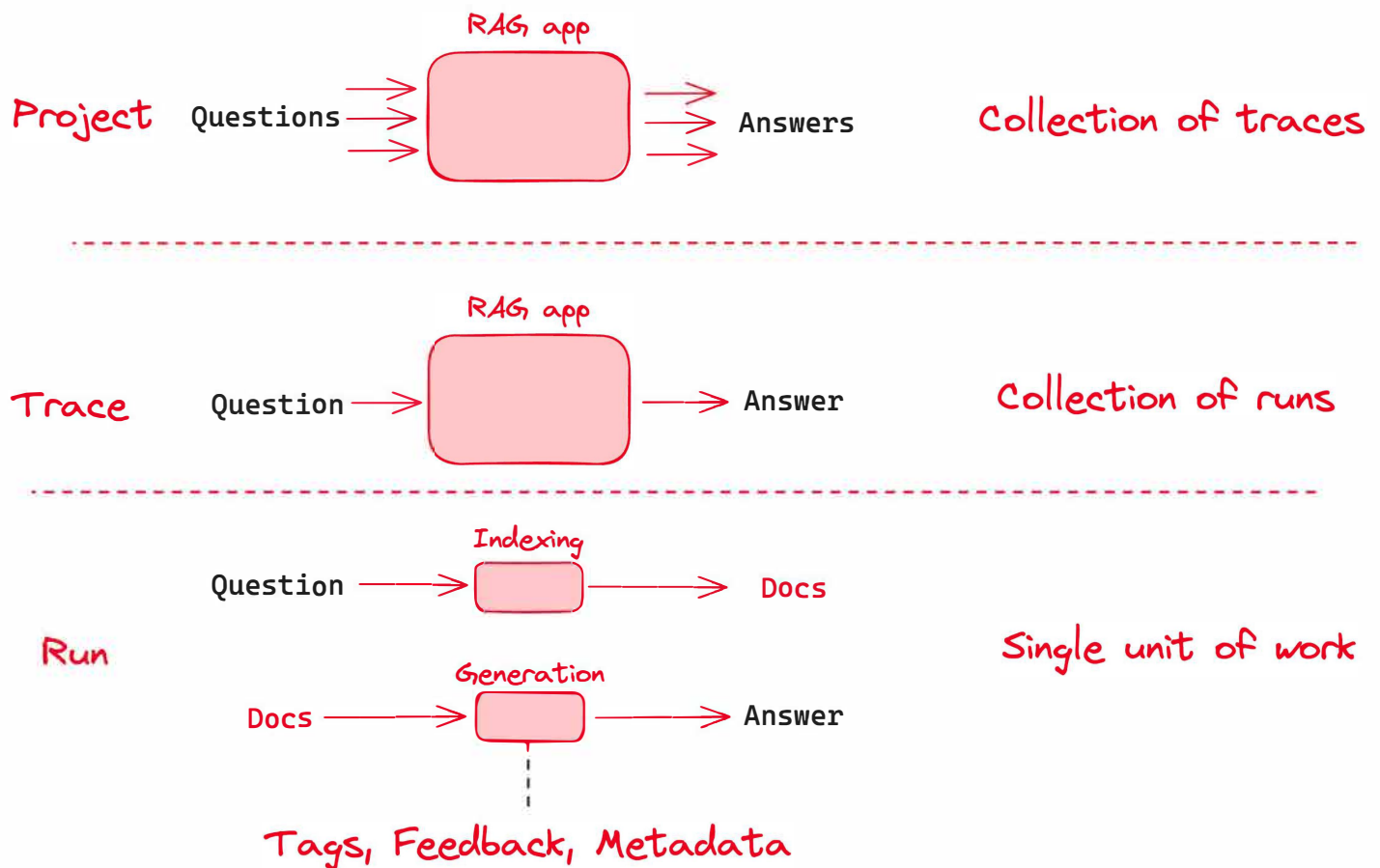




# Concepts

This conceptual guide covers topics that are important to understand when logging traces to LangSmith. A **Trace** is essentially a series of steps that your application takes to go from input to output. Each of these individual steps is represented by a **Run**. A **Project** is simply a collection of traces. The following diagram displays these concepts in the context of a simple RAG app, which retrieves documents from an index and generates an answer.

