Building and deploying LLM applications with Apache Airflow

ASTRONOMER

Julian LaNeve
Senior Product Manager @ Astronomer

Kaxil Naik
Apache Airflow Committer & PMC Member
Director of Eng @ Astronomer

Agenda

Why Airflow should be at the centre of LLMOps?

Real Use-case & reference architecture

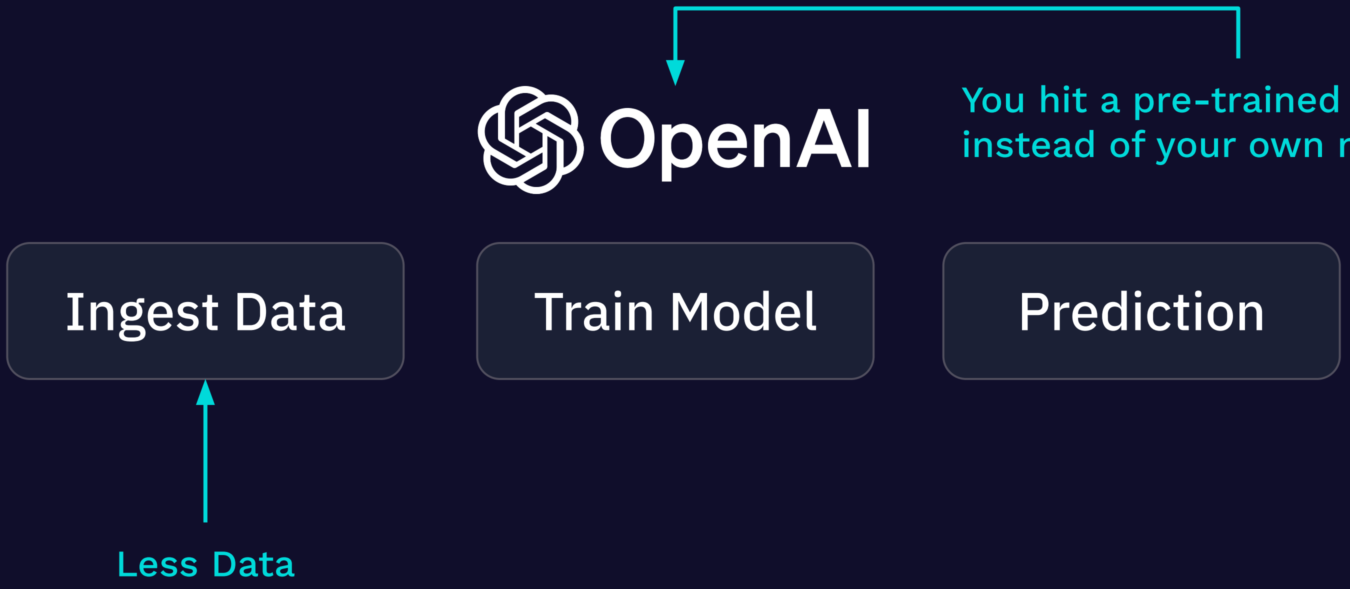
Next Steps: Community collaboration

Generative AI:
A Creative New World

A powerful new class of large language models is making it possible for machines to write, code, draw, and create with credible and sometimes superhuman results.

Normally, for ML, you need to...

