Quickstart

This quickstart guides you through installing the Agent Development Kit (ADK), setting up a basic agent with multiple tools, and running it locally either in the terminal or in the interactive, browser-based dev UI.

This quickstart assumes a local IDE (VS Code, PyCharm, etc.) with Python 3.9+ and terminal access. This method runs the application entirely on your machine and is recommended for internal development.

1. Setup Environment & Install ADK

Create & Activate Virtual Environment (Recommended):

```
# Create
python -m venv .venv
# Activate (each new terminal)
# macOS/Linux: source .venv/bin/activate
# Windows CMD: .venv\Scripts\activate.bat
# Windows PowerShell: .venv\Scripts\Activate.ps1
```

Install ADK:

```
pip install google-adk
```

2. Create Agent Project

Project structure

You will need to create the following project structure:

```
parent_folder/
    multi_tool_agent/
        __init__.py
        agent.py
        .env
```

Create the folder `multi_tool_agent`:

```
mkdir multi_tool_agent/
```

`__init__.py`

Now create an `__init__.py` file in the folder:

```
echo "from . import agent" > multi_tool_agent/__init__.py
```

Your `__init__.py` should now look like this:

multi_tool_agent/__init__.py

```
from . import agent
```

`agent.py`

Create an `agent.py` file in the same folder:

```
touch multi_tool_agent/agent.py
```

Copy and paste the following code into `agent.py`:

multi_tool_agent/agent.py

```python
import datetime
from zoneinfo import ZoneInfo
from google.adk.agents import Agent

def get_weather(city: str) -> dict:
    """Retrieves the current weather report for a specified city.

    Args:
        city (str): The name of the city for which to retrieve the weather report.

    Returns:
        dict: status and result or error msg.
    """
    if city.lower() == "new york":
        return {
            "status": "success",
            "report": (
                "The weather in New York is sunny with a temperature of 25 degrees"
                " Celsius (41 degrees Fahrenheit)."
            ),
        }
    else:
        return {
            "status": "error",
            "error_message": f"Weather information for '{city}' is not available.",
        }


def get_current_time(city: str) -> dict:
```

```
    """Returns the current time in a specified city.

    Args:
        city (str): The name of the city for which to retrieve the current time.

    Returns:
        dict: status and result or error msg.
    """

    if city.lower() == "new york":
        tz_identifier = "America/New_York"
    else:
        return {
            "status": "error",
            "error_message": (
                f"Sorry, I don't have timezone information for {city}."
            ),
        }

    tz = ZoneInfo(tz_identifier)
    now = datetime.datetime.now(tz)
    report = (
        f'The current time in {city} is {now.strftime("%Y-%m-%d %H:%M:%S %Z%z")}'
    )
    return {"status": "success", "report": report}


root_agent = Agent(
    name="weather_time_agent",
    model="gemini-2.0-flash-exp",
    description=(
        "Agent to answer questions about the time and weather in a city."
    ),
    instruction=(
        "I can answer your questions about the time and weather in a city."
    ),
    tools=[get_weather, get_current_time],
```

```
)
```

`.env`

Create a `.env` file in the same folder:

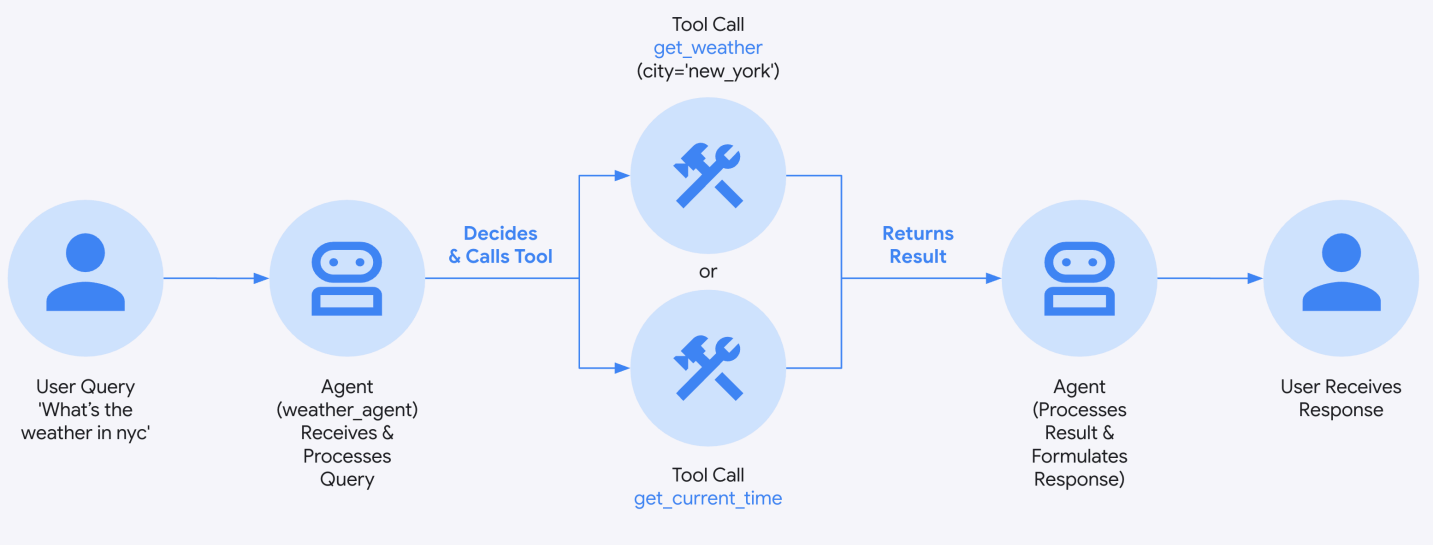
```
touch multi_tool_agent/.env
```

You can just copy and paste the following code for now, as more instructions are describe in the next section on Setup the model.

multi_tool_agent/.env

```
# If using Gemini via Google AI Studio
GOOGLE_GENAI_USE_VERTEXAI="False"
GOOGLE_API_KEY="paste-your-actual-key-here"

# # If using Gemini via Vertex AI on Google CLoud
# GOOGLE_CLOUD_PROJECT="your-project-id"
# GOOGLE_CLOUD_LOCATION="your-location" #e.g. us-central1
# GOOGLE_GENAI_USE_VERTEXAI="True"
```



3. Setup the model

Your agent's ability to understand user requests and generate responses is powered by a Large Language Model (LLM). Your agent needs to make secure calls to this external LLM service, which requires authentication credentials. Without valid authentication, the LLM service will deny the agent's requests, and the agent will be unable to function.

Gemini - Google AI Studio

1. Get an API key from Google AI Studio.
2. Open the `.env` file located inside (`multi_tool_agent/`) and copy-paste the following code.

multi_tool_agent/.env

```
GOOGLE_GENAI_USE_VERTEXAI=FALSE
GOOGLE_API_KEY=PASTE_YOUR_ACTUAL_API_KEY_HERE
```

3. Replace `GOOGLE_API_KEY` with your actual `API KEY`.

Gemini - Google Cloud Vertex AI

1. You need an existing Google Cloud account and a project.
   - Set up a Google Cloud project
   - Set up the gcloud CLI
   - Authenticate to Google Cloud, from the terminal by running `gcloud auth login`.
   - Enable the Vertex AI API.
2. Open the `.env` file located inside (`multi_tool_agent/`). Copy-paste the following code and update the project ID and location.

multi_tool_agent/.env

```
GOOGLE_GENAI_USE_VERTEXAI=TRUE
GOOGLE_CLOUD_PROJECT=YOUR_PROJECT_ID
GOOGLE_CLOUD_LOCATION=LOCATION
```

4. Run Your Agent

Using the terminal, navigate to the parent directory of your agent project (e.g. using `cd ..`):

```
parent_folder/      <-- navigate to this directory
```

```
    multi_tool_agent/
        __init__.py
        agent.py
        .env
```

There are multiple ways to interact with your agent:

Dev UI (adk web)

Run the following command to launch the **dev UI**.