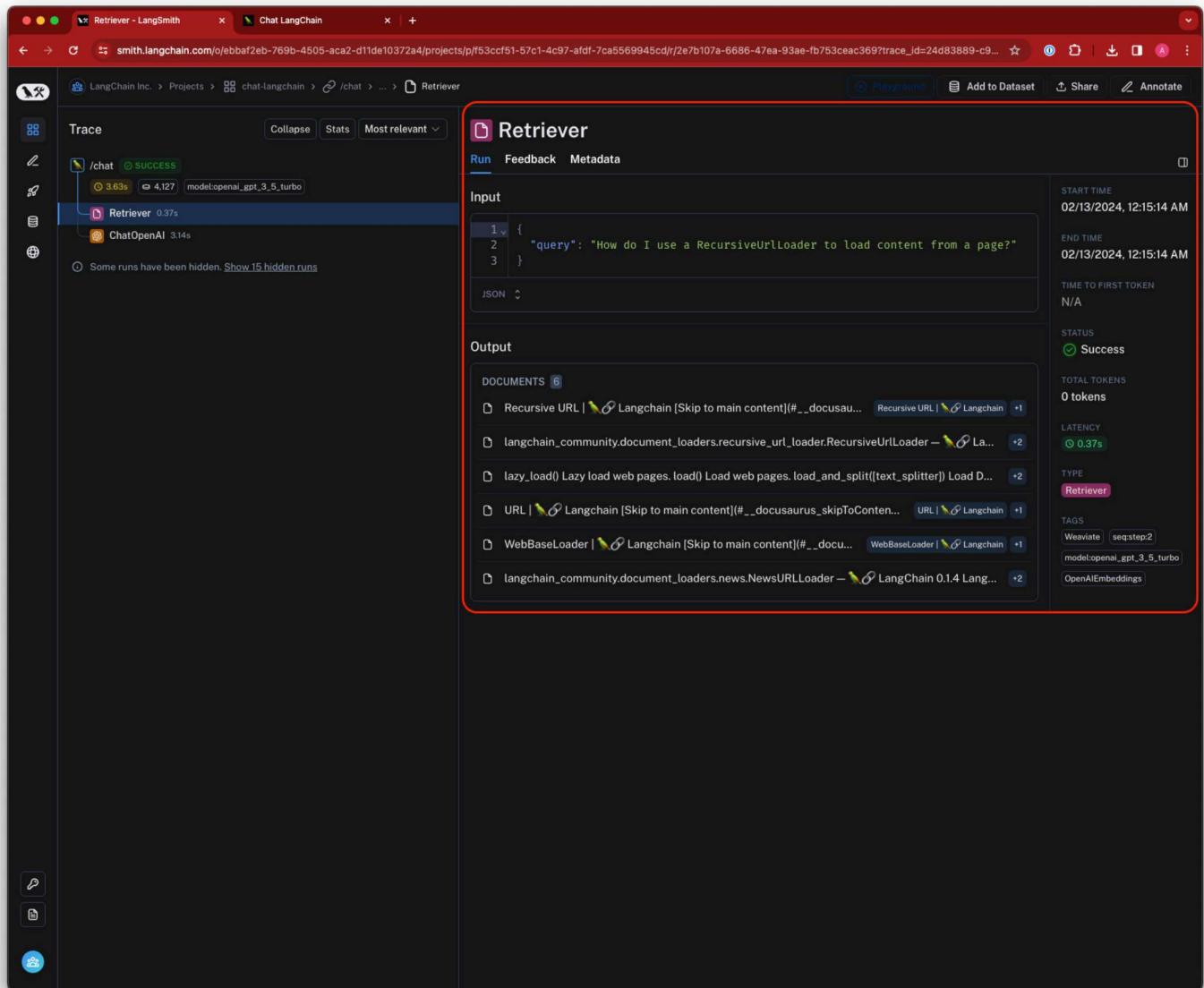


Primitive datatypes in LangSmith

## Runs

A **Run** is a span representing a single unit of work or operation within your LLM application. This could be anything from single call to an LLM or chain, to a prompt formatting call, to a runnable lambda invocation. If you are familiar with [OpenTelemetry](#), you can think of a run as a span.