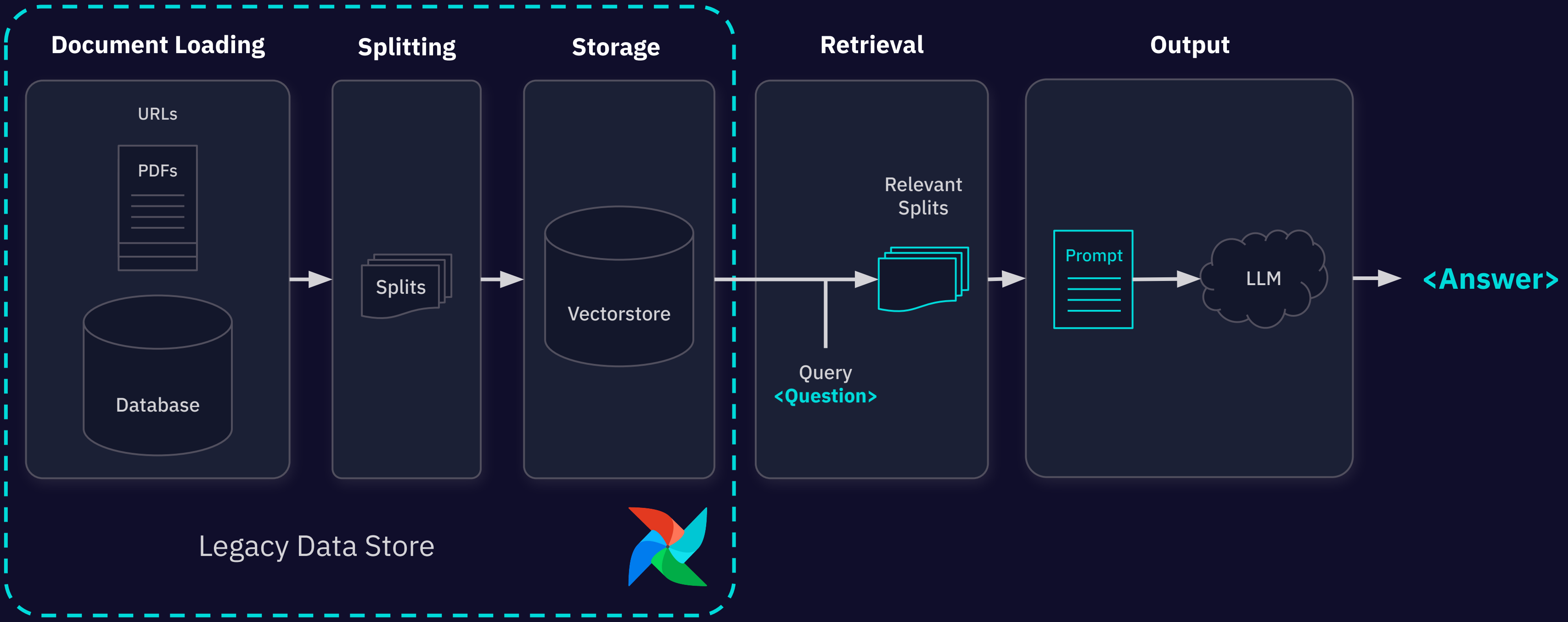
...but now you can:



Going from “Idea to Production” with LLM Apps involves solving a lot of data engineering problems:

- Ingestion from several sources
- Day 2 operations on data pipelines
- Data preparation
- Data privacy
- Data freshness
- Model deployment & monitoring
- Scaling Models
- Experimentation & fine-tuning
- Feedback Loops

Typical Architecture for Q&A use-case using LLM



Source: https://python.langchain.com/docs/use_cases/question_answering/

Airflow is a Natural Fit...

Python Native

The language of data scientists and ML engineers.

Common Interface

Between Data Engineering, Data Science, ML Engineering and Operations.

Document Parsing

Decorator and pythonic interfaces for standard LLM tools

Monitoring & Alerting

Built in features for logging, monitoring and alerting to external systems.

Extensible

Standardize custom operators and templates for common DS tasks across the organization.

Ingestion

Extract and load data into vectordbs and other destinations

Pluggable Compute

GPUs, Kubernetes, EC2, VMs etc.

Data Agnostic

But data aware.

Day 2 Ops

Handle retries, dependencies, and all other day 2 ops associated with data pipelines

