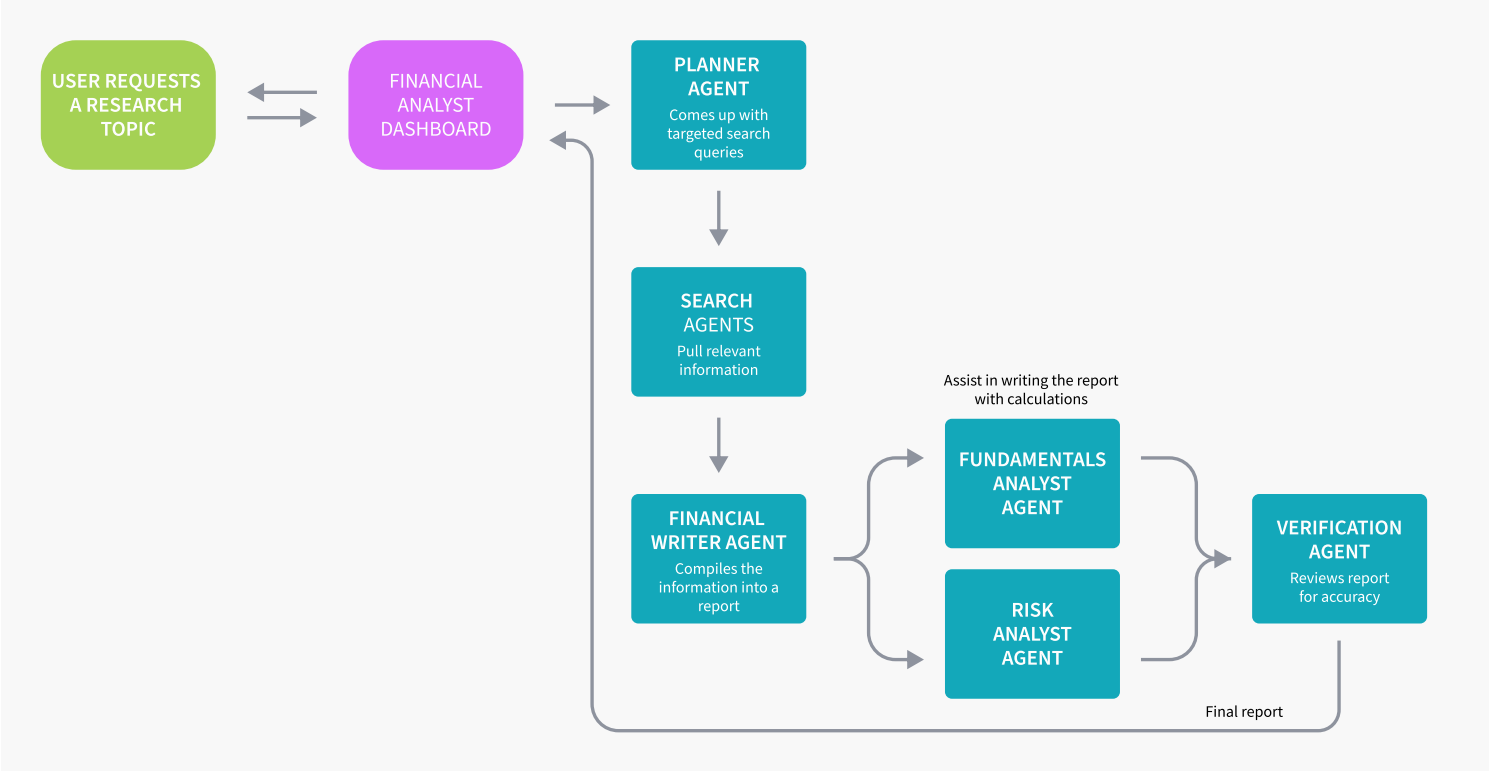
A typical agent development workflow begins by exploring models and prompts to understand LLM responses for various agent tasks. W&B Weave offers an intuitive Playground, allowing you to use proprietary models such as those from OpenAI and Anthropic, open-source models like DeepSeek and Llama, and custom LLMs. After selecting your model and baseline prompts, you move into the prototyping phase to build the initial version of your application. Weave's powerful tracing capabilities help you quickly debug your system, providing specialized trace views for agentic AI and extensive multimodal support. These tools offer clear visibility into data flows and decision points, enabling rapid troubleshooting.

Next, you iterate on your application by conducting experiments and leveraging real-time evaluation for online learning. Weave Evaluations facilitate scalable experimentation with a flexible framework featuring pre-built scorers and compatibility with third-party or custom scorers. Once your agent meets desired performance levels, you deploy it into production, gather real-world user feedback, and continuously refine its capabilities. Weave's monitoring tools support ongoing online evaluations, enabling you to quickly identify and resolve issues while tracking performance trends over time. Additionally, production-collected traces can be utilized to build high-quality datasets for fine-tuning the underlying LLMs, leveraging W&B Models, an industry-leading platform for managing and training customized models.