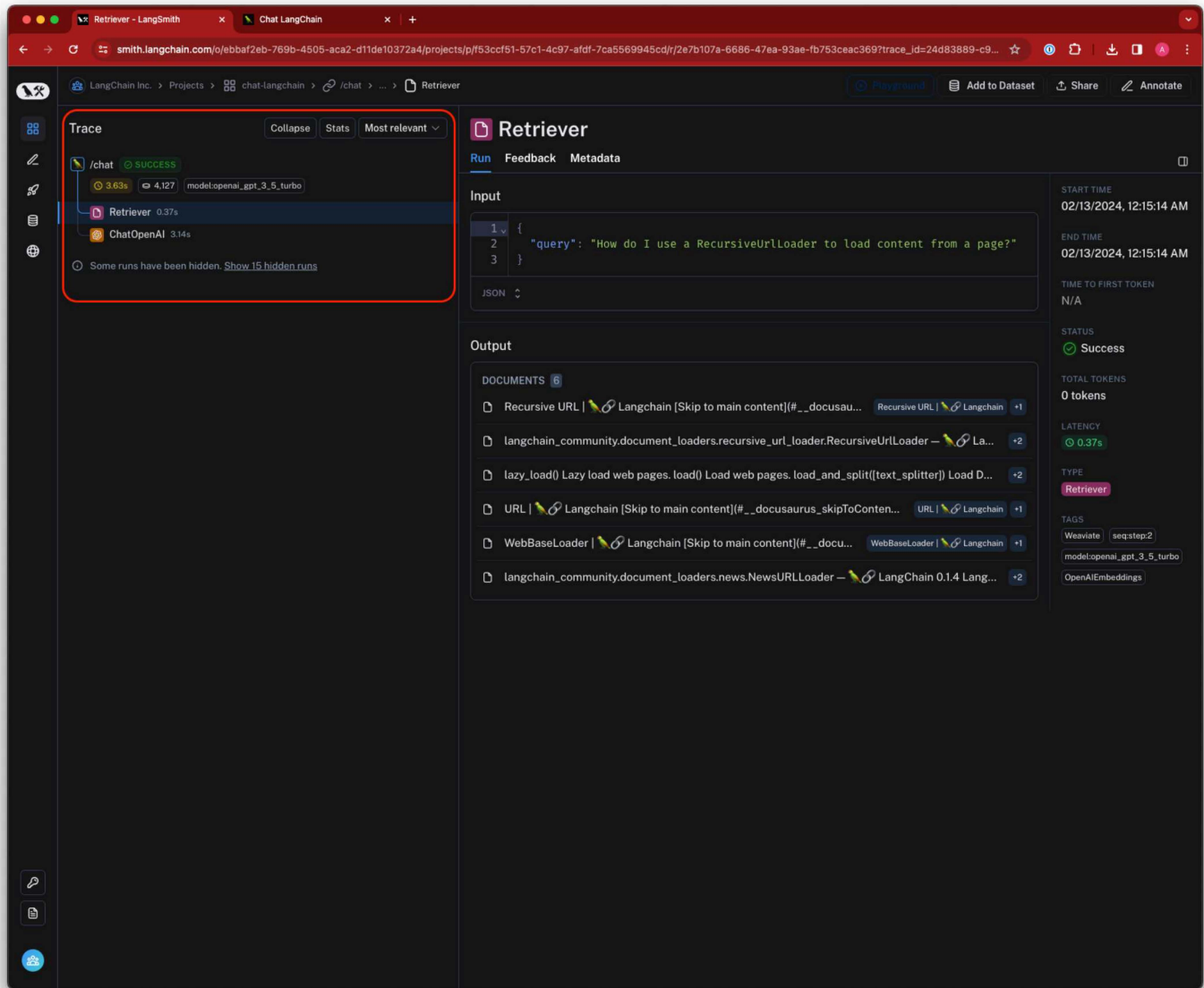
To learn more about how runs are stored in the application, see [this reference guide](#).



## Traces

A **Trace** is a collection of runs that are related to a single operation. For example, if you have a user request that triggers a chain, and that chain makes a call to an LLM, then to an output parser, and so on, all of these runs would be part of the same trace. If you are familiar with [OpenTelemetry](#), you can

think of a LangSmith trace as a collection of spans. Runs are bound to a trace by a unique trace ID.



The screenshot displays the LangSmith web interface. On the left, a 'Trace' panel shows a sequence of spans: '/chat' (SUCCESS, 3.63s), 'Retriever' (0.37s), and 'ChatOpenAI' (3.14s). The 'Retriever' span is selected, showing its details in the main panel. The 'Input' section shows a JSON object with a 'query' field. The 'Output' section shows a list of documents retrieved from various sources. The right sidebar contains metadata such as 'START TIME', 'END TIME', 'TIME TO FIRST TOKEN', 'STATUS', 'TOTAL TOKENS', 'LATENCY', 'TYPE', and 'TAGS'.

