Agents

======

Learn how to build agents with the OpenAI API.

Agents represent \*\*systems that intelligently accomplish tasks\*\*, ranging from executing simple workflows to pursuing complex, open-ended objectives.

OpenAI provides a \*\*rich set of composable primitives that enable you to build agents\*\*. This guide walks through those primitives, and how they come together to form a robust agentic platform.

Overview

--------

Building agents involves assembling components across several domains—such as \*\*models, tools, knowledge and memory, audio and speech, guardrails, and orchestration\*\*—and OpenAI provides composable primitives for each.

|Domain|Description|OpenAI Primitives|

|---|---|---|

|Models|Core intelligence capable of reasoning, making decisions, and processing different modalities.|o1, o3-mini, GPT-4.5, GPT-4o, GPT-4o-mini|

|Tools|Interface to the world, interact with environment, function calling, built-in tools, etc.|Function calling, Web search, File search, Computer use|

|Knowledge and memory|Augment agents with external and persistent knowledge.|Vector stores, File search, Embeddings|

|Audio and speech|Create agents that can understand audio and respond back in natural language.|Audio generation, realtime, Audio agents|

|Guardrails|Prevent irrelevant, harmful, or undesirable behavior.|Moderation, Instruction hierarchy|

|Orchestration|Develop, deploy, monitor, and improve agents.|Agents SDK, Tracing, Evaluations, Fine-tuning|

|Voice agents|Create agents that can understand audio and respond back in natural language.|Realtime API, Voice support in the Agents SDK|

Models

------

|Model|Agentic Strengths|

|---|---|

|o1 and o3-mini|Best for long-term planning, hard tasks, and reasoning.|

|GPT-4.5|Best for agentic execution.|

|GPT-4o|Good balance of agentic capability and latency.|

|GPT-4o-mini|Best for low-latency.|

Large language models (LLMs) are at the core of many agentic systems, responsible for making decisions and interacting with the world. OpenAI’s models support a wide range of capabilities:

\* \*\*High intelligence:\*\* Capable of [reasoning](/docs/guides/reasoning) and planning to tackle the most difficult tasks.

\* \*\*Tools:\*\* [Call your functions](/docs/guides/function-calling) and leverage OpenAI's [built-in tools](/docs/guides/tools).

\* \*\*Multimodality:\*\* Natively understand text, images, audio, code, and documents.

\* \*\*Low-latency:\*\* Support for [real-time audio](/docs/guides/realtime) conversations and smaller, faster models.

For detailed model comparisons, visit the [models](/docs/models) page.

Tools

-----

Tools enable agents to interact with the world. OpenAI supports [\*\*function calling\*\*](/docs/guides/function-calling) to connect with your code, and [\*\*built-in tools\*\*](/docs/guides/tools) for common tasks like web searches and data retrieval.

|Tool|Description|

|---|---|

|Function calling|Interact with developer-defined code.|

|Web search|Fetch up-to-date information from the web.|

|File search|Perform semantic search across your documents.|

|Computer use|Understand and control a computer or browser.|

Knowledge and memory

--------------------

Knowledge and memory help agents store, retrieve, and utilize information beyond their initial training data. \*\*Vector stores\*\* enable agents to search your documents semantically and retrieve relevant information at runtime. Meanwhile, \*\*embeddings\*\* represent data efficiently for quick retrieval, powering dynamic knowledge solutions and long-term agent memory. You can integrate your data using OpenAI’s [vector stores](/docs/guides/retrieval#vector-stores) and [Embeddings API](/docs/guides/embeddings).

Guardrails

----------

Guardrails ensure your agents behave safely, consistently, and within your intended boundaries—critical for production deployments. Use OpenAI’s free [Moderation API](/docs/guides/moderation) to automatically filter unsafe content. Further control your agent’s behavior by leveraging the [instruction hierarchy](https://openai.github.io/openai-agents-python/guardrails/), which prioritizes developer-defined prompts and mitigates unwanted agent behaviors.

Orchestration

-------------

Building agents is a process. OpenAI provides tools to effectively build, deploy, monitor, evaluate, and improve agentic systems.

