OpenAI API for Developers : Dive into OpenAI's API & SDK

Welcome to the comprehensive guide for building intelligent applications using the Open AI Agents SDK. This presentation will walk you through creating, connecting, and deploying AI agents that can work together to solve complex problems.

By- Anubhav Chaudhary

What is OpenAI?

OpenAI is the leading AI research laboratory creating advanced artificial intelligence systems with a mission to ensure AI benefits all of humanity.

Founded in 2015

Started as a non-profit research lab by Sam Altman, Elon Musk, and others to develop safe and beneficial AI

Creator of Powerful AI Models

Develops GPT models (3.5, 4, 4o), DALL·E for image generation, Whisper for speech recognition, and the Agents SDK

API-First Approach

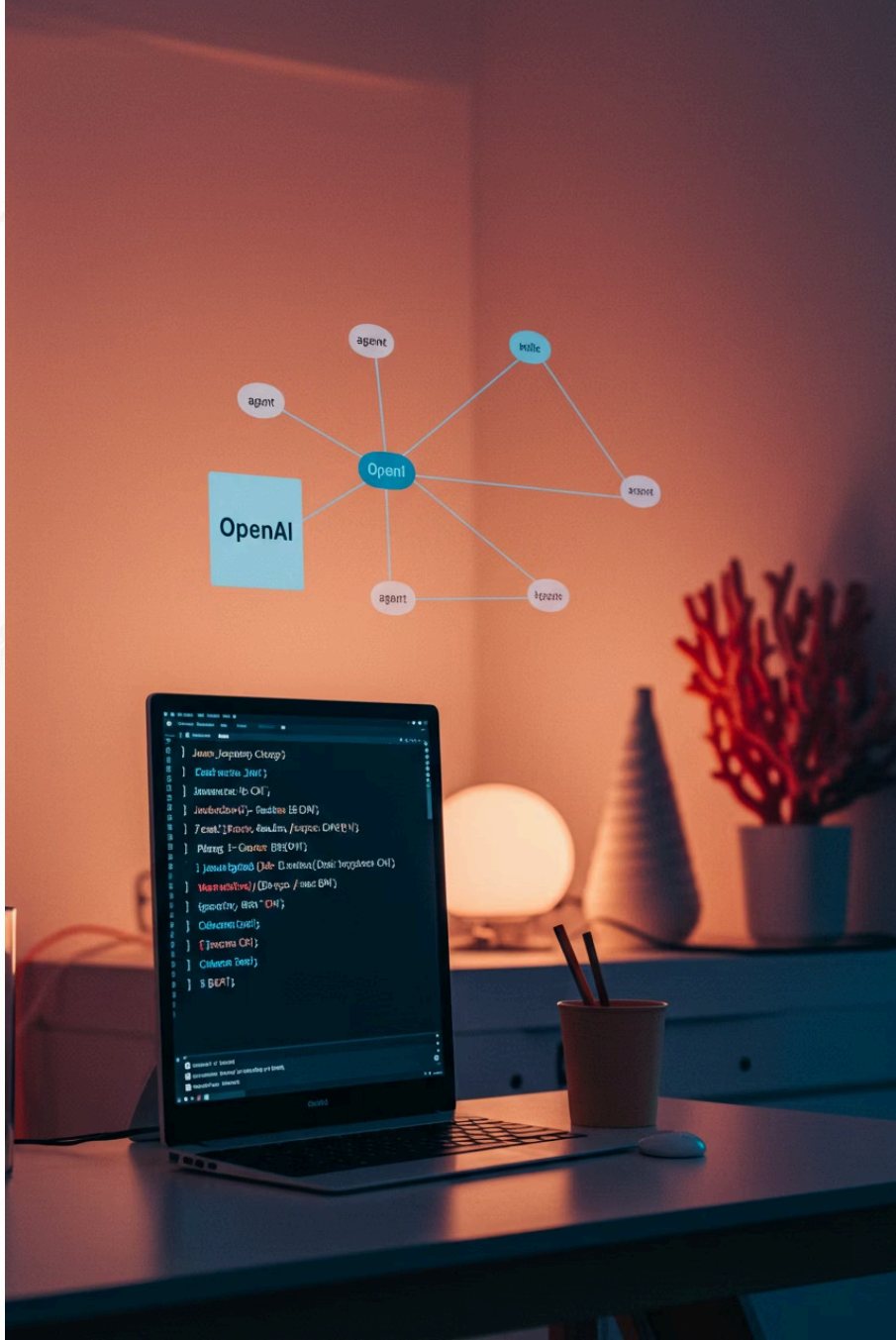
Provides developer-friendly APIs that power thousands of applications across industries

OpenAI's technology powers the AI agent capabilities discussed in this presentation

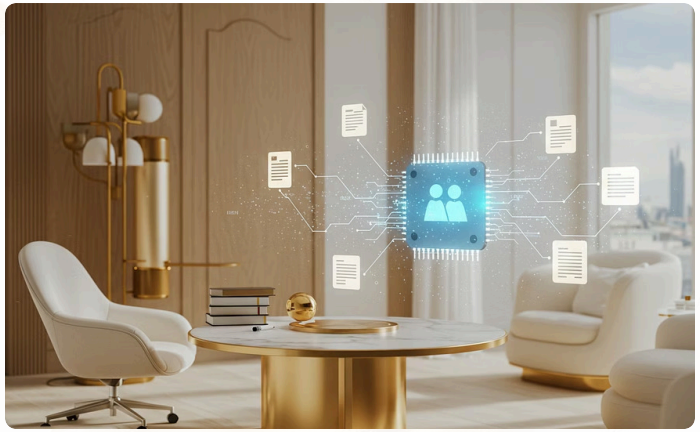
Build Agentic Apps

The OpenAI Agents SDK provides a powerful framework for creating AI-powered applications with minimal code. Whether you're building a customer support system, an educational tool, or a complex workflow automation, this SDK gives you the building blocks to create intelligent, interconnected agents.

In this quickstart guide, we'll walk through building your first agentic application step by step, from basic setup to complex agent orchestration.



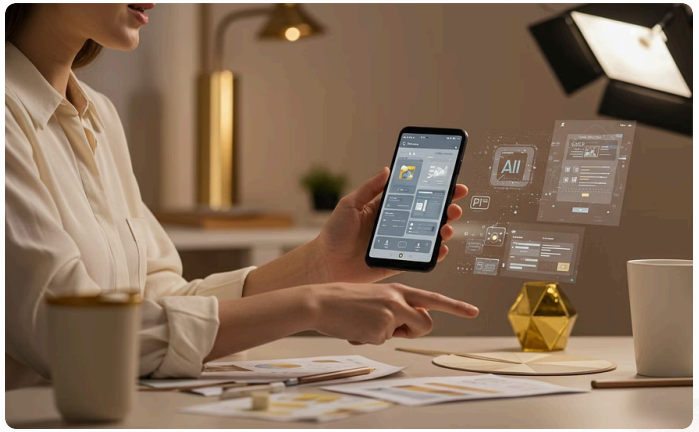
Top Use Cases for AI Agents



RAG (Retrieval-Augmented Generation)

Connect AI agents to knowledge bases, PDFs, and databases via tool functions to provide accurate, contextual responses based on proprietary information.

