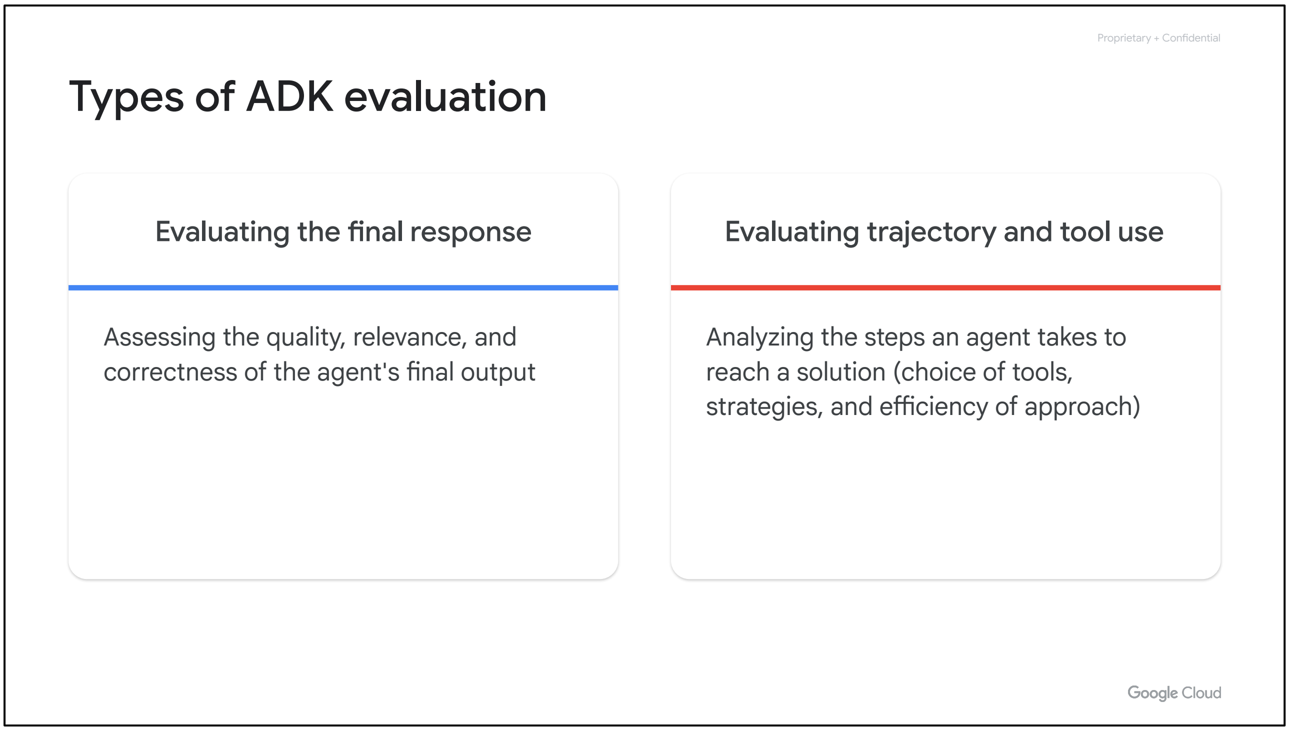
Types of ADK evaluation

Let’s start off with types of ADK evaluation.


Proprietary + Confidential

Types of ADK evaluation

Evaluating the final response

Assessing the quality, relevance, and correctness of the agent's final output

Evaluating trajectory and tool use

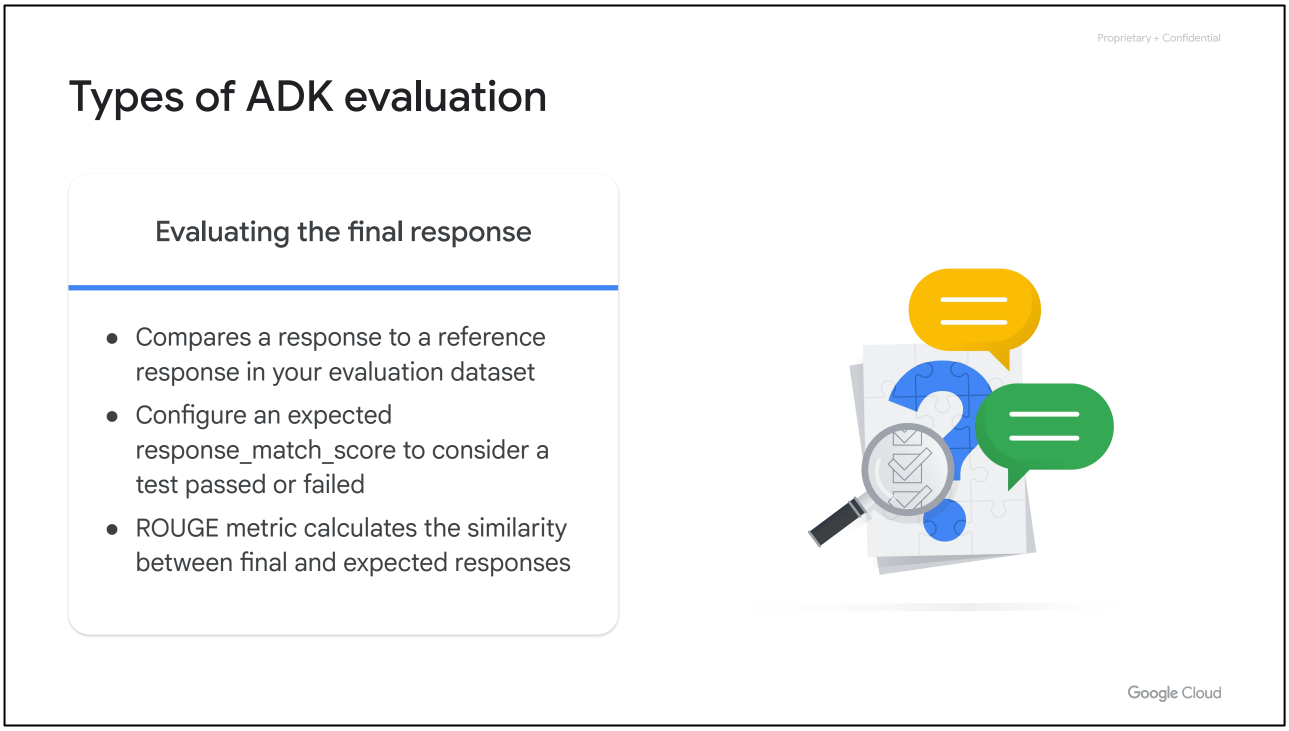
Analyzing the steps an agent takes to reach a solution (choice of tools, strategies, and efficiency of approach)

Google Cloud

ADK agent evaluation currently provides two types of evaluation:

- Evaluating the final response involves assessing the quality, relevance, and correctness of the agent's final output.
- Evaluating trajectory and tool use refers to analyzing the steps an agent takes to reach a solution, including its choice of tools, strategies, and the efficiency of its approach. The trajectory is a list of steps the agent took before it returned to the user. You can compare that against the list of steps you expect the agent to have taken.