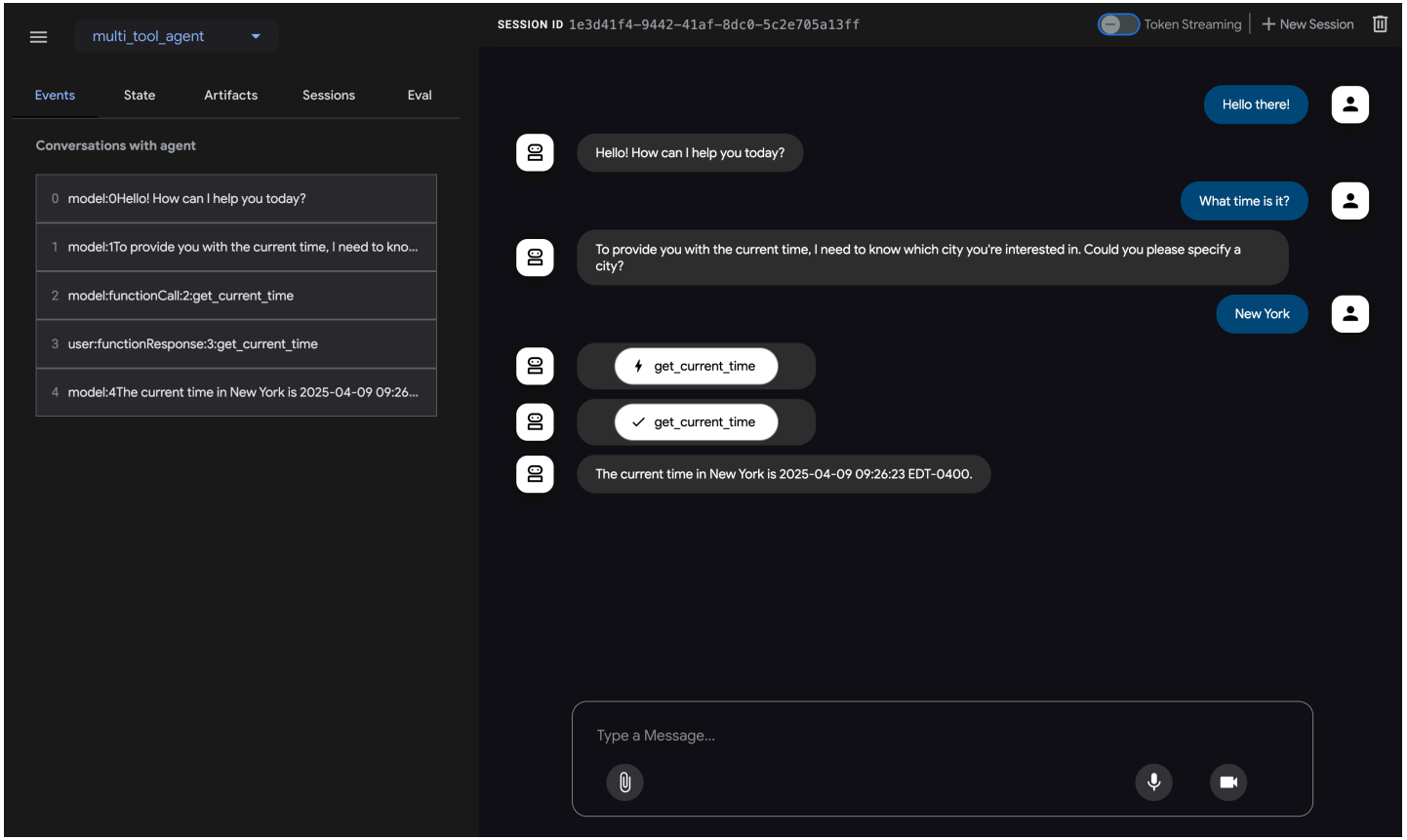
```
adk web
```

Step 1: Open the URL provided (usually `http://localhost:8000` or `http://127.0.0.1:8000`) directly in your browser.