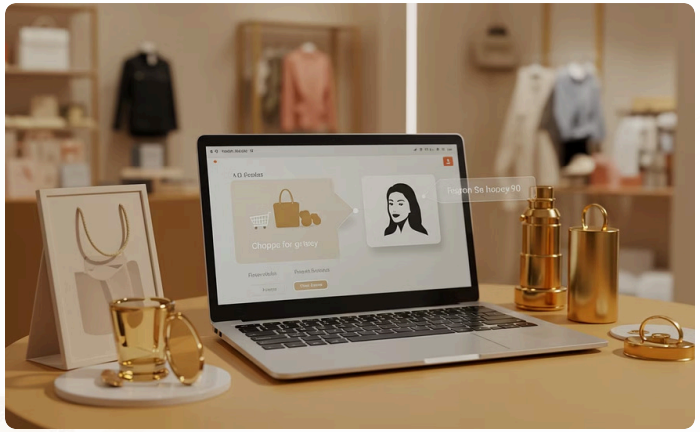
Example: HR Assistant that answers policy questions by referencing internal documentation in real-time.



Multimodal Interfaces

Leverage GPT-4o to create agents that seamlessly understand and generate text, images, and audio for comprehensive interactions.

Example: Design Agent that converts voice descriptions into UI mockups using DALL·E integration.



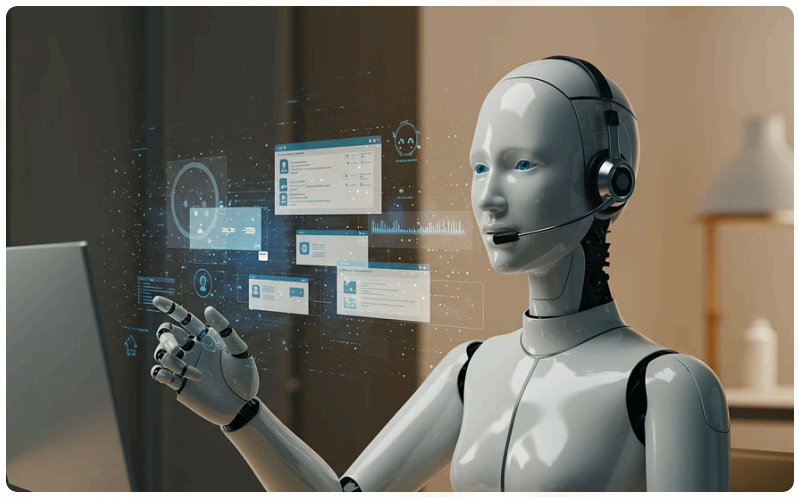
AI Concierge for Websites

Enhance user experience with contextually-aware agents that provide personalized recommendations and support.

Example: E-commerce concierge that suggests relevant products while answering policy questions in the same conversation.

These implementations demonstrate how AI agents can transform business operations by connecting to existing systems, processing multiple input types, and creating more intuitive customer experiences.

Top Use Cases for AI Agents



Customer Support Bots

Auto-handle routine queries, escalate complex issues via handoffs, and pull customer data using specialized tools.

Example: "Order Tracker Bot" that delegates shipping issues to specialized agents while maintaining conversation context.

AI Teaching Assistants

Tutor students across subjects using coordinated agent networks that provide personalized instruction.

Example: Interconnected Math, History, and Science Agents with a central router to guide comprehensive learning.

Workflow Automation

Streamline business processes by updating CRMs, scheduling meetings, and generating reports automatically.

Example: Sales Assistant Agent that logs call summaries, updates Salesforce, and schedules follow-ups.

⚠️ Disadvantages

Paid API (usage-based)

Costs scale with usage, requiring budget planning

Potential hallucinations

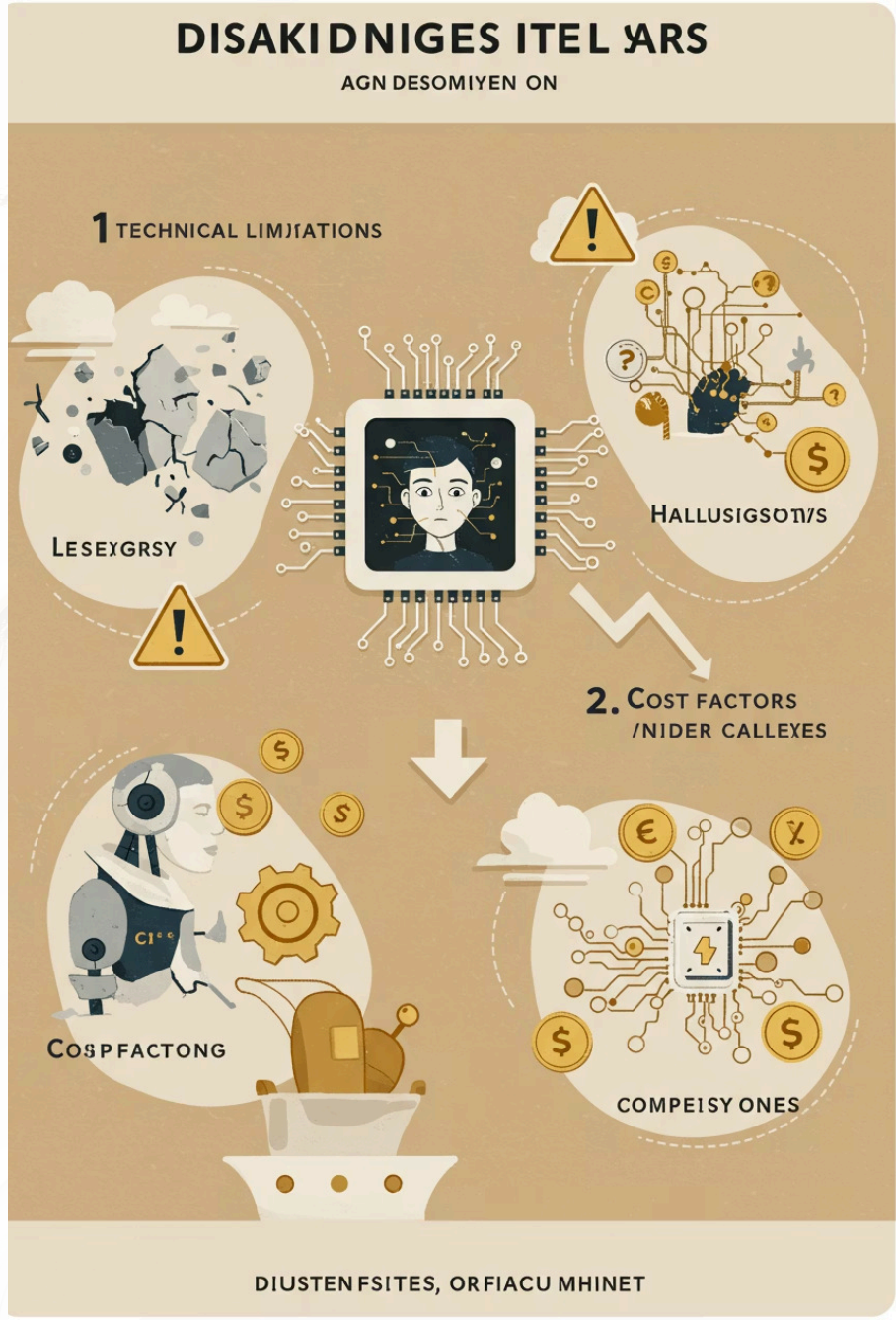
AI may generate incorrect or fabricated information