Let’s look at each of these in more detail.



Proprietary + Confidential

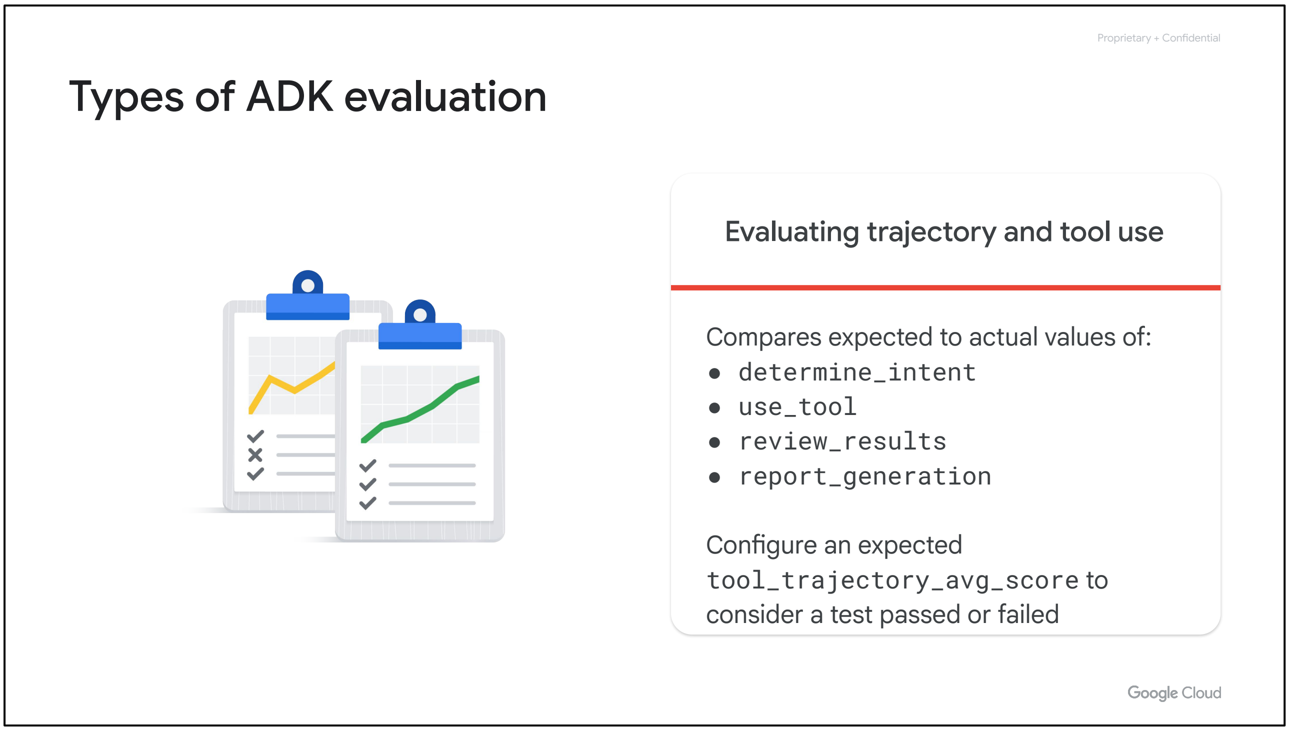
Types of ADK evaluation

Evaluating the final response

- Compares a response to a reference response in your evaluation dataset
- Configure an expected response_match_score to consider a test passed or failed
- ROUGE metric calculates the similarity between final and expected responses

Google Cloud

- Evaluating the final response compares a response to a reference response in your eval dataset. In a test file the reference is the expected final response from the model.
- You configure an expected response_match_score to consider a test passed or failed.
- This metric compares the agent's final natural language response to the expected final response, stored in the reference field. The ROUGE metric is used to calculate the similarity between the two responses.
- The response match score defaults to 0.8, allowing for a small margin of error in the agent's natural language responses.



Types of ADK evaluation

Evaluating trajectory and tool use

Compares expected to actual values of:
- `determine_intent`
- `use_tool`
- `review_results`
- `report_generation`

Configure an expected `tool_trajectory_avg_score` to consider a test passed or failed

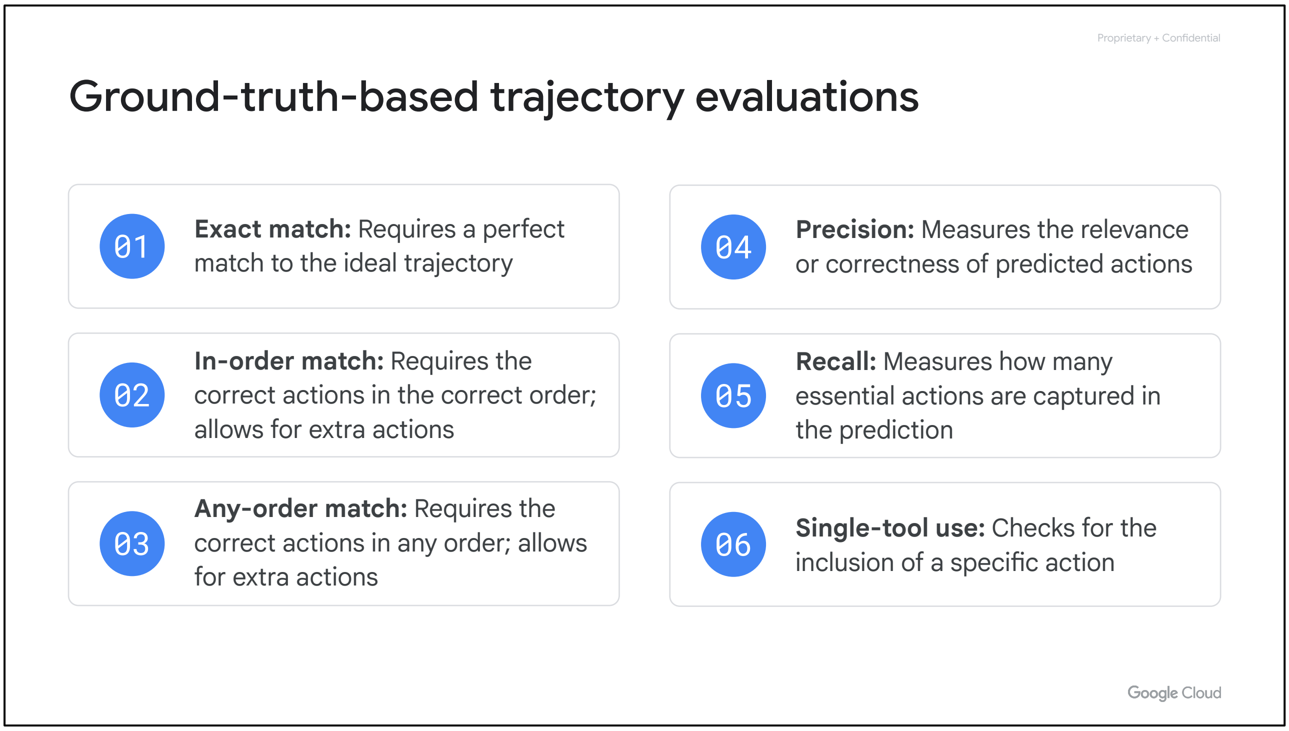
Before responding to a user, an agent typically performs a series of actions, which is known as a 'trajectory.' It might compare the user input with session history to disambiguate a term, or lookup a policy document, search a knowledge base or invoke an API to save a ticket. This is called a ‘trajectory’ of actions. Evaluating an agent's performance, requires comparing its actual trajectory, to an expected, or ideal one. This comparison can reveal errors and inefficiencies in the agent's process. The expected trajectory represents the ground truth -- the list of steps you anticipate the agent should take.