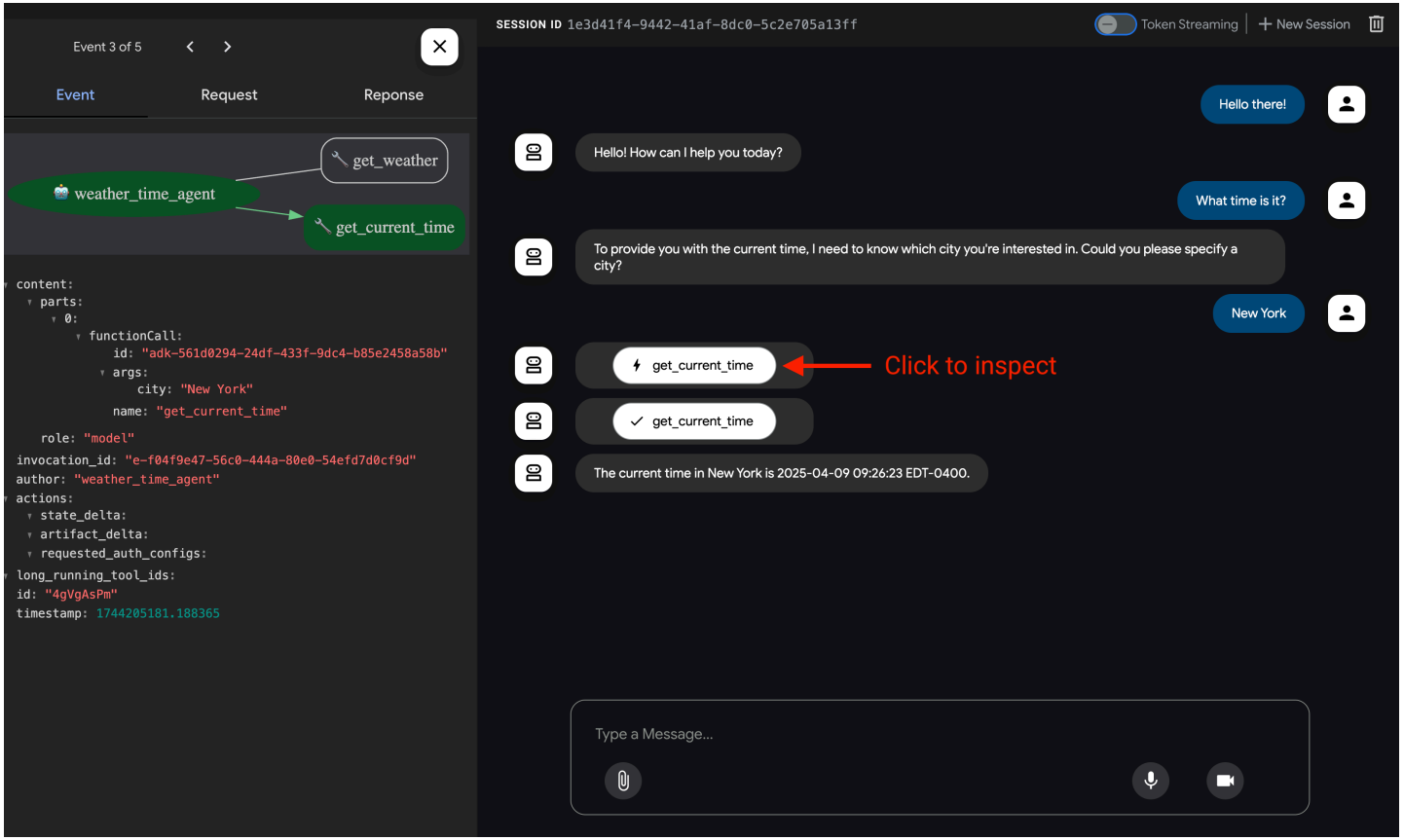
Step 2. In the top-left corner of the UI, you can select your agent in the dropdown. Select "multi_tool_agent".

> **Troubleshooting**
>
> If you do not see "multi_tool_agent" in the dropdown menu, make sure you are running `adk web` in the **parent folder** of your agent folder (i.e. the parent folder of multi_tool_agent).