Requires backend/internet setup

Needs technical infrastructure to operate

May fail in highly technical or context-heavy scenarios

Complex situations can exceed AI capabilities



Challenges

Rate limit handling

Managing API request quotas and preventing service disruptions

API cost optimization

Balancing functionality with budget constraints

Prompt engineering for reliable workflows

Crafting effective instructions for consistent results

Context management across steps

Maintaining relevant information throughout multi-stage processes

Securing outputs to prevent data leaks

Implementing safeguards for sensitive information

Leveraging OpenAI API

Unlock the full potential of OpenAI's advanced models directly in your applications. The OpenAI API allows developers to integrate cutting-edge AI capabilities like text generation, image creation, and speech processing into their products and services.

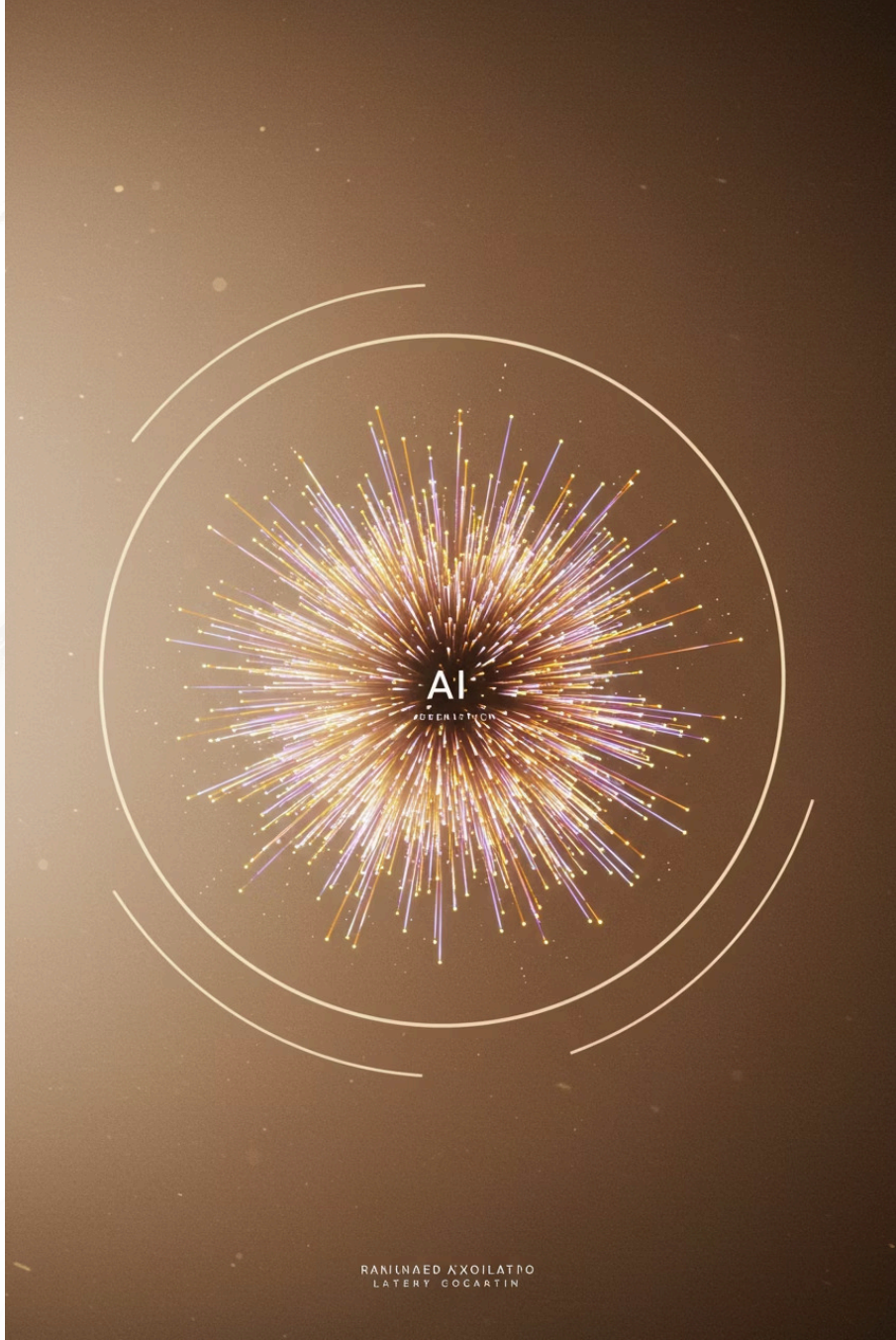
In the following sections, we will explore two key functionalities that enable rich, multimodal AI experiences.

Text-to-Image with DALL·E

Generate stunning visual assets from simple text descriptions, enhancing creative workflows and content generation.

Text-to-Speech with TTS

Convert written text into natural-sounding speech, perfect for audio content, accessibility features, and interactive voice interfaces.



🖼️ 1.0 — Text-to-Image with DALL·E

Turn imagination into visuals using OpenAI's DALL·E API

- Convert natural-language descriptions into high-quality images
- Ideal for prototyping, design, storytelling, creative tasks
- Supports variations, editing, inpainting
- Works with GPT-4o & prompt-chaining

🖼️ 1A — Text-to-Image with DALL·E

Turn imagination into visuals using OpenAI's DALL·E API

Creative Flexibility

Convert natural-language descriptions into photorealistic or artistic images without design expertise

Advanced Capabilities

Create variations of existing images, edit specific portions with inpainting, and extend images beyond original boundaries

Workflow Integration

Chain DALL·E with GPT-4o to automatically generate image prompts from content descriptions

Try DALL·E API | View Examples

API implementation enables seamless integration with your +applications

🖼️ 1B — Use Cases: DALL·E in Action

Design Mockups

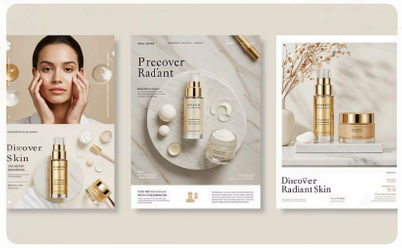
Transform written specifications into polished UI/UX previews, allowing designers to visualize concepts before coding begins.

Children's Books

Create consistent character illustrations and vibrant scenes that match narratives, reducing illustration costs and production time.