**Trace**

- /chat SUCCESS 3.63s 4.127 modelopenai\_gpt\_3\_5\_turbo
- Retriever 0.37s
- ChatOpenAI 3.14s

Some runs have been hidden. [Show 15 hidden runs](#)

**Retriever**

Run Feedback Metadata

**Input**

```
1 {
2   "query": "How do I use a RecursiveUrlLoader to load content from a page?"
3 }
```

JSON

**Output**

DOCUMENTS 6

- Recursive URL | Langchain [Skip to main content](#\_\_docsausau... Recursive URL | Langchain +1
- langchain\_community.document\_loaders.recursive\_url\_loader.RecursiveUrlLoader — La... +2
- lazy\_load() Lazy load web pages. load() Load web pages. load\_and\_split([text\_splitter]) Load D... +2
- URL | Langchain [Skip to main content](#\_\_docsausurus\_skipToConten... URL | Langchain +1
- WebBaseLoader | Langchain [Skip to main content](#\_\_docu... WebBaseLoader | Langchain +1
- langchain\_community.document\_loaders.news.NewsURLLoader — LangChain 0.1.4 Lang... +2

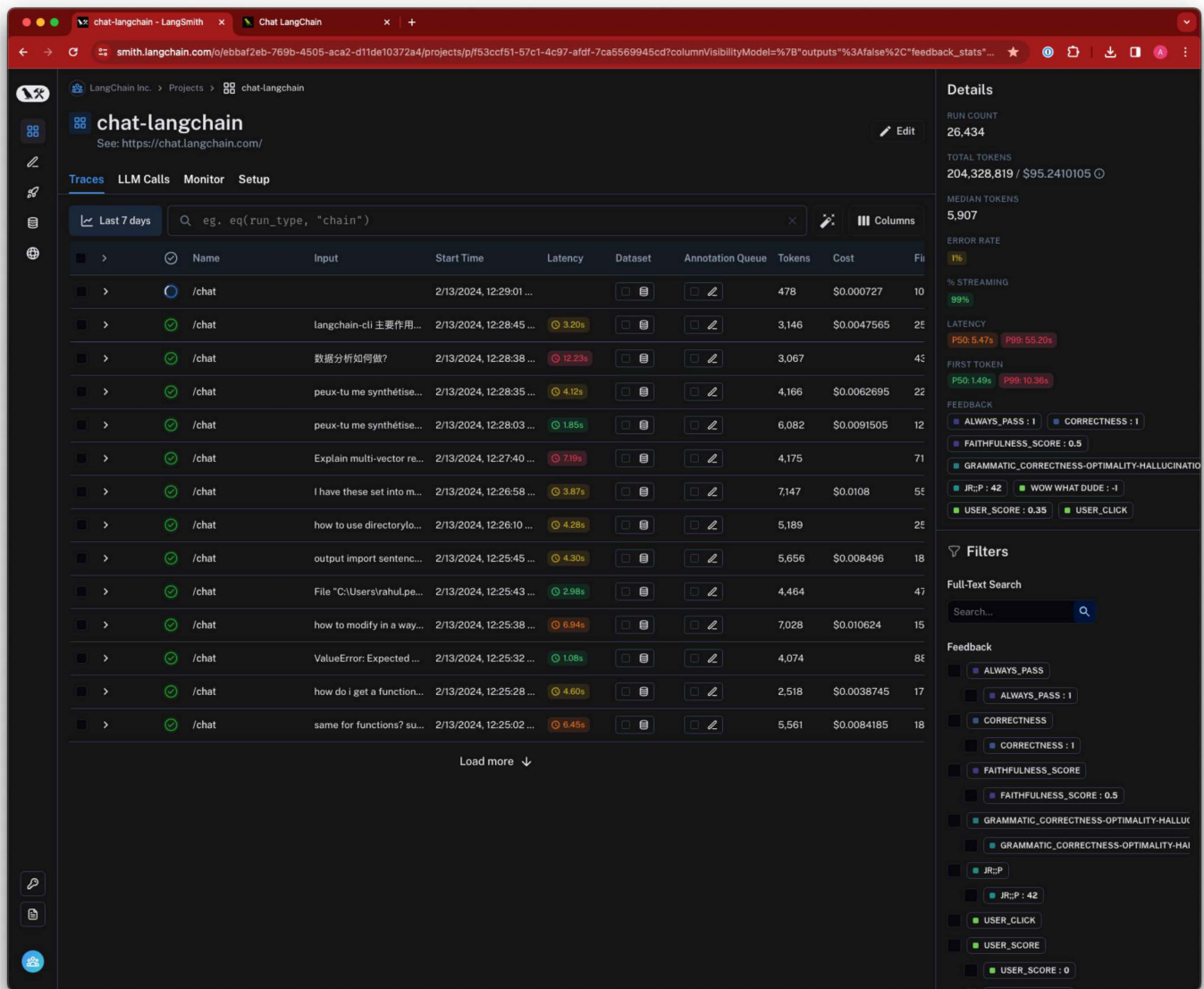
**Metadata**

- START TIME: 02/13/2024, 12:15:14 AM
- END TIME: 02/13/2024, 12:15:14 AM
- TIME TO FIRST TOKEN: N/A
- STATUS: Success
- TOTAL TOKENS: 0 tokens
- LATENCY: 0.37s
- TYPE: Retriever