Financial research agent: a real-life example

Now that we get what an agent is and how the development workflow usually goes, let's check out a real-life example. Inspired by OpenAI's agent examples, we built a financial research agent that quickly gathers stock quotes and market data. It helps investors make sense of recent market trends, answering questions like, "Which NASDAQ sectors have been the most resilient over the past month?" When you submit a research topic via the financial analyst dashboard, the Planner Agent kicks off by creating a plan with 5-15 targeted search queries. Once it collects all the info, a Financial Writer Agent summarizes the findings with help from the Fundamentals Analyst and Risk Analyst Agents. Finally, a Verification Agent reviews everything for accuracy. After all that, the finished report is ready for the user right in the dashboard.

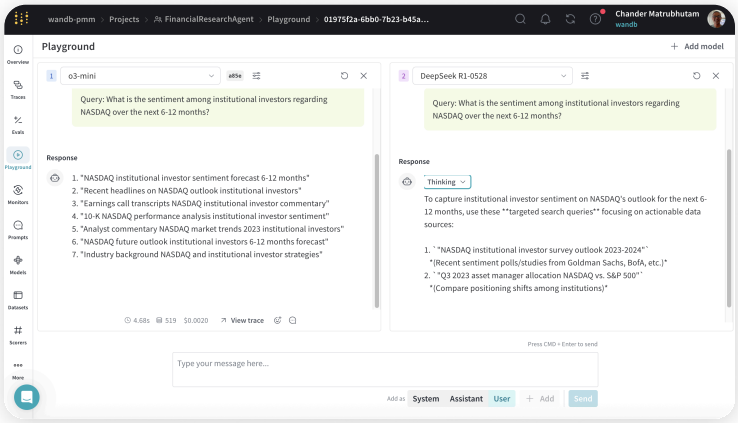


A walk through the workflow

Let us walk you through the steps we took to build the Financial Research Agent. Follow along step-by-step in this demo video. Below is a brief summary of each step.

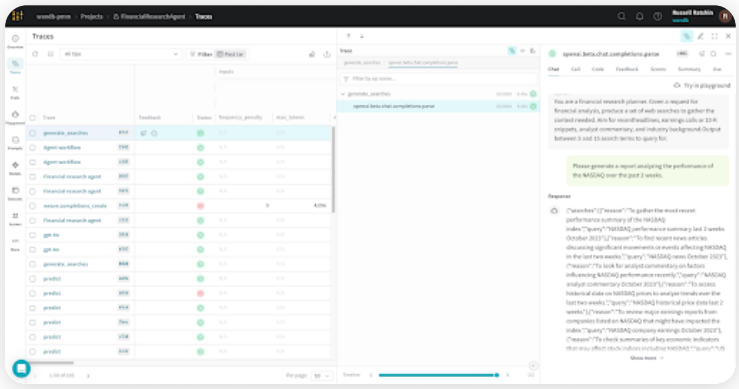
Explore

Every agent development process has to start somewhere, so let's begin with the first step: experimenting with various LLMs and prompts in the Weave Playground. We test typical user queries, assess the quality of the responses, and gain insight into latency, token usage, and cost. Finally, we compare different LLMs side-by-side for our specific use cases.



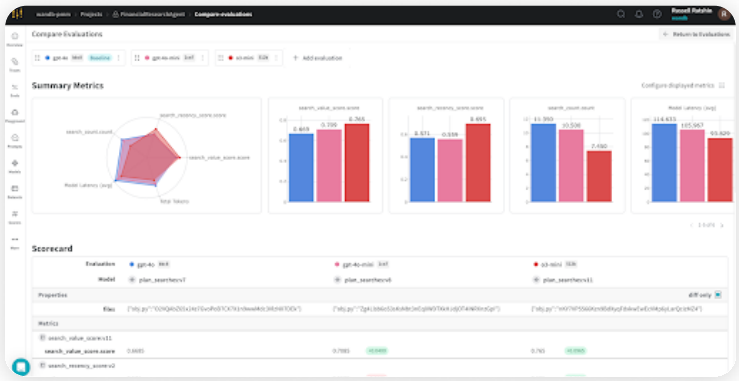
Prototype

Next, we prototype the Planner Agent in a notebook using Python, starting with an LLM call that generates queries for individual Search Agents. We add one line of code to initialize Weave and a few Weave.op decorators to enhance trace data. After executing the call, we review it in Weave, drill into outputs to see generated queries, and send it to the Playground for further exploration.