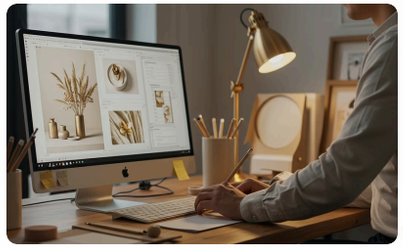
E-commerce

Generate product visualizations for concepts still in development, enabling market testing before manufacturing investment.



Marketing Assets

Convert campaign concepts into cohesive visual assets across multiple formats while maintaining brand consistency.



AI + Human Collaboration

Rapidly iterate through design variations using refined prompts, with humans directing the creative process and making final selections.

Try DALL·E API | View Implementation Examples

🖼️ 1C — Quick-Start Code: Generate an Image

```
// 1. npm install openai  &&  export OPENAI_API_KEY=sk-...
import OpenAI from "openai";
const openai = new OpenAI();

const prompt = "A cozy cabin in the snow, sunset, digital painting";
const imgRes = await openai.images.generate({
  model: "dall-e-3",
  prompt,
  n: 1,
  size: "1024x1024"
});
console.log(imgRes.data[0].url); // → Share / download the image
```

Just 3 lines to integrate DALL·E into your application. Modify the prompt, count (n), or dimensions to iterate quickly.

Customizable

Adjust model, size, quality, and style parameters

Fast

Generate high-quality images in seconds

Try DALL·E API | View Documentation

🔊 2A — Text-to-Speech (TTS) with OpenAI API

Give voice to your apps with natural-sounding audio

Convert Any Text

Transform written content into lifelike speech with just a few lines of code, supporting multiple languages and long-form content.

Multiple Voices

Choose from diverse voice options with different tones, emotions, and accents to match your brand or use case requirements.

Low Latency

Deliver audio in real-time with fast streaming capabilities, enabling responsive conversational interfaces and dynamic content.

Ideal for creating accessible applications, virtual assistants, audiobooks, educational content, and immersive storytelling experiences without recording studios.

Try TTS API | View Examples

2B — Use Cases: TTS in Action

AI Tutors

Transform educational content into engaging spoken lessons across multiple languages, making learning more accessible and accommodating different learning styles.

Voice Assistants

Enhance conversational AI with natural-sounding voices that reflect brand identity and create more human-like interactions without recording studios.

Mobile Apps

Convert text content like news articles, emails, and navigation instructions into audio, enabling hands-free consumption and improving accessibility.

Games

Generate dynamic character dialogue on the fly, allowing for contextual speech without pre-recording thousands of voice lines for every possible scenario.

Language Learning

Provide instant pronunciation examples and feedback in multiple languages, helping learners develop proper speaking skills through accurate audio models.

Try TTS API | View Code Examples

🖼️ 2C — Quick-Start Code: Generate an Audio

```
// 1. npm install openai fs  &&  export OPENAI_API_KEY=sk-...
import fs from "node:fs/promises";
import OpenAI from "openai";
const openai = new OpenAI();

const ttsRes = await openai.audio.speech.create({
  model: "gpt-4o-tts",
  input: "Welcome to our AI-powered app!",
  voice: "alloy",        // try: "echo", "nova", etc.
  format: "mp3"
});

// Save to file
await fs.writeFile("welcome.mp3", Buffer.from(await ttsRes.arrayBuffer()));
console.log("console.log("Audio saved → welcome.mp3");
```

Swap `voice`, `input`, or `format` (`wav`, `ogg`) as needed; stream for real-time playback.

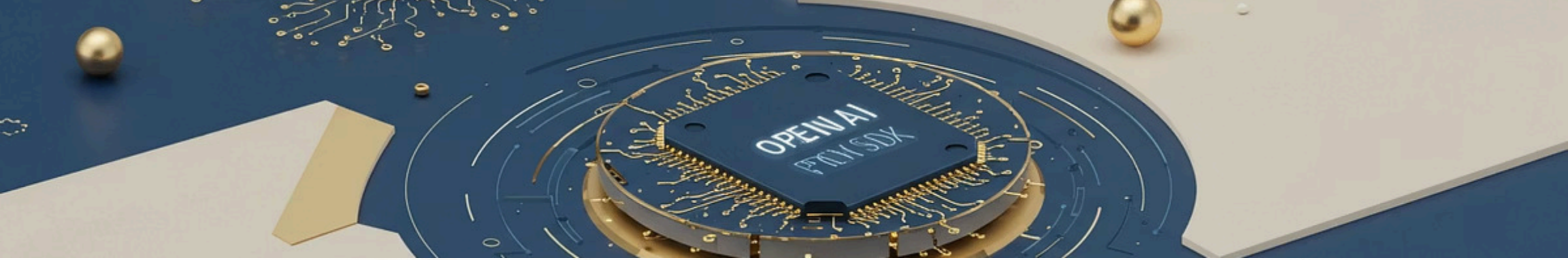
Try DALL·E API | View Documentation

Customizable

Adjust model, size, quality, and style parameters

Fast

Generate high-quality Voice in seconds



Why Agents SDK?

Agents

AI workers with specific instructions and capabilities that can perform tasks and make decisions

Tools

Typed, validated external functions that extend agent capabilities and connect to external systems

Handoffs

Delegation system allowing agents to transfer tasks to specialized agents for optimal processing

Guardrails

Schema validation ensuring safe inputs and outputs, preventing unexpected behaviors

Tracing

Comprehensive debugging tools to inspect every step of the agent workflow

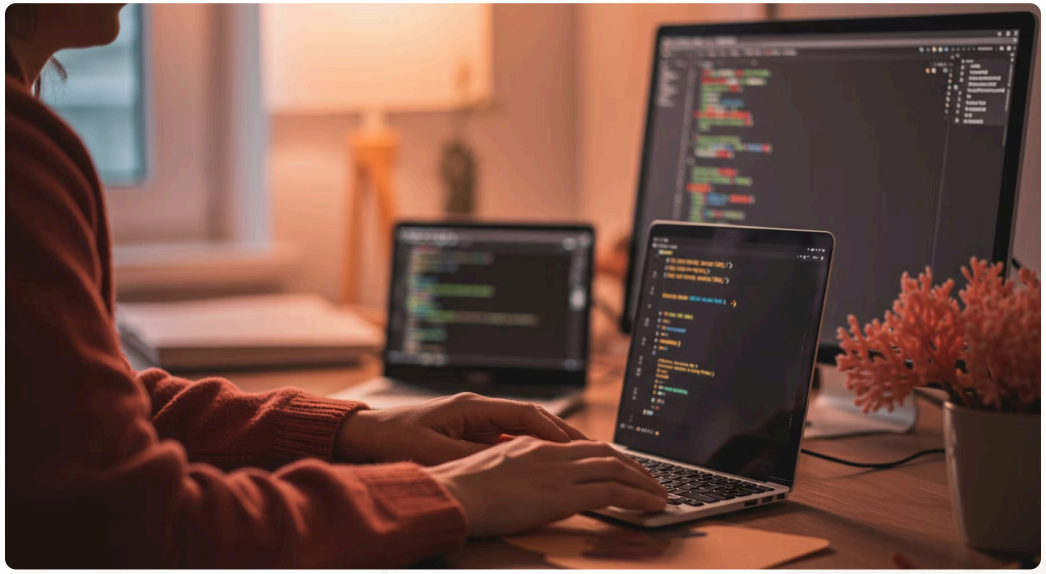
Step 1: Project Setup

Setting Up Your Environment

Before building agents, you'll need to set up a new project with the necessary dependencies and configure your OpenAI API key.