- TAGS: Weaviate, seqstep:2, modelopenai\_gpt\_3\_5\_turbo, OpenAIEmbeddings

## Projects

A **Project** is a collection of traces. You can think of a project as a container for all the traces that are related to a single application or service. You can have multiple projects, and each project can

have multiple traces.



The screenshot displays the LangSmith web interface for a project named 'chat-langchain'. The main area shows a table of traces with columns for Name, Input, Start Time, Latency, Dataset, Annotation Queue, Tokens, Cost, and File. The table lists several chat interactions with their respective inputs, timestamps, latencies, and costs. The right-hand panel provides summary statistics and filters for the traces.

**Details Summary:**

- RUN COUNT: 26,434
- TOTAL TOKENS: 204,328,819 / \$95.2410105
- MEDIAN TOKENS: 5,907
- ERROR RATE: 1%
- % STREAMING: 99%
- LATENCY: P50: 5.47s, P99: 55.20s
- FIRST TOKEN: P50: 1.49s, P99: 10.36s

**Feedback Summary:**

- ALWAYS\_PASS: 1
- CORRECTNESS: 1
- FAITHFULNESS\_SCORE: 0.5
- GRAMMATIC\_CORRECTNESS-OPTIMALITY-HALLUCINATION: 1
- JR:P: 42
- WOW WHAT DUDE: -1
- USER\_SCORE: 0.35
- USER\_CLICK: 1