Evaluating trajectory and tool use compares expected values to actual values of:
- determine_intent
- use_tool
- review_results
- report_generation

You configure an expected tool_trajectory_avg_score to consider a test passed or failed.
This metric compares the agent's actual tool usage during the evaluation against the expected tool usage defined in the expected_tool_use field. Each matching tool usage step receives a score of 1, while a mismatch receives a score of 0. The final score is the average of these matches, representing the accuracy of the tool usage trajectory.

Tool trajectory avg score defaults to 1.0, requiring a 100% match in the tool usage trajectory.

Let’s look at some existing trajectory evaluations.



Ground-truth-based trajectory evaluations

01 **Exact match:** Requires a perfect match to the ideal trajectory

02 **In-order match:** Requires the correct actions in the correct order; allows for extra actions

03 **Any-order match:** Requires the correct actions in any order; allows for extra actions

04 **Precision:** Measures the relevance or correctness of predicted actions

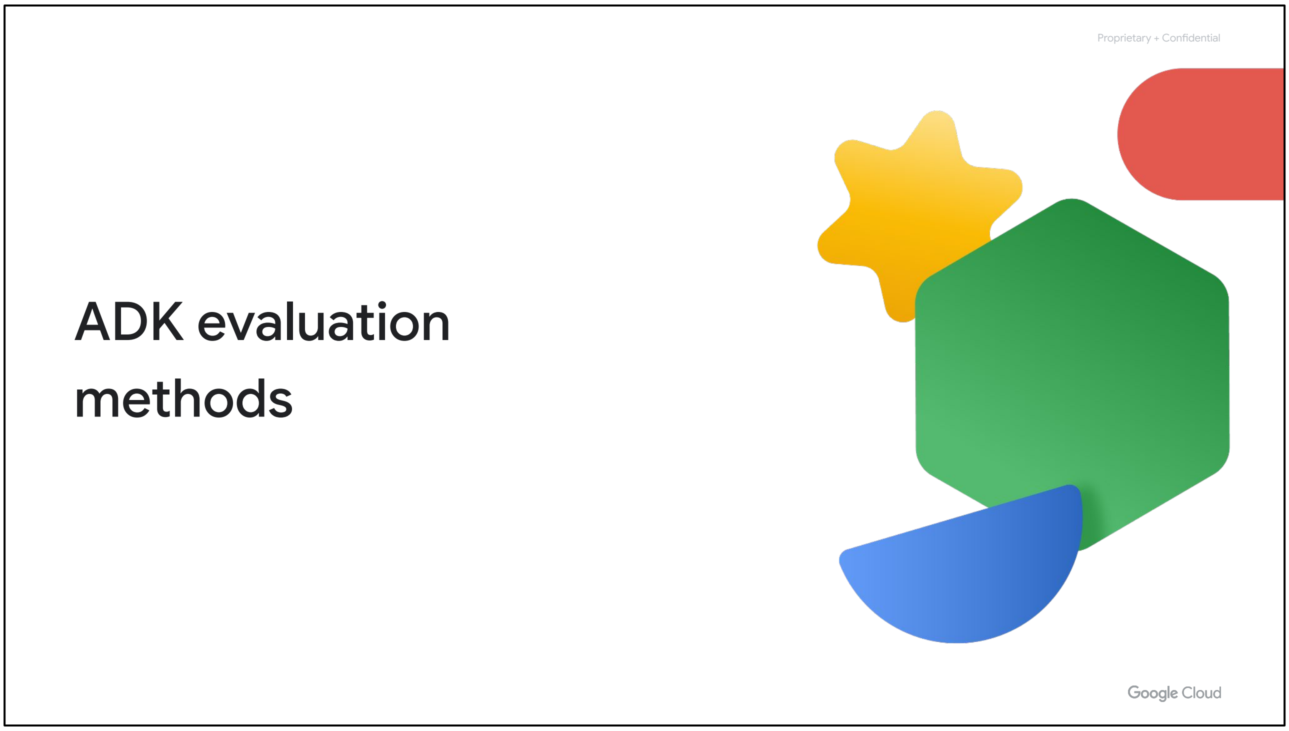
05 **Recall:** Measures how many essential actions are captured in the prediction

06 **Single-tool use:** Checks for the inclusion of a specific action

There are several ground-truth-based trajectory evaluations:

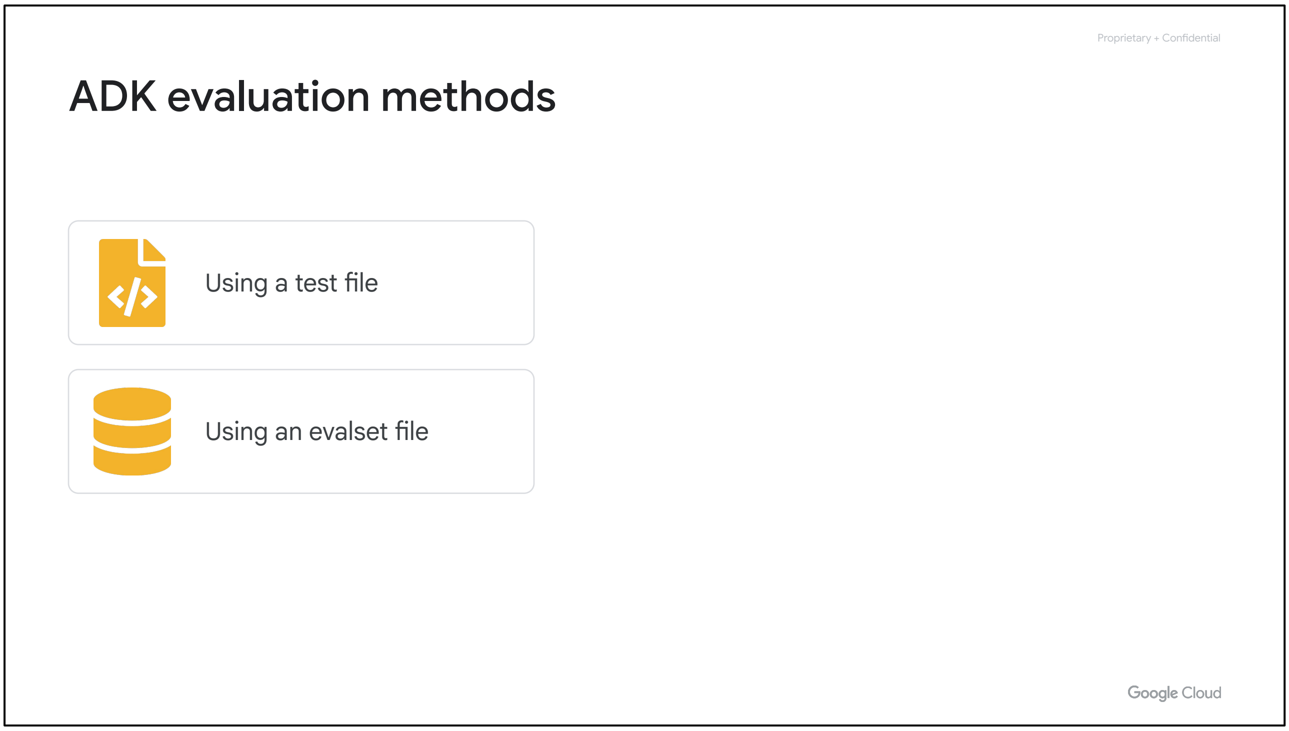
- **Exact match:** Requires a perfect match to the ideal trajectory.
- **In-order match:** Requires the correct actions in the correct order, and allows for extra actions.
- **Any-order match:** Requires the correct actions in any order, and allows for extra actions.
- **Precision:** Measures the relevance or correctness of predicted actions.
- **Recall:** Measures how many essential actions are captured in the prediction.
- And **Single-tool use:** Checks for the inclusion of a specific action.

Choosing the right evaluation metric depends on the specific requirements and goals of your agent. For instance, in high-stakes scenarios, an exact match might be crucial, while in more flexible situations, an in-order or any-order match might suffice.



ADK evaluation methods

Now let’s look at ADK evaluation methods.

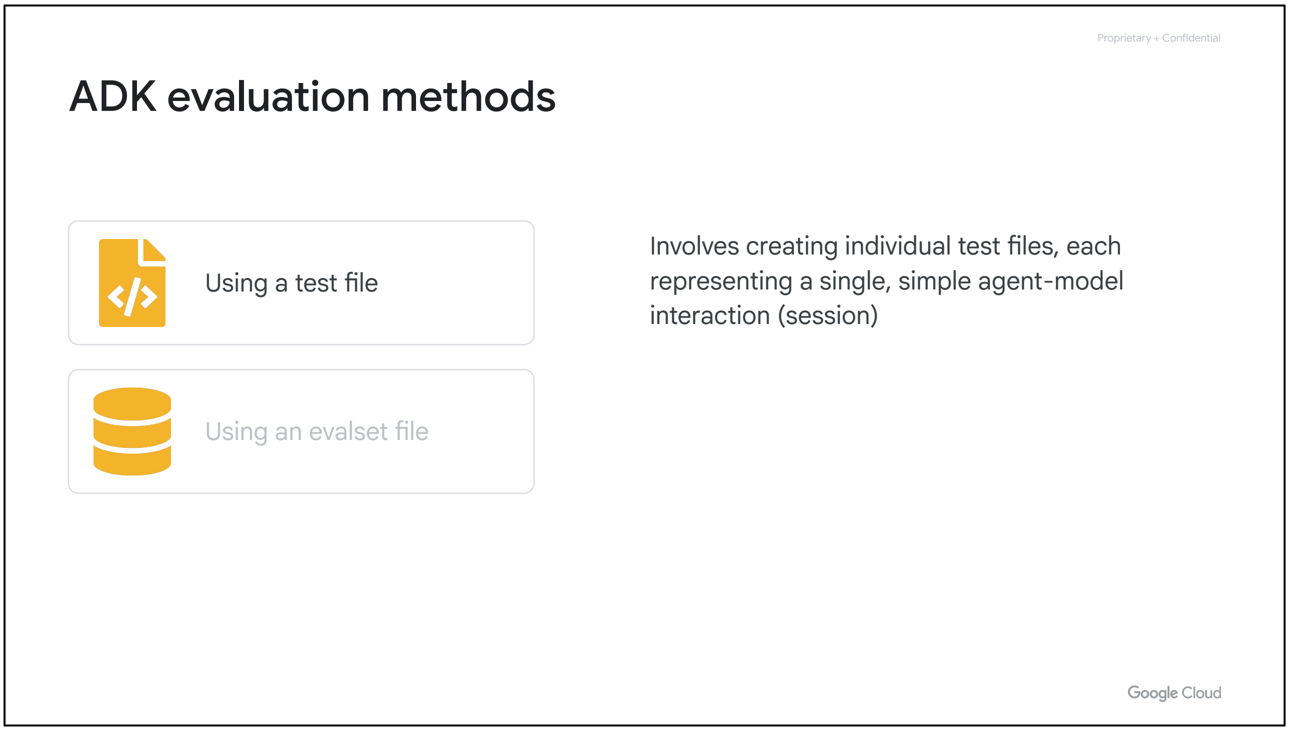


ADK evaluation methods

Using a test file

Using an evalset file

ADK evaluation currently provides two evaluation methods:



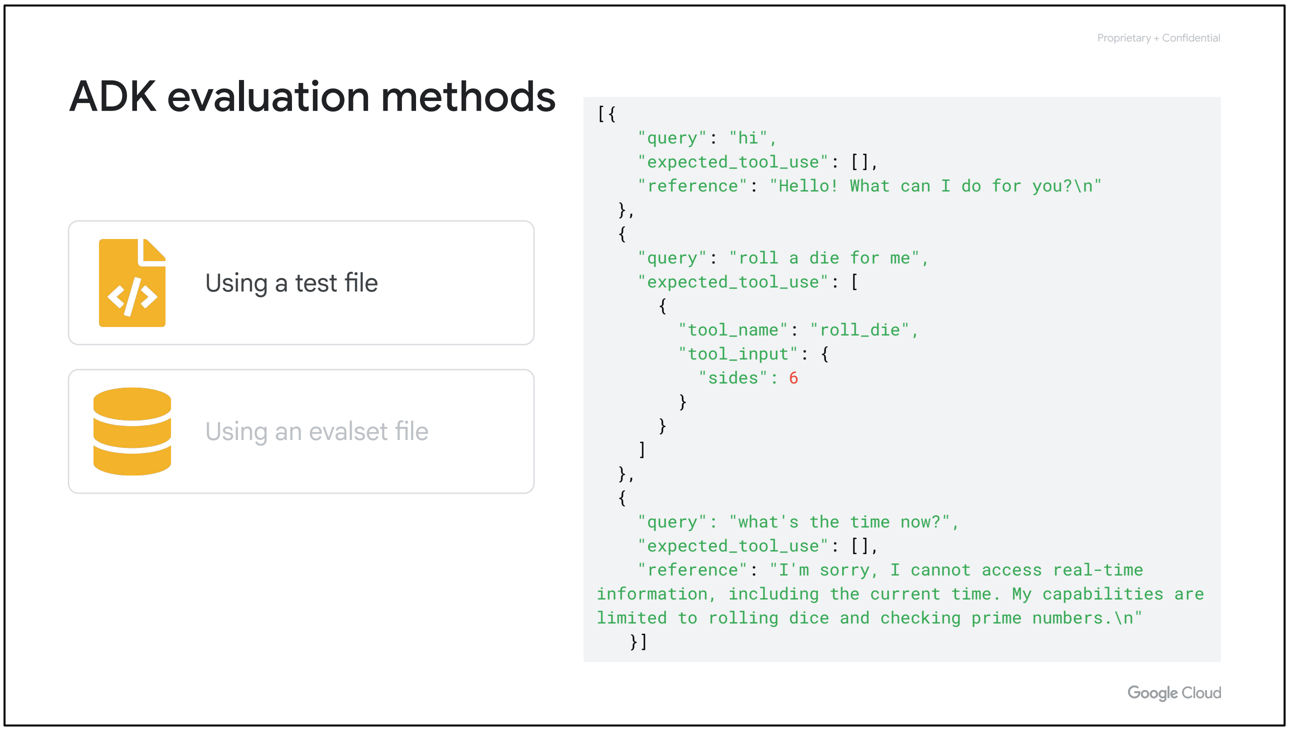
ADK evaluation methods

Using a test file

Using an evalset file

Involves creating individual test files, each representing a single, simple agent-model interaction (session)

- The **test file approach** involves creating individual test files, each representing a single, simple agent-model interaction (a session). It's most effective during active agent development, serving as a form of unit testing. These tests are designed for rapid execution and should focus on simple session complexity. Each test file contains a single session, which may consist of multiple turns. A turn represents a single interaction between the user and the agent.



ADK evaluation methods

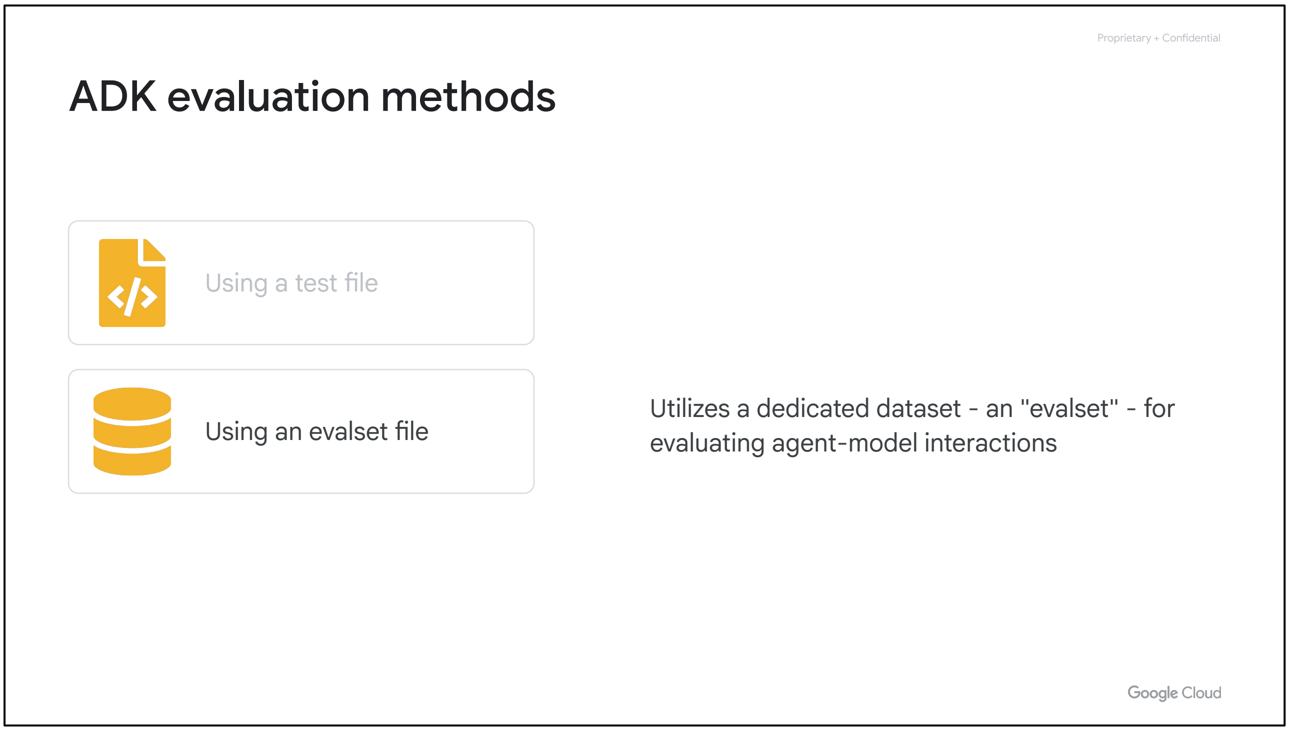
Using a test file

Using an evalset file

```
[{
    "query": "hi",
    "expected_tool_use": [],
    "reference": "Hello! What can I do for you?\n"
  },
  {
    "query": "roll a die for me",
    "expected_tool_use": [
      {
        "tool_name": "roll_die",
        "tool_input": {
          "sides": 6
        }
      }
    ]
  },
  {
    "query": "what's the time now?",
    "expected_tool_use": [],
    "reference": "I'm sorry, I cannot access real-time
information, including the current time. My capabilities are
limited to rolling dice and checking prime numbers.\n"
   }]
```

Here is an example test file.