Follow these commands to create a new project directory, initialize a package.json file, and install the required packages:

```
mkdir my_project && cd my_project
npm init -y
npm install @openai/agents zod
export OPENAI_API_KEY="sk-•••"
```



The @openai/agents package provides the core SDK functionality, while zod is used for creating schema validation for inputs and outputs.

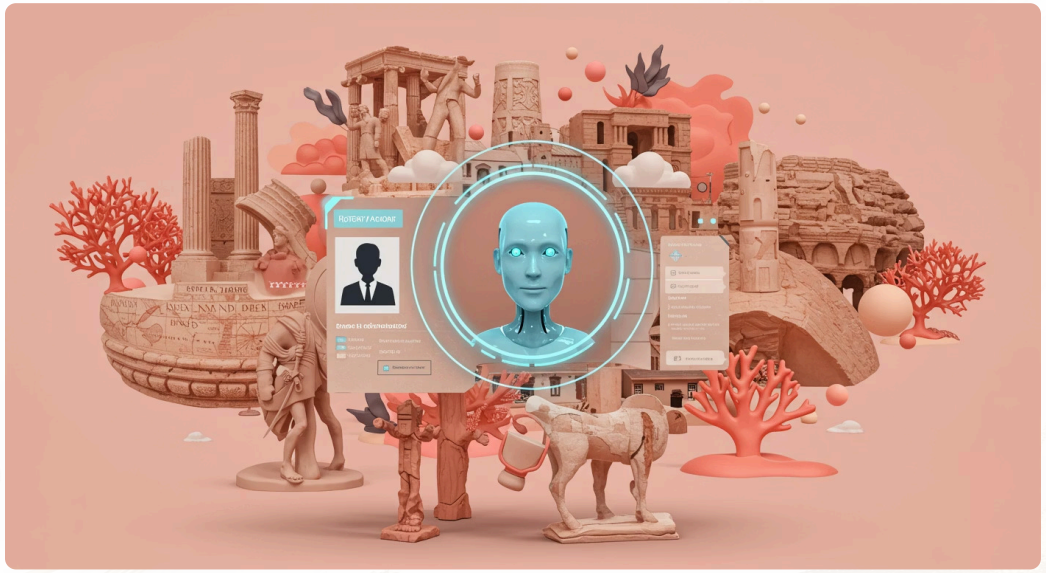
Step 2: Create Your First Agent

Defining Agent Behavior

Agents are the core building blocks of the SDK. Each agent has a name and a set of instructions that define its behavior and capabilities.

```
import { Agent } from "@openai/agents";

const historyTutor = new Agent({
  name: "History Tutor",
  instructions: "Answer historical questions with clear context.",
});
```

This creates a History Tutor agent that can answer questions about historical events with appropriate context and accuracy.



The instructions parameter is crucial as it guides the agent's behavior. More detailed instructions result in more specific and useful responses.

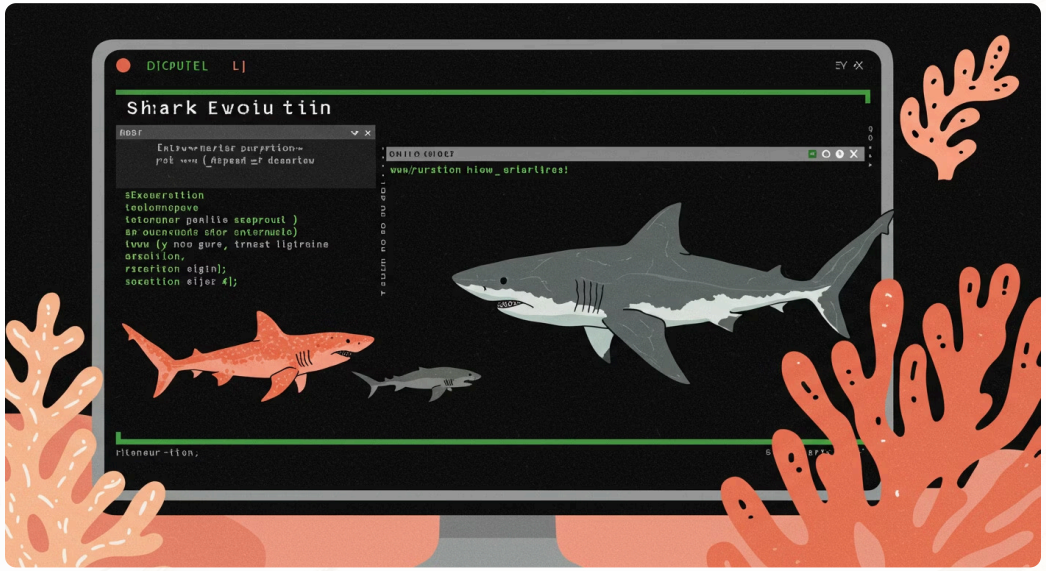
Step 3: Run the Agent

Invoking Your Agent

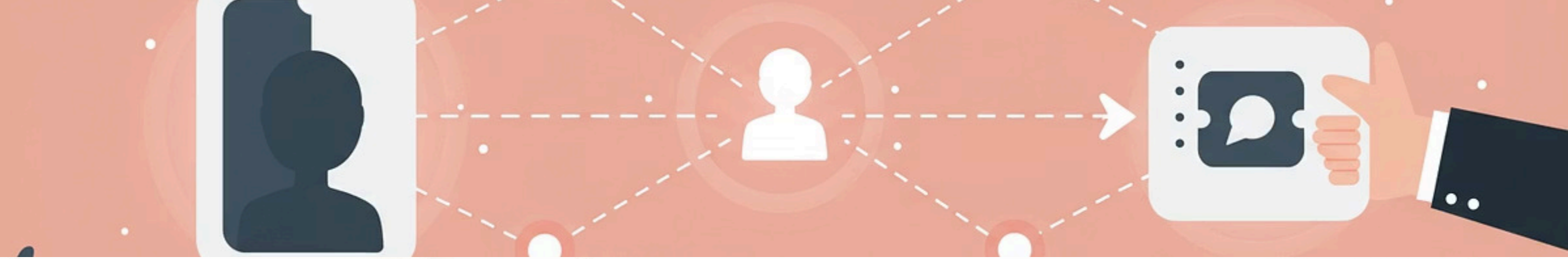
Once you've defined an agent, you can run it using the run() function from the SDK. This function takes the agent and a user query as inputs.

```
import { run } from "@openai/agents";

const res = await run(historyTutor,
  "When did sharks first appear?");

console.log(res.finalOutput);
```

The run() function handles the interaction with OpenAI's API and returns the agent's response, which you can access through the finalOutput property.



