

# OpenAI Agent Tools

# Agenda

- New tools
  - Web Search
  - Computer Use
  - File Search
- Agents SDK
- Observability

Tools

# Web Search

## “LLM + Google”

- Similar to Perplexity Sonar or LLM + serper tool
- Very expensive
- NB: This tool does not support zero data retention or data residency

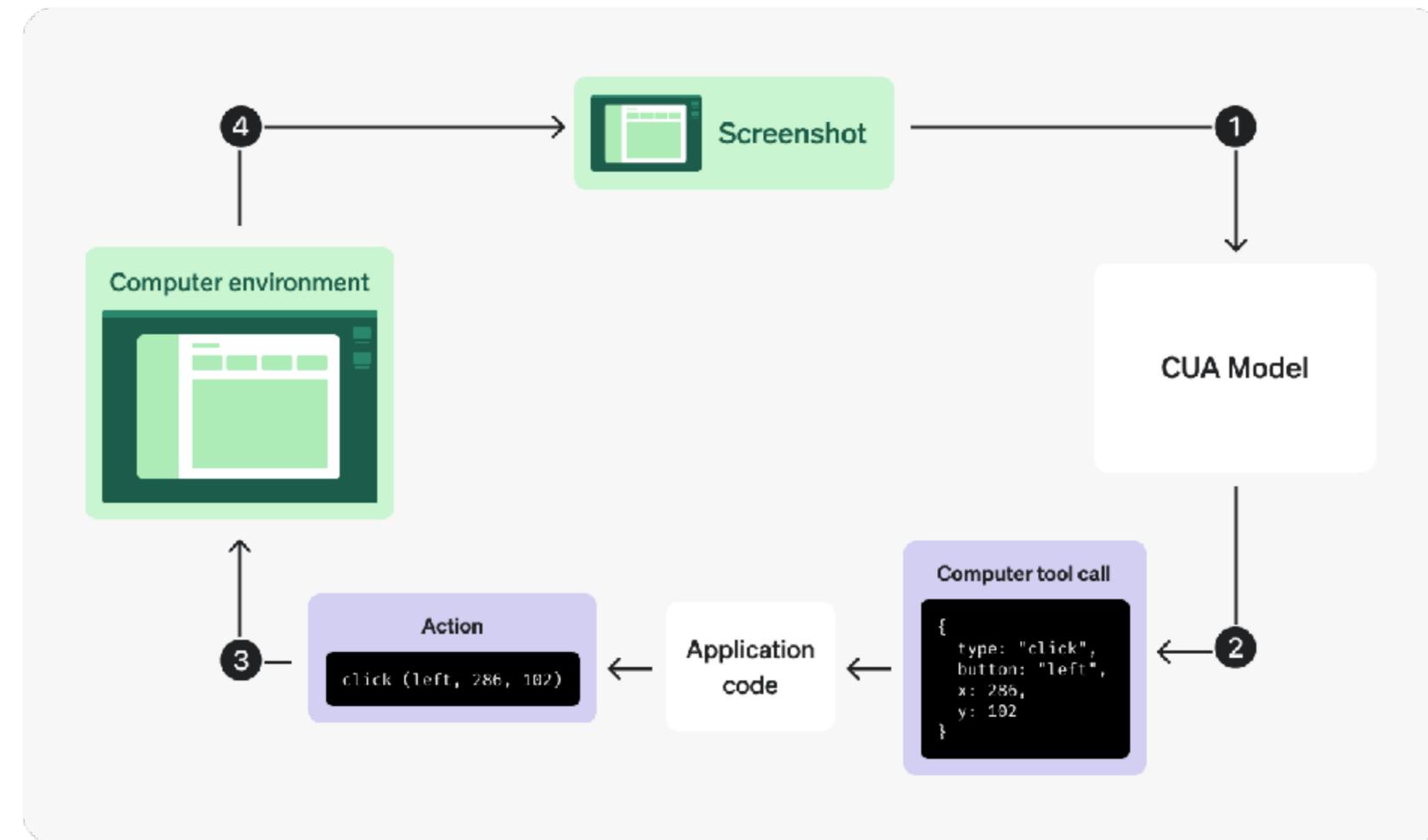
```
response = client.responses.create(
    model="gpt-4o",
    tools=[{"type": "web_search_preview"}],
    input="Last week news in Johor Bahru"
)

print(response.output_text)
# Print URL citations from the response
for output in response.output:
    if hasattr(output, 'content'):
        for content_item in output.content:
            if hasattr(content_item, 'annotations'):
                for annotation in content_item.annotations:
                    if annotation.type == 'url_citation':
                        print(f"Citation: {annotation.title}")
                        print(f"URL: {annotation.url}")
                        print("-" * 50)
```