A test file contains a session with queries, expected tool use, and reference responses. You can give the file any name for example evaluation.test.json.The framework only checks for the .test.json suffix, and the preceding part of the filename is not constrained.



ADK evaluation methods

Using a test file

Using an evalset file

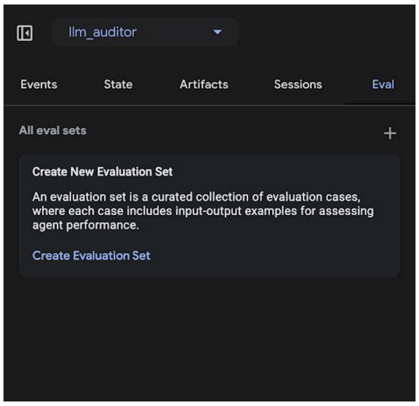
Utilizes a dedicated dataset - an "evalset" - for evaluating agent-model interactions

The **evalset** approach utilizes a dedicated dataset called an "evalset" for evaluating agent-model interactions. Similar to a test file, the evalset contains example interactions. However, an evalset can contain multiple, potentially lengthy sessions, making it ideal for simulating complex, multi-turn conversations. Due to its ability to represent complex sessions, the evalset is well-suited for integration tests. These tests are typically run less frequently than unit tests due to their more extensive nature.

ADK evaluation methods

Using a test file

Using an evalset file



Create New Evaluation Set

An evaluation set is a curated collection of evaluation cases, where each case includes input-output examples for assessing agent performance.

Create Evaluation Set

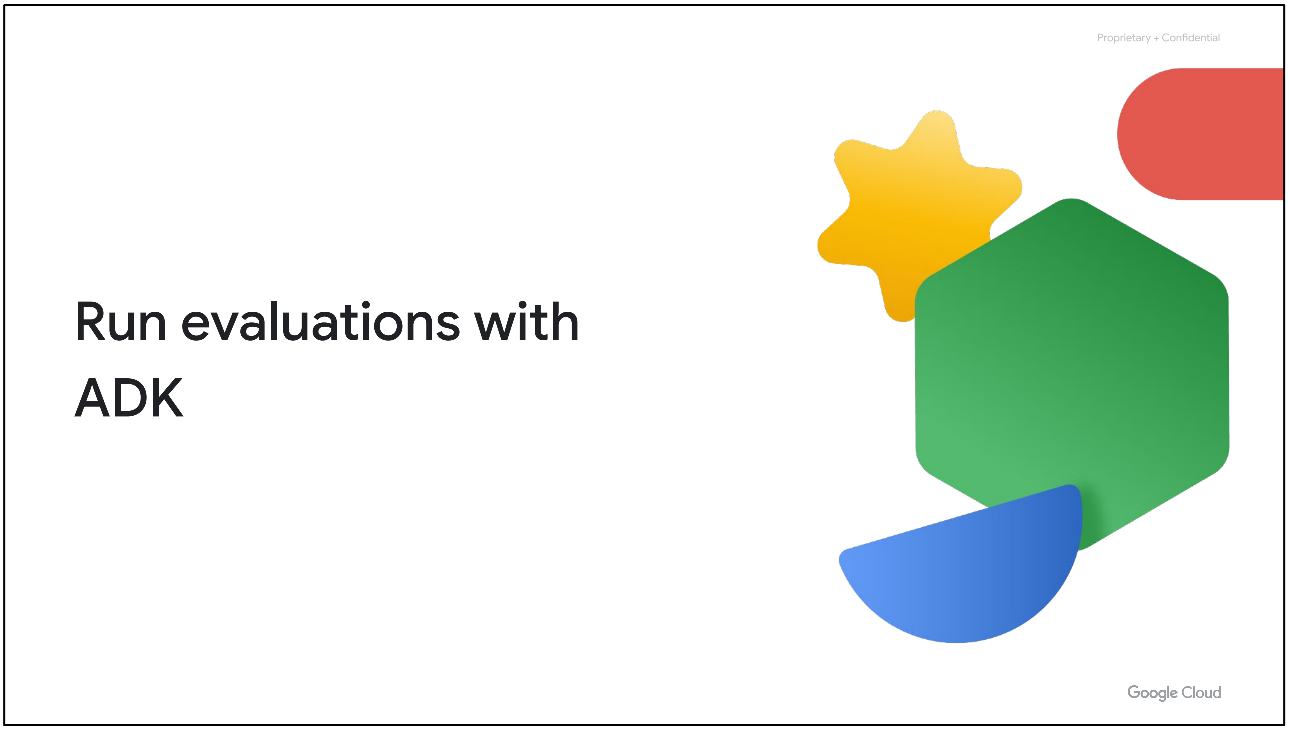
- Create an evaluation set file.
- Add a current session as an eval case to your file.
- Run evaluations with a new version of your agent.

An evalset file contains multiple "e-vals," each representing a distinct session. Each e-val consists of one or more "turns," which include the user query, expected tool use, expected intermediate agent responses, and a reference response. These fields have the same meaning as they do in the test file approach. Each e-val is identified by a unique name. Furthermore, each e-val includes an associated initial session state.

Creating evalsets manually can be complex, therefore UI tools are provided to help capture relevant sessions and easily convert them into evals within your evalset.