## Feedback

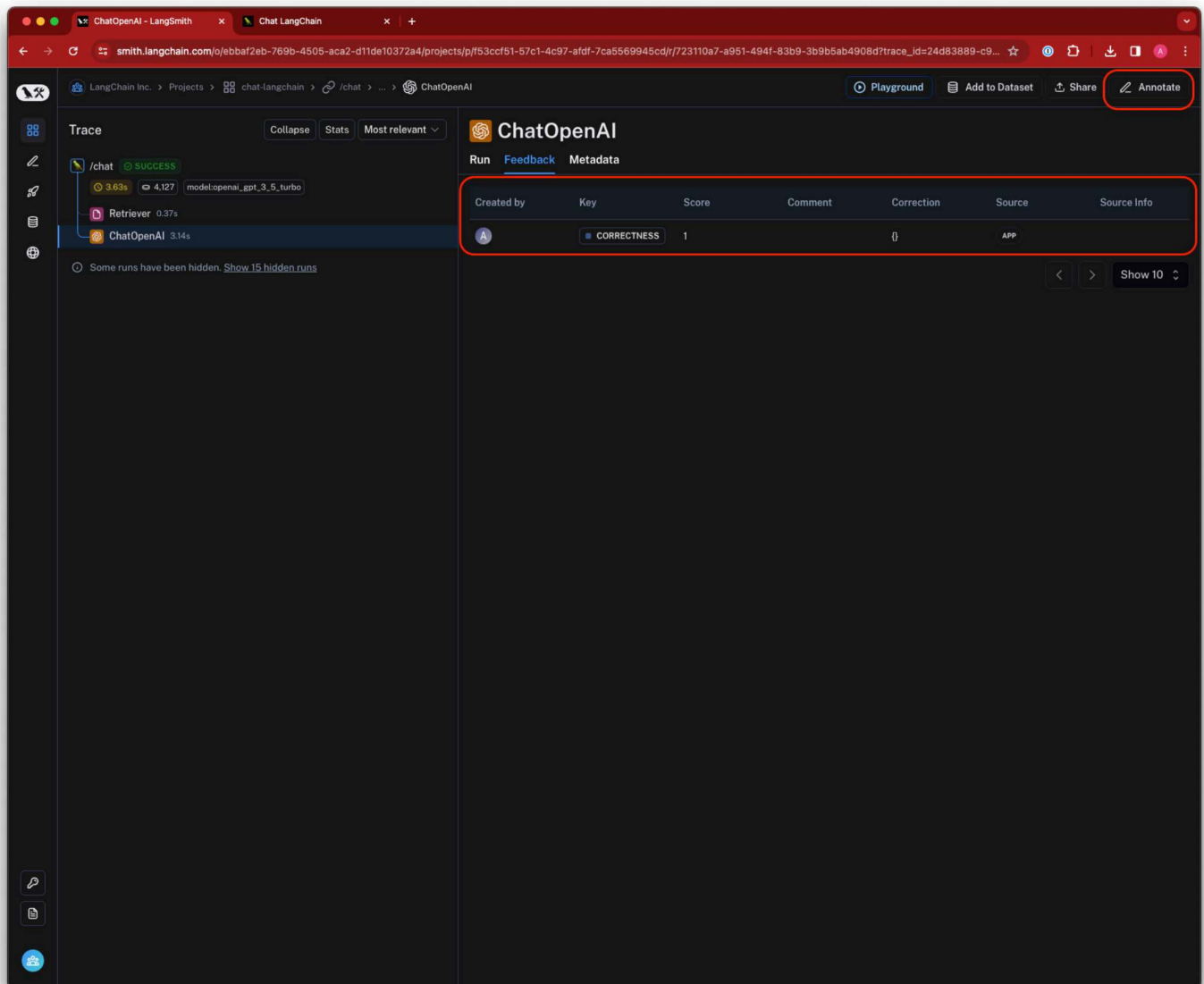
**Feedback** allows you to score an individual run based on certain criteria. Each feedback entry consists of a feedback tag and feedback score, and is bound to a run by a unique run ID. Feedback can currently be continuous or discrete (categorical), and you can reuse feedback tags across different runs within an organization.

Collecting feedback on runs can be done in a number of ways:

1. Sent up along with a trace from the LLM application
2. Generated by a user in the app [inline](#) or in an [annotation queue](#)
3. Generated by an automatic evaluator during [offline evaluation](#)