# Computer Tool

## Operator as API

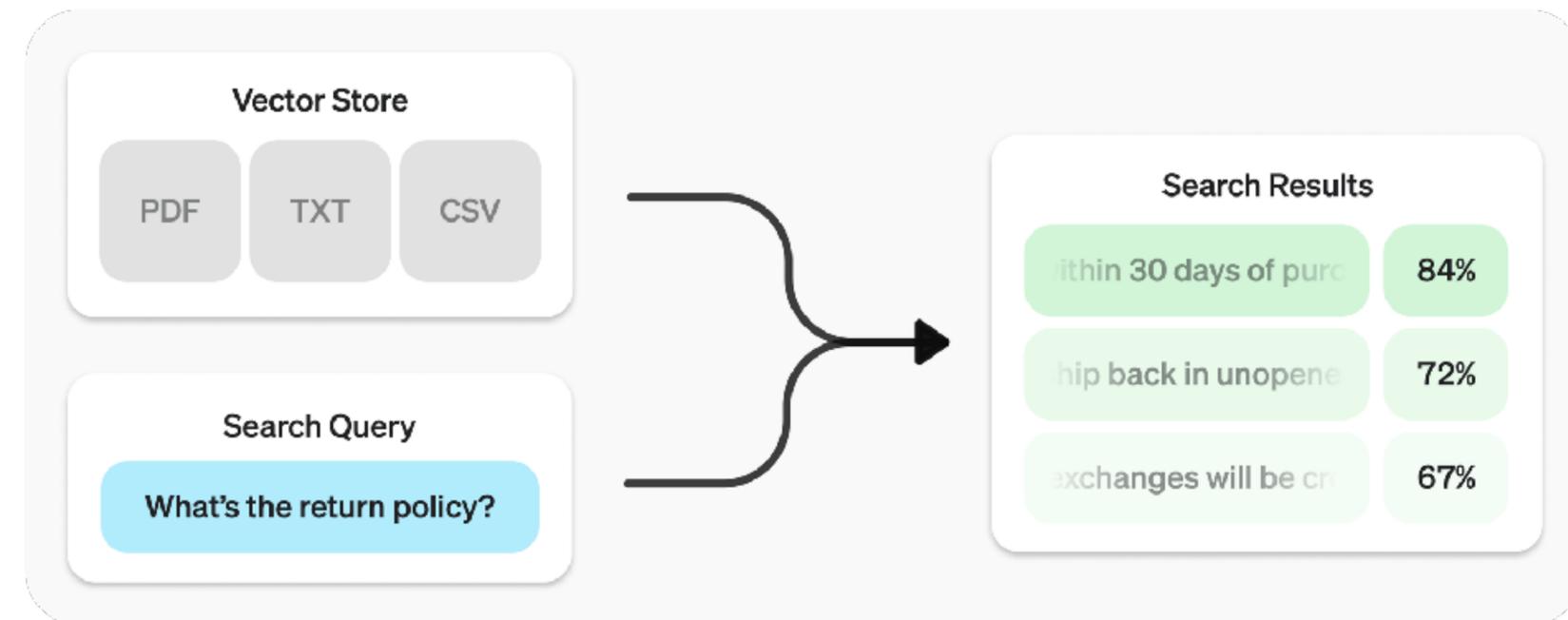
- Tech that powers OpenAI Operator
- Similar to browser-use
- You can use local browser or VM
- Sample app: <https://github.com/openai/openai-cua-sample-app>
- Recommended for browsers (38.1% on OSWorld)
- No zero data retention support



# File Search

## RAG as API

- Similar to Assistants API
- Combine text search with semantic search
- Has query-rewrite setting
- Supports attribute filtering & reranking
- Kinda expensive (\$0.1 per GB **per day**)