For this query, the History Tutor agent would respond with information about sharks first appearing approximately 450 million years ago, predating dinosaurs and even trees.



Step 4: Add a Simple Tool

Define Tool Function

Create a tool with a name, description, and execution logic

```
const funFact = tool({
  name: "history_fun_fact",
  description: "Short fun fact",
  execute: async () => "Sharks are older than trees.",
});
```

Attach to Agent

Add the tool to the agent's toolset

```
historyTutor.tools = [funFact];
```

Agent Uses Tool

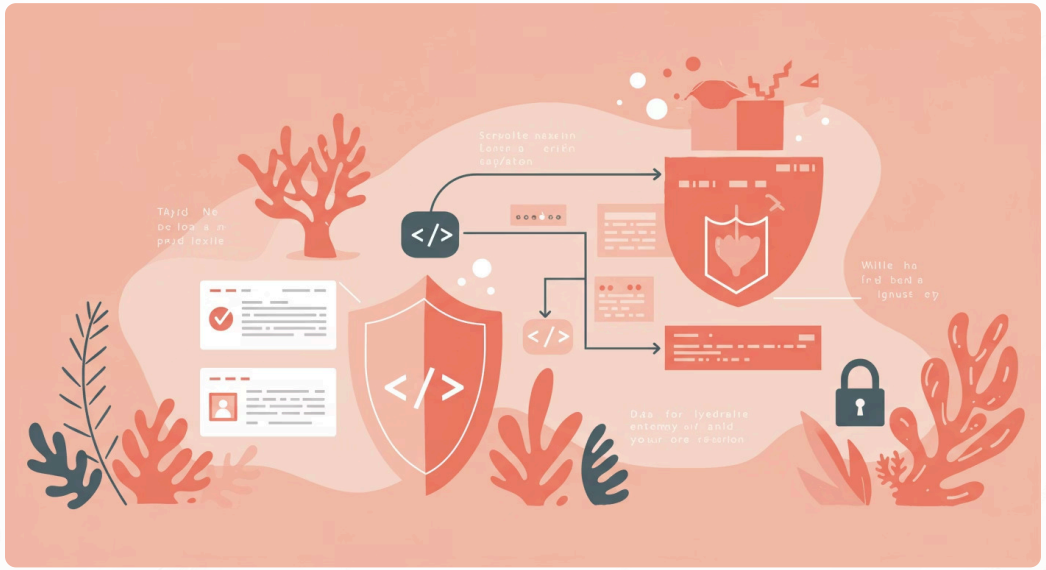
The agent automatically determines when to use the tool based on the query context

Step 5: Add Guardrails for Safety

Validating Inputs and Outputs

Zod schemas provide type safety and validation for tool inputs and outputs, preventing unexpected behaviors and ensuring data integrity.

```
import { z } from "zod";

const validatedFact = tool({
  name: "validated_fun_fact",
  inputSchema: z.object({
    topic: z.string().min(3).max(50),
  }),
  outputSchema: z.string().max(100),
  execute: async ({ topic }) =>
    "Great Wall is 13,000+ miles.",
});
```



The inputSchema validates that the topic parameter is a string between 3 and 50 characters, while the outputSchema ensures the response doesn't exceed 100 characters.

These guardrails help prevent malformed inputs and ensure consistent, safe outputs from your tools.

Step 6: Add More Agents

Expanding Your Agent Ecosystem

You can create multiple specialized agents to handle different types of tasks or domains of knowledge. This modular approach allows for better organization and more focused agent behaviors.

```
const mathTutor = new Agent({
  name: "Math Tutor",
  instructions: "Explain math step-by-step with examples.",
});
```

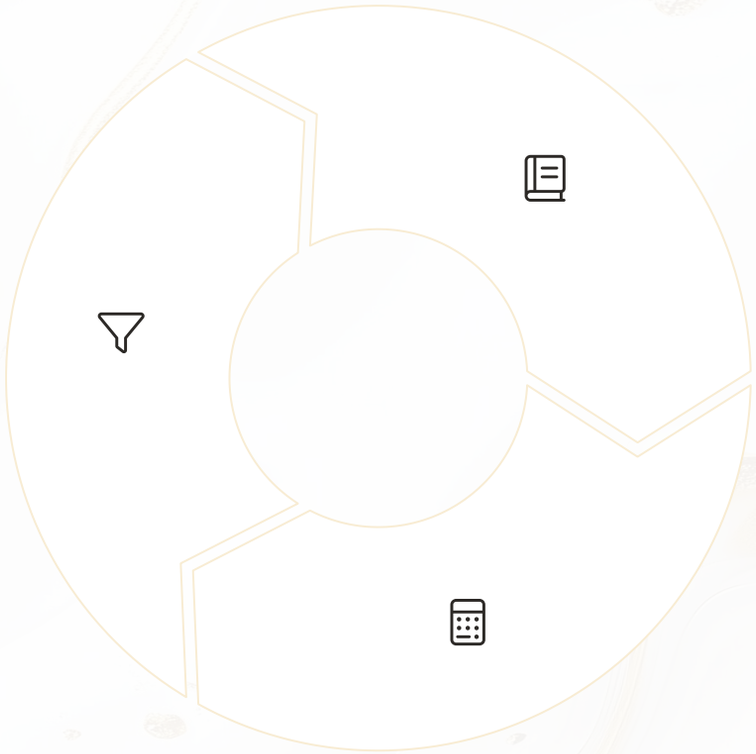
This Math Tutor agent is designed to handle mathematical questions with clear step-by-step explanations and relevant examples to illustrate concepts.

Creating specialized agents allows each one to excel in a specific domain rather than trying to build a single agent that handles everything. This improves response quality and makes your system more maintainable.

Step 7: Define Agent Handoffs

Triage Agent

Routes questions to specialized agents based on content and intent analysis



History Tutor

Handles historical questions with contextual information

Math Tutor

Provides step-by-step mathematical explanations

```
const triage = new Agent({
  name: "Triage Agent",
  instructions: "Route questions to the right tutor.",
  handoffs: [historyTutor, mathTutor],
});
```

The triage agent acts as a front door for user queries, analyzing the content and directing it to the most appropriate specialized agent. This creates a more efficient and accurate system overall.