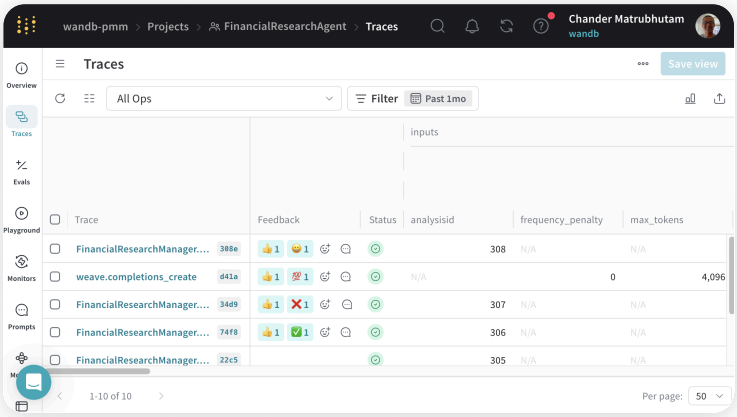
Iterate

Our prototype is functional but needs refinement before deployment, so we iterate on each agent task. For the generate_searches task, we evaluate using three scorers (topical relevance, temporal accuracy, and search count) across GPT-4o-mini, GPT-4o, and o3-mini. Weave's Evaluations page enables easy comparison, clearly indicating o3-mini performs best.



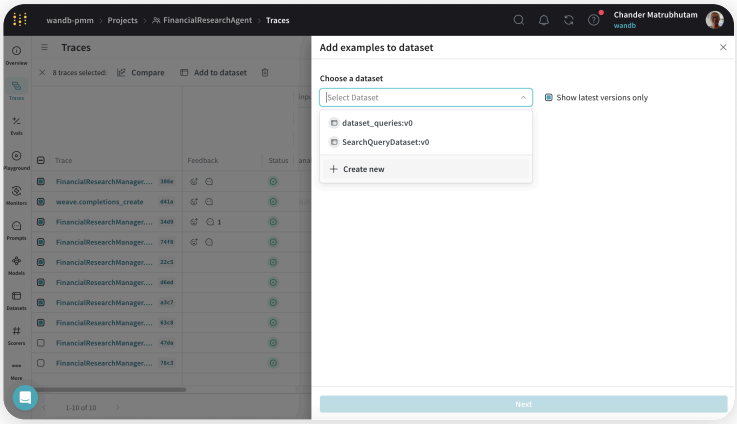
Deploy

After deploying our AI agent in production, we're continually monitoring its performance and exploring potential updates to enhance both the application and the end-user experience. We refine, optimize, and introduce new features, leveraging Weave to easily measure their impact. Additionally, we collect human feedback from real users alongside programmatic scores within the traces.



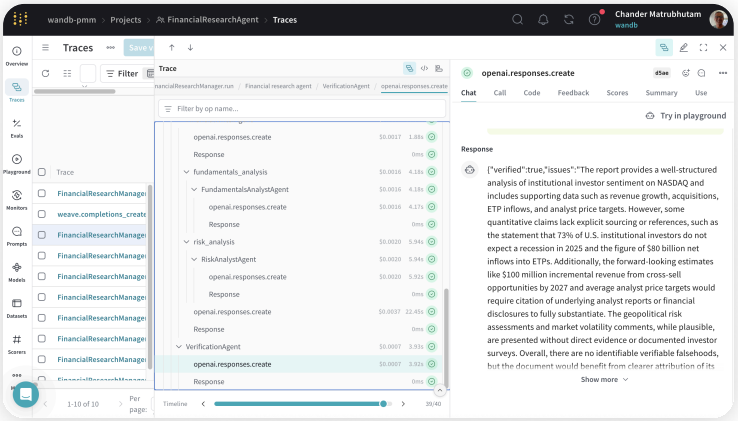
Fine-tune

As we collect real-world examples in production, scores can be easily recorded in Weave, used to filter traces, and added to datasets for fine-tuning of the underlying LLM. W&B Models enables fine-tuning of these LLMs and allows you to share them across your team through the W&B Registry. This method can meaningfully boost the agent's accuracy and reliability.



Governance

Our system includes a Verification Agent that audits reports, checking internal consistency, proper sourcing, and well-supported claims. For example, the Verification Agent might pass a report but still highlight areas needing improvement, or, if the report fails, prompt the planner to involve a human. This approach demonstrates the guardrails we implement to safeguard our agent.



Resources to get started

01

Free online course: Created in collaboration between experts from OpenAI and Weights & Biases, this free agents course is the easiest way to get started.

02

Watch our agents webinar: In this webinar, learn how to build AI agents with Model Context Protocol (MCP) and other protocols.

03

Run a Proof of Concept (POC): Begin with a small-scale pilot and scale swiftly as value is demonstrated.

04

Watch a demo: In this video demonstration, we show how W&B Weave provides the necessary tools for rapidly iterating and optimizing agents.

05

Engage our experts: Weights & Biases AI Advisory Services provide specialized consultative help and guidance in developing and productionizing your agent.

Build your first agent

Don't wait! Get started with just a few lines of code. Contact us for a free demo and a deep dive into your agent use case.

```python
import weave
weave.init("quickstart")
@weave.op()
def llm_app(prompt):
```