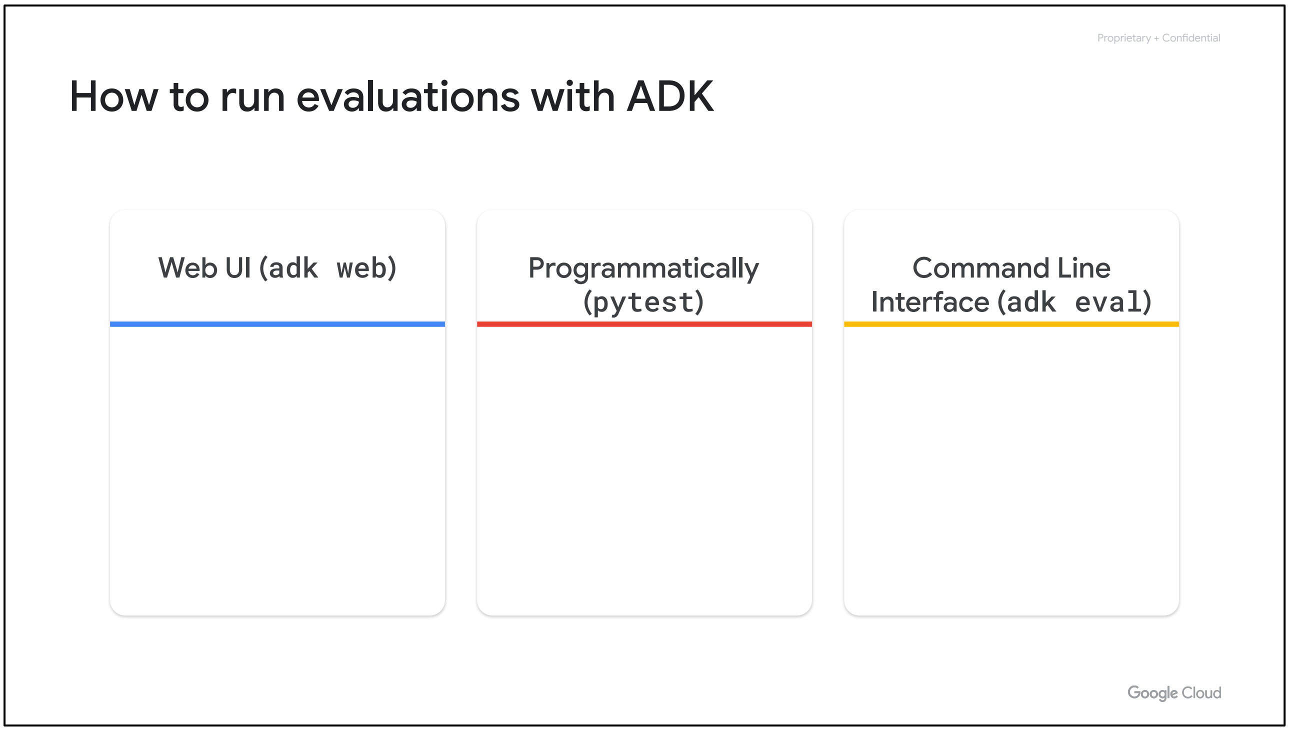
In the **Eval** tab of Web UI, you can **Create an evaluation set** file, and add a current session as an eval **Case** to your file. Please note, you can load past saved sessions and then add them to the e-val set.

You can also **run evaluations**, for example with a new version of your agent.



Run evaluations with ADK

Now let’s look at how to run evaluations with ADK.



Proprietary + Confidential

How to run evaluations with ADK

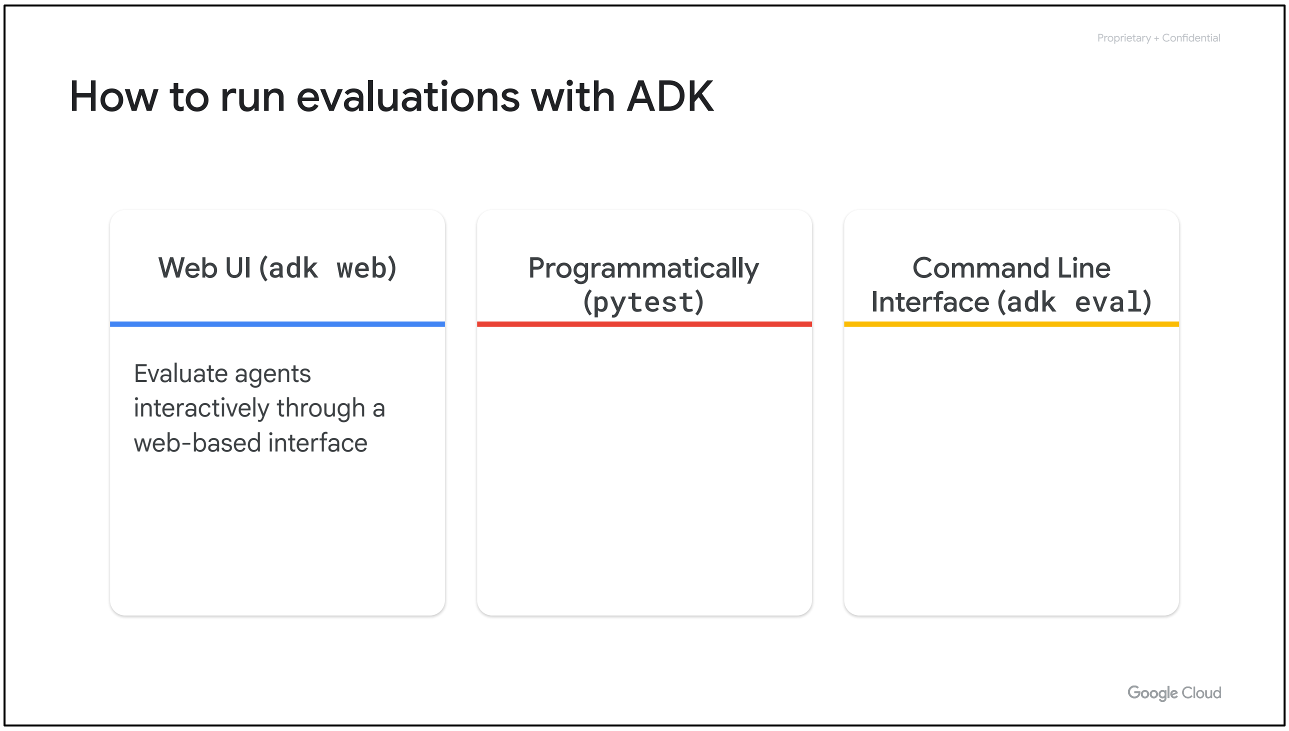
Web UI (`adk web`)

Programmatically (`pytest`)

Command Line Interface (`adk eval`)