#### 4. Generated by an [online evaluator](#)

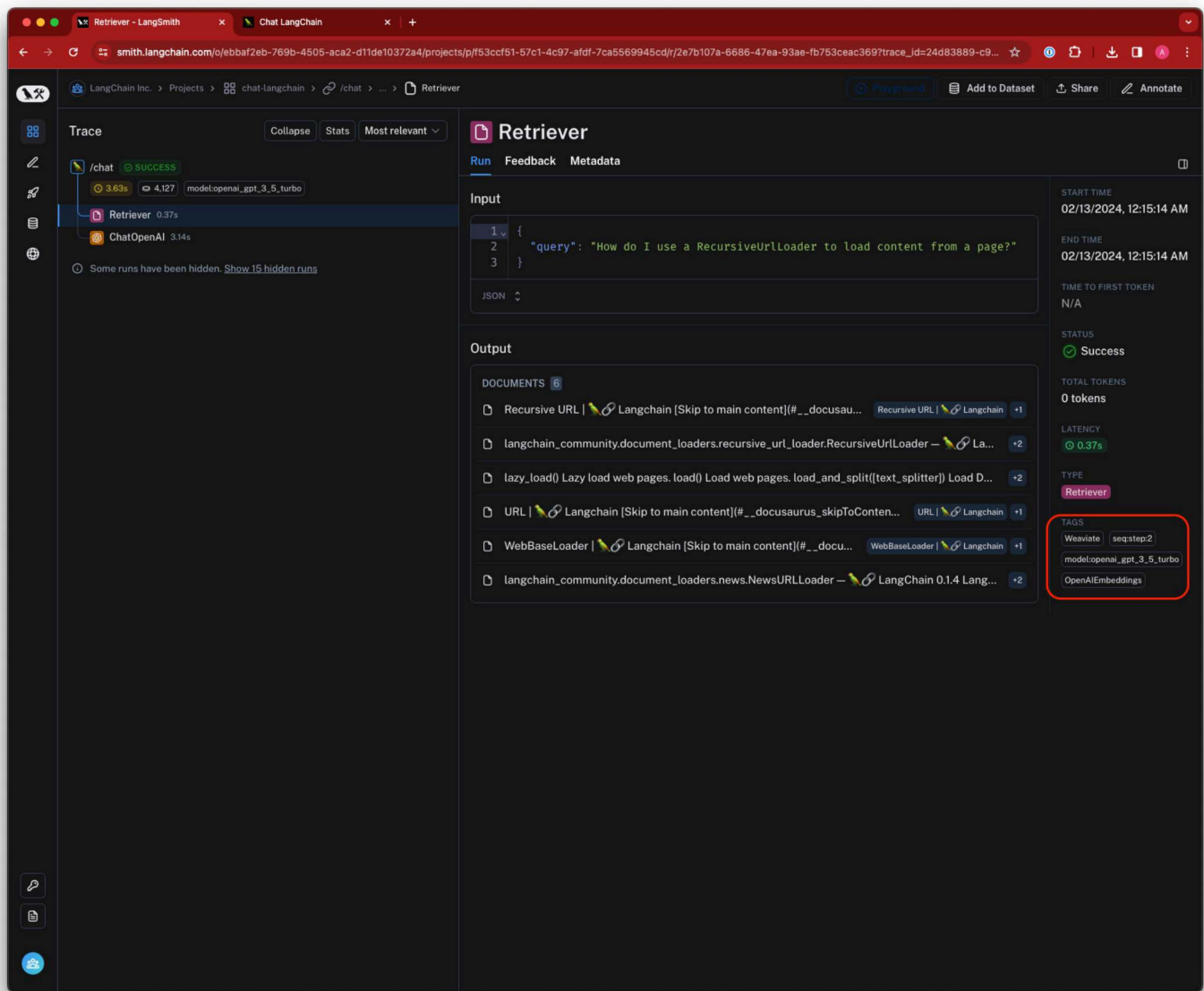
To learn more about how feedback is stored in the application, see [this reference guide](#).



## Tags

**Tags** are collections of strings that can be attached to runs. They are used to categorize runs and make it easier to search for them in the LangSmith UI. Tags can be used to filter runs in the LangSmith UI, and can be used to group runs together for analysis. [Learn how to attach tags to your](#)