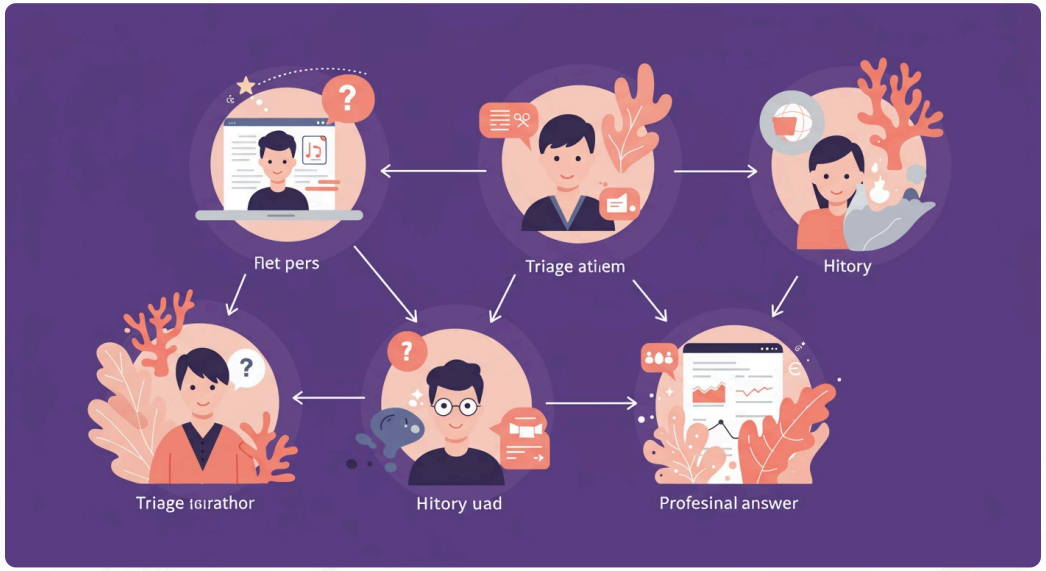
Step 8: Run the Agent Orchestration

Executing the Agent Network

Running an orchestrated agent system is as simple as running a single agent. The SDK handles all the delegation logic behind the scenes.

```
const answer = await run(
  triage,
  "What is the capital of France?"
);

console.log(answer.finalOutput); // Paris
```

When this code runs, the triage agent will analyze the question, determine it's a history question, and automatically delegate to the history tutor agent, which will then provide the answer.



The SDK's orchestration system intelligently manages the handoffs between agents, ensuring that each query is handled by the most appropriate agent without requiring explicit routing logic in your code.

Step 9: View Your Project

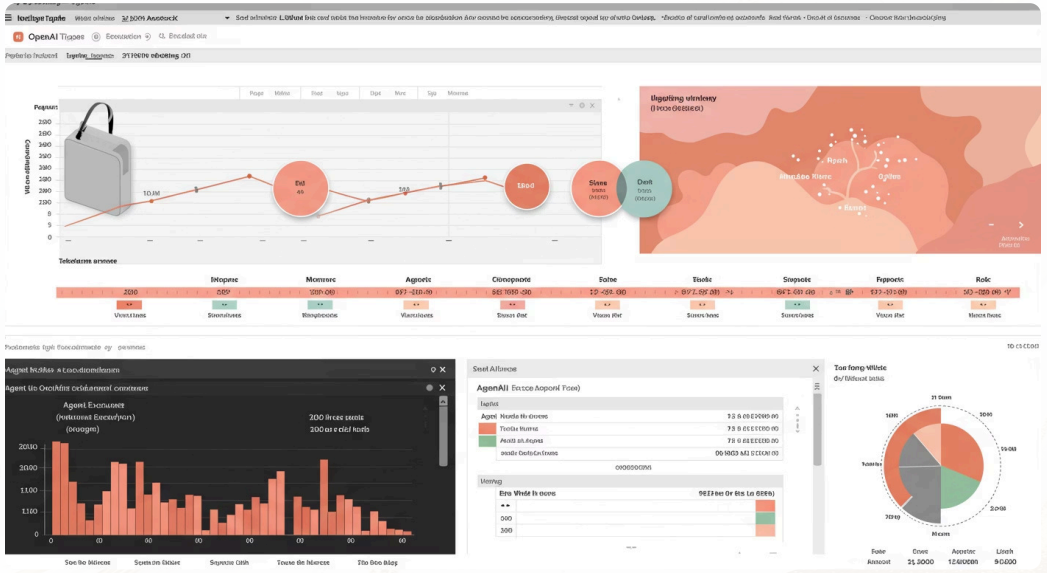
Debugging and Monitoring

The OpenAI Dashboard provides comprehensive tracing capabilities for your agent executions, allowing you to monitor and debug your applications.

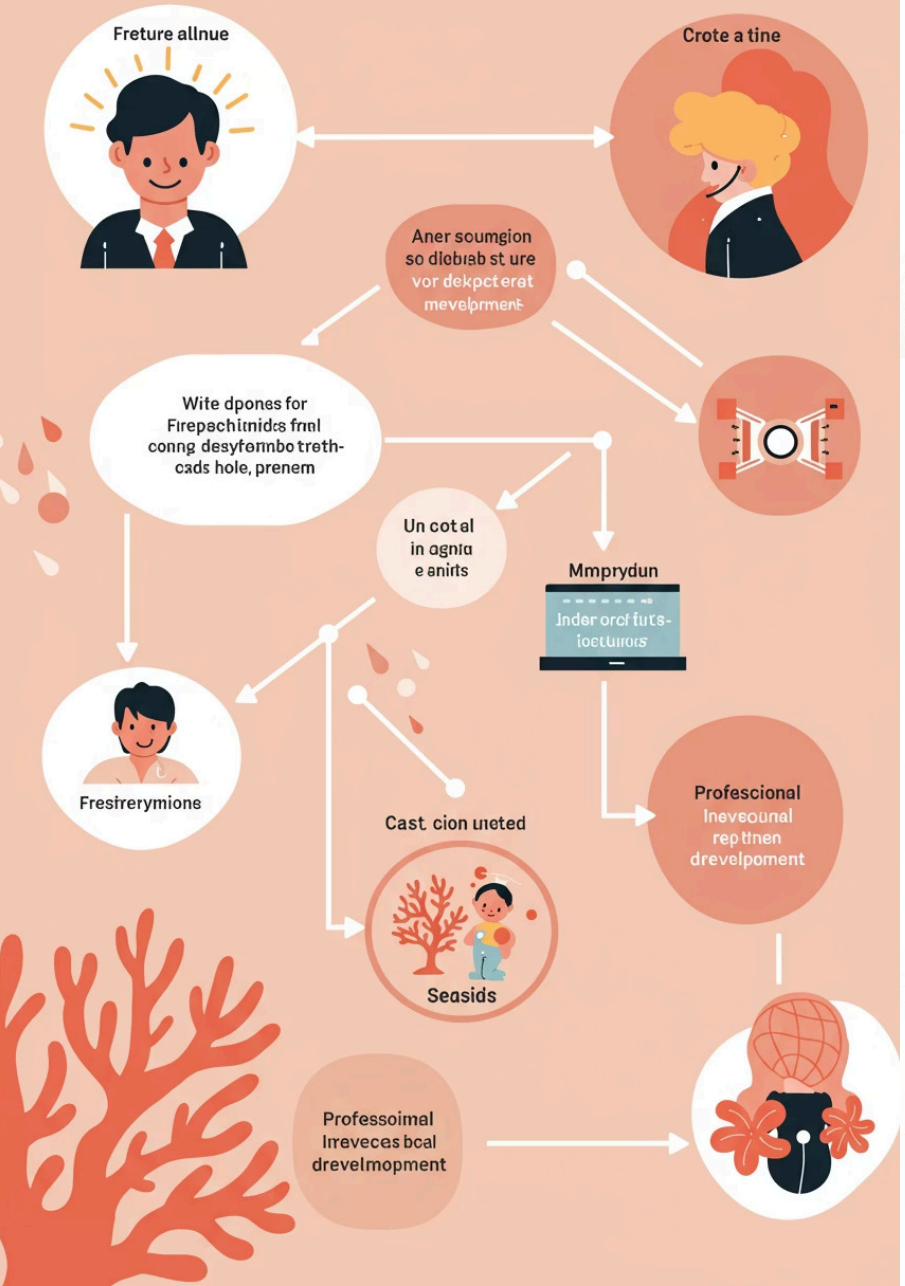
Visit https://platform.openai.com/account/usageh to access the tracing dashboard.