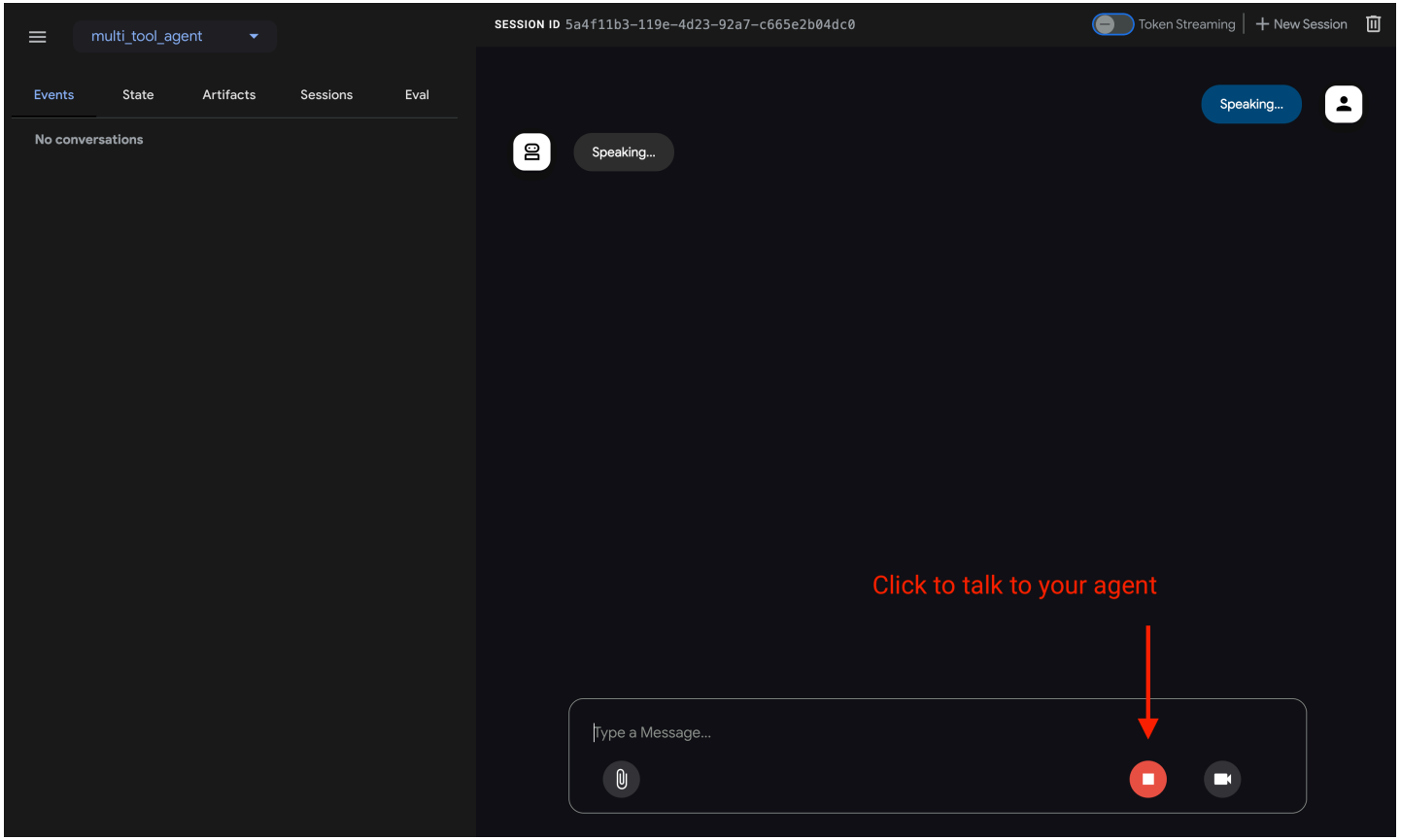
Step 3. Now you can chat with your agent using the textbox:



Step 4. You can also inspect individual function calls, responses and model responses by clicking on the actions:



Step 5. You can also enable your microphone and talk to your agent:



Model support

Currently only `gemini-2.0-flash-exp` supports talking to your agent via audio/video, and can be used either with your API key from Google AI Studio or via Vertex AI.

Terminal (adk run)

Run the following command, to chat with your Google Search agent.

```
adk run multi_tool_agent
```

```
Running agent weather_time_agent, type exit to exit.
user: hello!
[weather_time_agent]: Hello! How can I help you today?

user: what's the weather like today?
[weather_time_agent]: Could you please tell me which city you're interested in?

user: new york
[weather_time_agent]: OK. The weather in New York is sunny with a temperature of 25 degrees Celsius (41 degrees Fahrenh
eit).

user:
```

To exit, use Cmd/Ctrl+C.

API Server (adk api_server)

`adk api_server` enables you to create a local FastAPI server in a single command, enabling you to test local cURL requests before you deploy your agent.

```
INFO:     Started server process [12809]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

To learn how to use `adk api_server`, see the documentation on local testing.

📝 Example prompts to try

- What is the weather in New York?
- What is the time in New York?
- What is the weather in Paris?
- What is the time in Paris?

🎉 Congratulations!

You've successfully created and interacted with your first agent using ADK!

🛣️ Next steps

- **Go to the tutorial**: Learn how to add memory, session, state to your agent: tutorial.
- **Delve into advanced configuration:** Explore the setup section for deeper dives into project structure, configuration, and other interfaces.
- **Understand Core Concepts:** Learn about agents concepts.