## traces

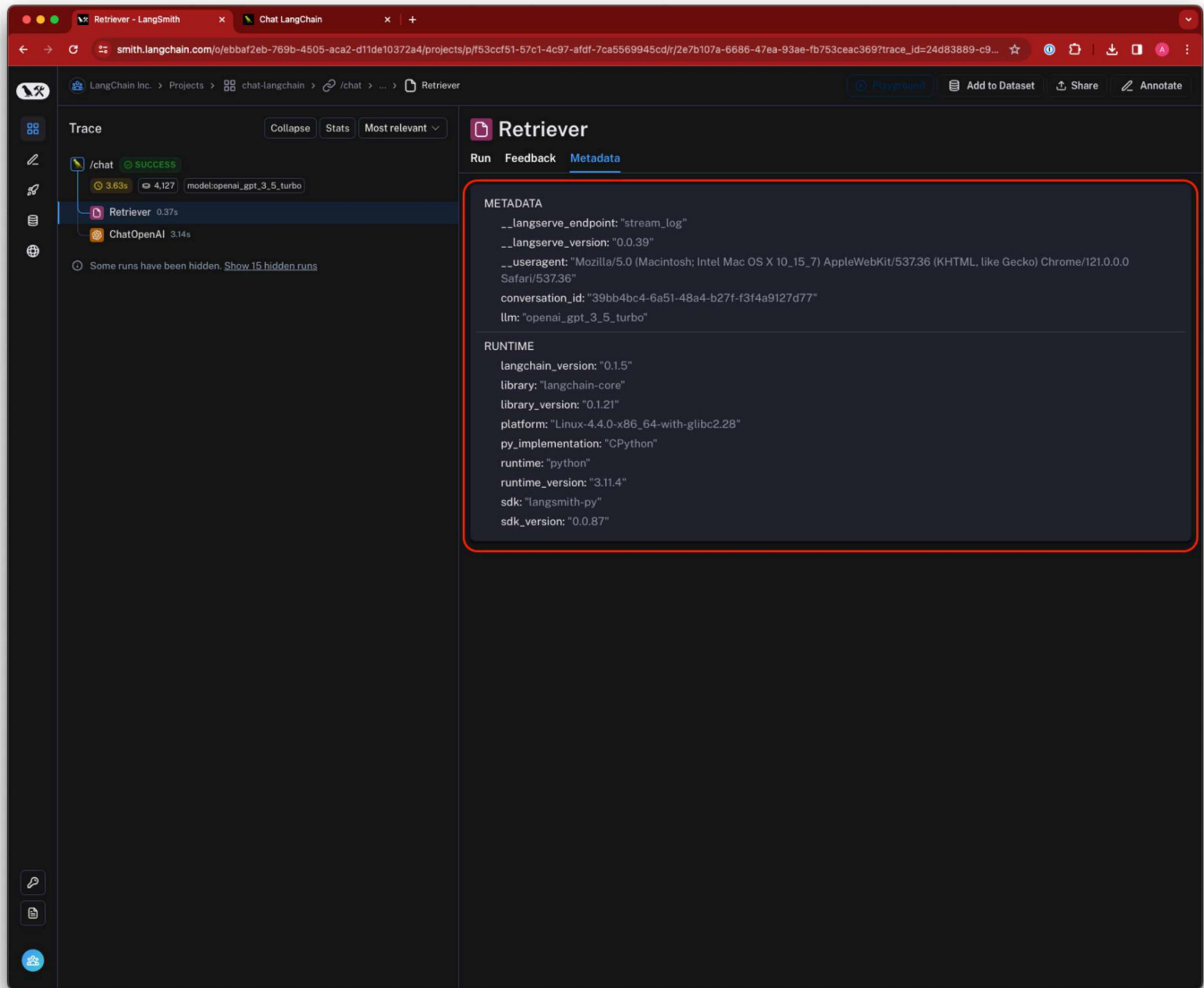


The screenshot displays the LangSmith interface for a 'Retriever' component. The left sidebar shows a trace for the '/chat' endpoint, which is successful. The main panel shows the 'Retriever' component's input and output. The input is a JSON object with a 'query' field. The output is a list of documents. The right sidebar shows the run's metadata, including start and end times, status (Success), total tokens (0), latency (0.37s), and type (Retriever). The 'TAGS' section is highlighted with a red box, showing 'Weaviate', 'seqstep:2', 'model:openai\_gpt\_3\_5\_turbo', and 'OpenAIEmbeddings'.

## Metadata

**Metadata** is a collection of key-value pairs that can be attached to runs. Metadata can be used to store additional information about a run, such as the version of the application that generated the run, the environment in which the run was generated, or any other information that you want to associate with a run. Similar to tags, you can use metadata to filter runs in the LangSmith UI, and can

be used to group runs together for analysis. [Learn how to add metadata to your traces](#)



## Data storage and retention

For traces ingested on or after Wednesday, May 22, 2024, LangSmith (SaaS) retains trace data for a maximum of 400 days past the date and time the trace was inserted into the LangSmith trace database.

After 400 days, the traces are permanently deleted from LangSmith, with a limited amount of metadata be retained for the purpose of showing accurate statistics such as historic usage and cost.

### NOTE

If you wish to keep tracing data longer than the data retention period, you can add it to a dataset. A dataset allows you to store the trace inputs and outputs (e.g., as a key-value

dataset), and will persist indefinitely, even after the trace gets deleted.

## Deleting traces from LangSmith

If you wish to remove a trace from LangSmith sooner than the expiration date, LangSmith supports deleting traces via deleting a project.

This can be accomplished:

- in the LangSmith UI via the "Delete" option on the Project's overflow menu
- via the [Delete Tracer Sessions](#) API endpoint
- via `delete_project()` (Python) or `deleteProject()` (JS/TS) in the LangSmith SDK

LangSmith does not support self-service deletion of individual traces at this time.

If you have a need to delete a single trace (or set of traces) from LangSmith project before its expiration date, please have your account owner reach out to [LangSmith Support](#) with your organization ID and trace IDs.

Was this page helpful?



You can leave detailed feedback [on GitHub](#).