# Agents SDK

- Reminds CrewAI
- Liked “agent as tool” feature
- Vendor lock-in attempt?
- Make sure to use o-models (o3-mini works great)

```
> icp == "PHD" Aa .:,* No results
icp_agent = Agent(
    name="Ideal customer profile agent",
    instructions="Your job is to parse the given company's website and generate ideal customer profile description",
    tools=[WebSearchTool()]
)

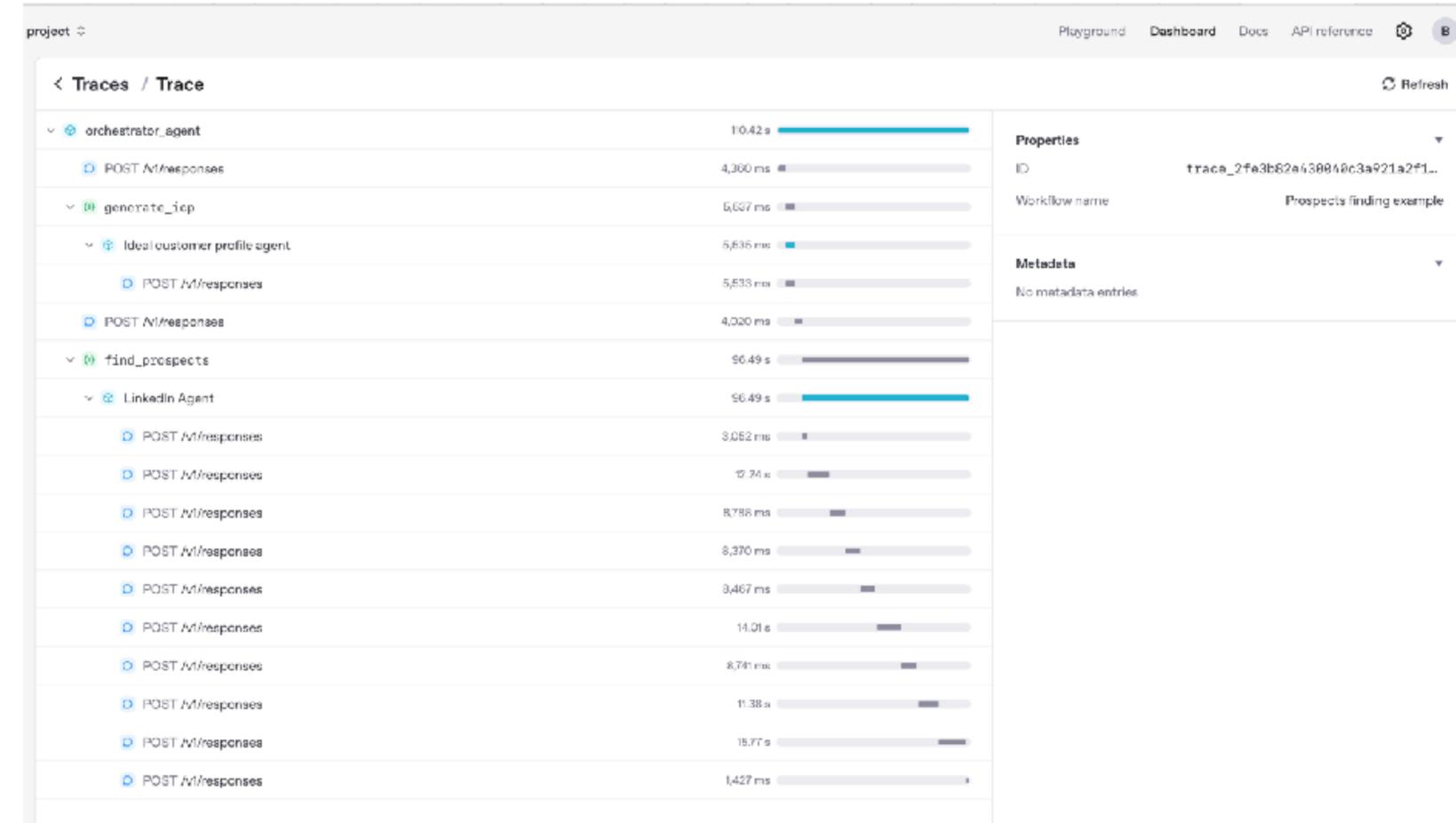
async def setup_linkedin_agent():
    # Create a new computer instance for each agent call
    computer = await LocalPlaywrightComputer().__aenter__()
    agent = Agent(
        name="LinkedIn Agent",
        instructions="Your job is to find people on LinkedIn that fit the Ideal Customer Profile",
        tools=[ComputerTool(computer)],
        # Use the computer using model, and set truncation to auto because its required
        model="computer-use-preview",
        model_settings=ModelSettings(truncation="auto"),
    )
    return agent, computer

linkedin_agent = None # Will be initialized in main()
linkedin_computer = None # Store the computer instance

orchestrator_agent = Agent(
    name="orchestrator agent",
    model="o3-mini",
    instructions="You are a Prospects Finding Agent. You use the tools given to you to generate company's ideal customer profile and find relevant prospects",
    tools=[
        icp_agent.as_tool(
            tool_name="generate_icp",
            tool_description="Generate ideal customer profile for a given company website"
        ),
        # linkedin_agent will be added in main()
    ]
)
```

# Observability

- Reminds Langfuse, but less powerful
- Works out-of-the-box, does not require any configuration



# Final Thoughts

- Strong focus on AI Agents
- Attempt to lock developers in
- Expensive (for now)

Q&A