Google Cloud

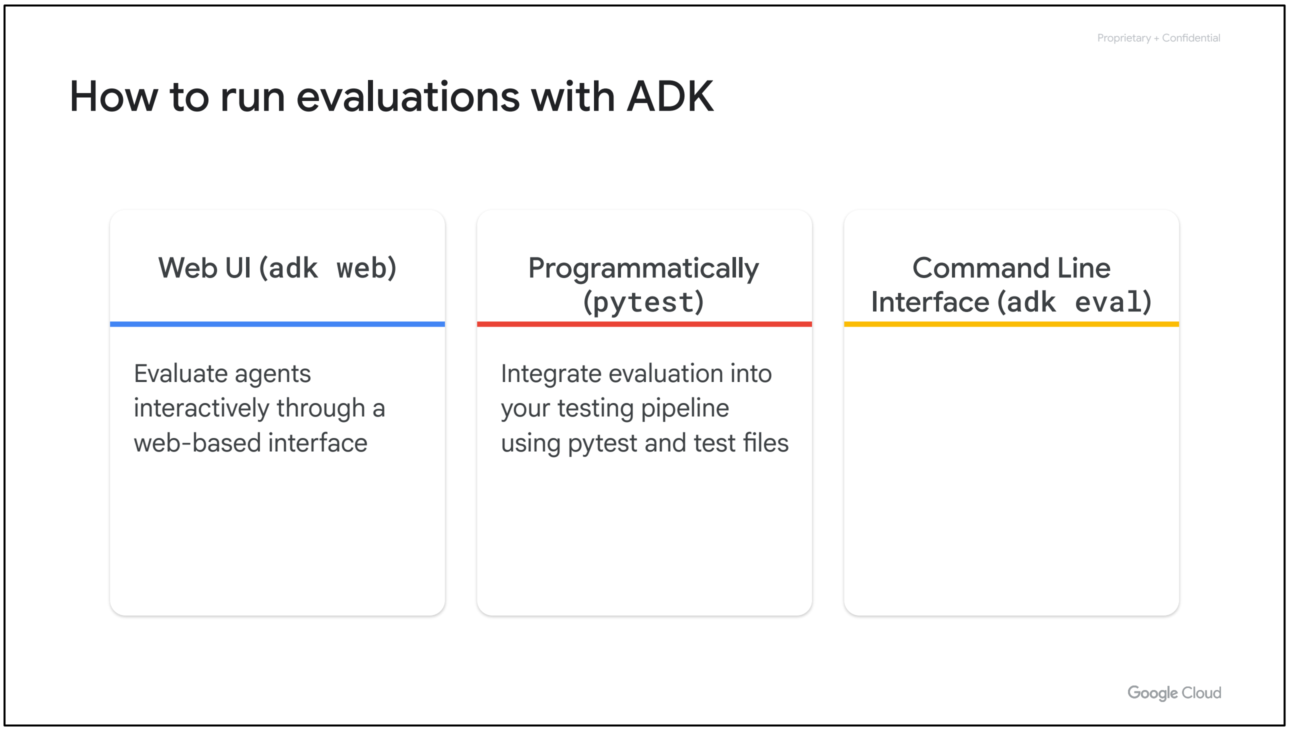
As a developer, you can evaluate your agents using ADK in the following ways:



How to run evaluations with ADK

Web UI (`adk web`)	Programmatically (`pytest`)	Command Line Interface (`adk eval`)
Evaluate agents interactively through a web-based interface		

- Through the web-based UI, you can evaluate agents interactively through a web-based interface.



How to run evaluations with ADK

Web UI (`adk web`)

Evaluate agents interactively through a web-based interface

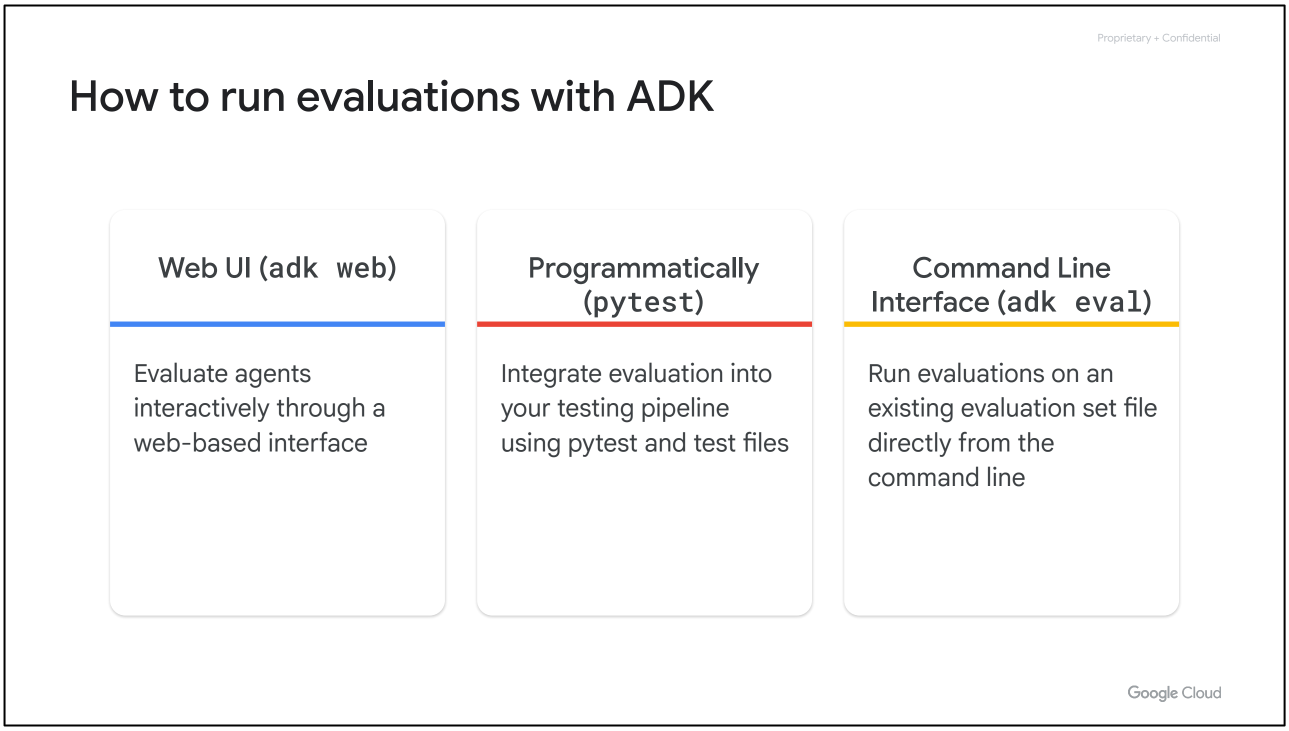
Programmatically (`pytest`)

Integrate evaluation into your testing pipeline using pytest and test files

Command Line Interface (`adk eval`)

Google Cloud

- Programmatically, you can integrate evaluation into your testing pipeline using pytest and test files.



How to run evaluations with ADK

Web UI (`adk web`)

Evaluate agents interactively through a web-based interface

Programmatically (`pytest`)

Integrate evaluation into your testing pipeline using pytest and test files

Command Line Interface (`adk eval`)

Run evaluations on an existing evaluation set file directly from the command line

Google Cloud

- And, through a Command Line Interface, you can run evaluations on an existing evaluation set file directly from the command line.