![Agent Traces UI in OpenAI Dashboard](https://cdn.openai.com/API/docs/images/orchestration.png)

|Phase|Description|OpenAI Primitives|

|---|---|---|

|Build and deploy|Rapidly build agents, enforce guardrails, and handle conversational flows using the Agents SDK.|Agents SDK|

|Monitor|Observe agent behavior in real-time, debug issues, and gain insights through tracing.|Tracing|

|Evaluate and improve|Measure agent performance, identify areas for improvement, and refine your agents.|EvaluationsFine-tuning|

Get started

-----------

Get started by installing the [OpenAI Agents SDK for Python](https://github.com/openai/openai-agents-python) via:

```text

pip install openai-agents

```

Explore the [repository](https://github.com/openai/openai-agents-python) and [documentation](https://openai.github.io/openai-agents-python/) for more details.

Voice agents

============

Learn how to build voice agents that can understand audio and respond back in natural language.

Use the OpenAI API and Agents SDK to create powerful, context-aware voice agents for applications like customer support and language tutoring. This guide helps you design and build a voice agent.

Choose the right architecture

-----------------------------

OpenAI provides two primary architectures for building voice agents:

1. Speech-to-speech (multimodal)

2. Chained (speech-to-text → LLM → text-to-speech)

### Speech-to-speech (multimodal) architecture

The multimodal speech-to-speech (S2S) architecture directly processes audio inputs and outputs, handling speech in real time in a single multimodal model, `gpt-4o-realtime-preview`. The model thinks and responds in speech. It doesn't rely on a transcript of the user's input—it hears emotion and intent, filters out noise, and responds directly in speech. Use this approach for highly interactive, low-latency, conversational use cases.

|Strengths|Best for|

|---|---|

|Low latency interactions|Interactive and unstructured conversations|

|Rich multimodal understanding (audio and text simultaneously)|Language tutoring and interactive learning experiences|

|Natural, fluid conversational flow|Conversational search and discovery|

|Enhanced user experience through vocal context understanding|Interactive customer service scenarios|

### Chained architecture

A chained architecture processes audio sequentially, converting audio to text, generating intelligent responses using large language models (LLMs), and synthesizing audio from text. We recommend this predictable architecture if you're new to building voice agents. Both the user input and model's response are in text, so you have a transcript and can control what happens in your application. It's also a reliable way to convert an existing LLM-based application into a voice agent.

You're chaining these models: `gpt-4o-transcribe` → `gpt-4o` → `gpt-4o-mini-tts`

|Strengths|Best for|

|---|---|

|High control and transparency|Structured workflows focused on specific user objectives|

|Robust function calling and structured interactions|Customer support|

|Reliable, predictable responses|Sales and inbound triage|

|Support for extended conversational context|Scenarios that involve transcripts and scripted responses|

Build a voice agent

-------------------

Use OpenAI's APIs and SDKs to create powerful, context-aware voice agents.

### Use a speech-to-speech architecture for realtime processing

Building a speech-to-speech voice agent requires:

1. Establishing a connection for realtime data transfer

2. Creating a realtime session with the Realtime API

3. Using an OpenAI model with realtime audio input and output capabilities

To get started, read [the Realtime API guide](/docs/guides/realtime) and [the Realtime API reference](/docs/api-reference/realtime-sessions/create). Compatible models include `gpt-4o-realtime-preview` and `gpt-4o-mini-realtime-preview`.

### Chain together audio input → text processing → audio output

The Agents SDK supports extending your existing agents with voice capabilities. Get started by installing the [OpenAI Agents SDK for Python](https://openai.github.io/openai-agents-python/voice/quickstart/) with voice support:

```text

pip install openai-agents[voice]

```

See the [Agents SDK voice agents quickstart in GitHub](https://openai.github.io/openai-agents-python/voice/quickstart/) to follow a complete example.

In the example, you'll:

\* Run a speech-to-text model to turn audio into text.

\* Run your code, which is usually an agentic workflow, to produce a result.

\* Run a text-to-speech model to turn the result text back into audio.