The trace view shows:

- Agent handoffs and delegations
- Tool executions and their inputs/outputs
- Token usage for cost optimization
- Detailed execution paths



Tracing is crucial for understanding how your agents are making decisions, which tools they're using, and identifying potential bottlenecks or issues in your agent workflows.



Step 10: Next Steps

Advanced Features

- Implement streaming responses for real-time interactions
- Connect agents to external APIs and data sources
- Add database integration for persistent agent memory

Monitoring & Observability

- Export traces to OpenTelemetry
- Integrate with monitoring tools like Datadog
- Set up alerting for error conditions

Community Engagement

- Join the OpenAI Developer Forum
- Share your projects and get feedback
- Learn from other developers' implementations

Start Building Agentic Apps Today

The Power of Simplicity

With the OpenAI Agents SDK, you can create sophisticated AI workflows with minimal code. The power comes from the declarative approach to defining agents and their capabilities.

Key takeaways:

1 Unlock Potent Workflows

Empower powerful and intricate processes with minimal code.

2 Craft Intelligent Solutions

Create smart bots, assistants, and automations.

3 Control at Your Fingertips

Manage logic, tools, delegation, and boundaries with ease.

4 Robust Debugging and Monitoring

Monitor and troubleshoot with extensive capabilities.

Happy Coding! 🧠💻

The future of software development includes intelligent agents working together to solve complex problems. Start building your agentic applications today and be at the forefront of this exciting technology.

🔚 Conclusion – Why OpenAI Agents SDK?

Build more than just prompts.

Build agentic systems that think, delegate, and act.

✅ **Agents — LLMs with clear roles**

✅ **Tools — Typed, trusted function calls**

✅ **Guardrails — Schema validation = safer AI**

✅ **Handoffs — Agent-to-agent collaboration**

✅ **Tracing — Full transparency & debuggability**

🚀 From chatbots to automation — this SDK is your AI backend.

🔜 What's Next?

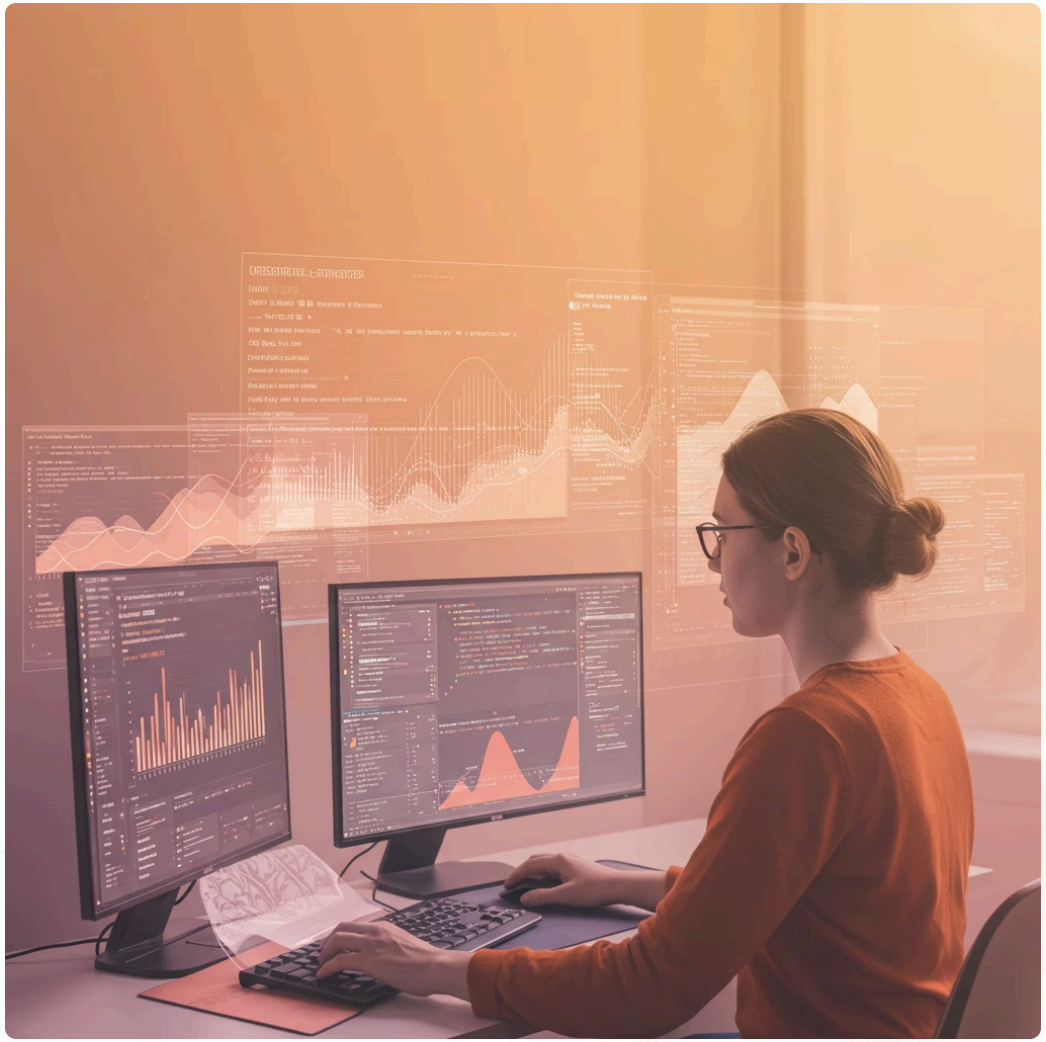
💡 **Start simple: one agent + one tool**

🔧 **Experiment: build multi-agent flows**

📈 **Monitor: explore the Trace Viewer**

🌐 Explore more:

- Streaming agents
- Real-time workflows
- External APIs and tools
- Custom exporters (OpenTelemetry, Datadog)



Additional References

- [Featured Cookbooks for Developers by OpenAI]
- [Generate Images with High Input Fidelity]
- [App Assistant Voice Agents]
- [Optimize Prompts]
- [GPT-4 Prompting Guide]

Thank You

I appreciate your attendance at this OpenAI session.

Continue your journey with OpenAI Agents SDK by exploring the official documentation.

Session By :- Anubhav Chaudhary