Let’s Talk About a Real Use Case

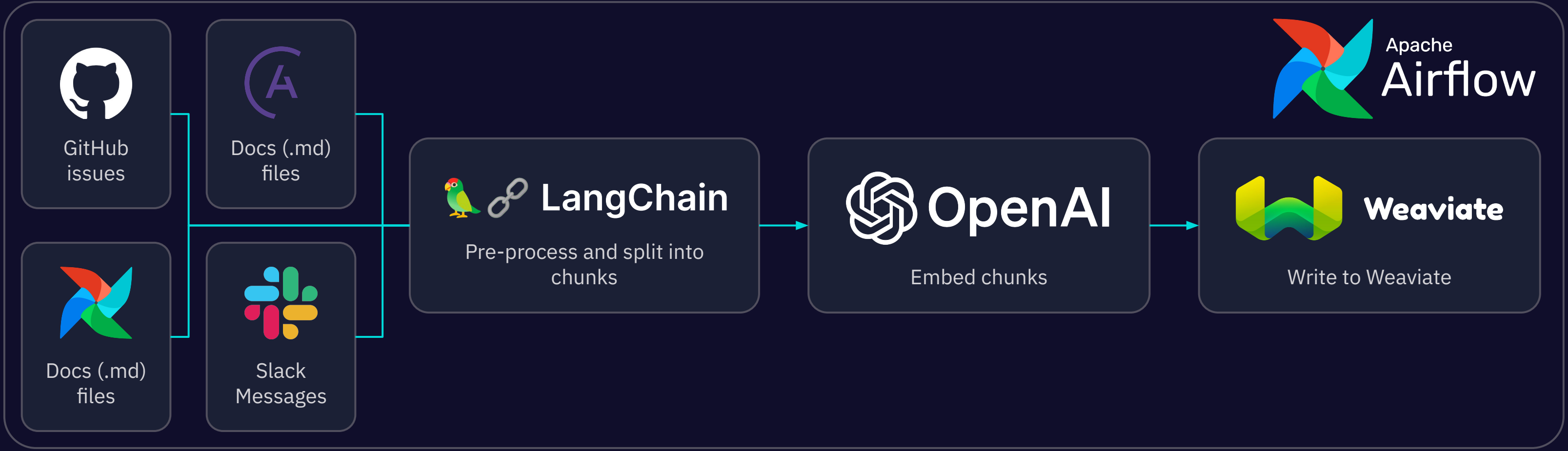
Problem Statement:

We have customers, employees, and community members that ask questions about our product with answers that exist across several sources of documentation.

How do we provide an easy interface for folks to get their questions answered without adding further strain to the team?

Ask Astro

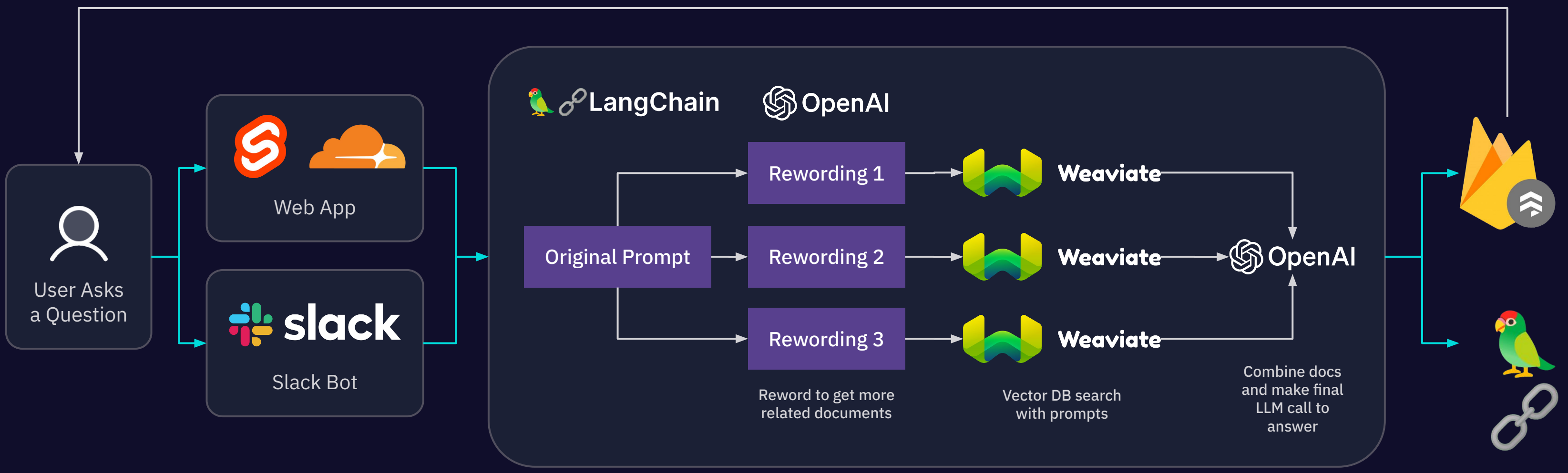
Data Ingestion, Processing, and Embedding



- Airflow gives a **framework to load data from** APIs & other sources into LangChain
- LangChain helps pre-process and **split documents into smaller chunks** depending on content type

- After content is split into chunks, each chunk is **embedded into vectors** (semantic representations)
- Those vectors are **written to Weaviate** for later retrieval

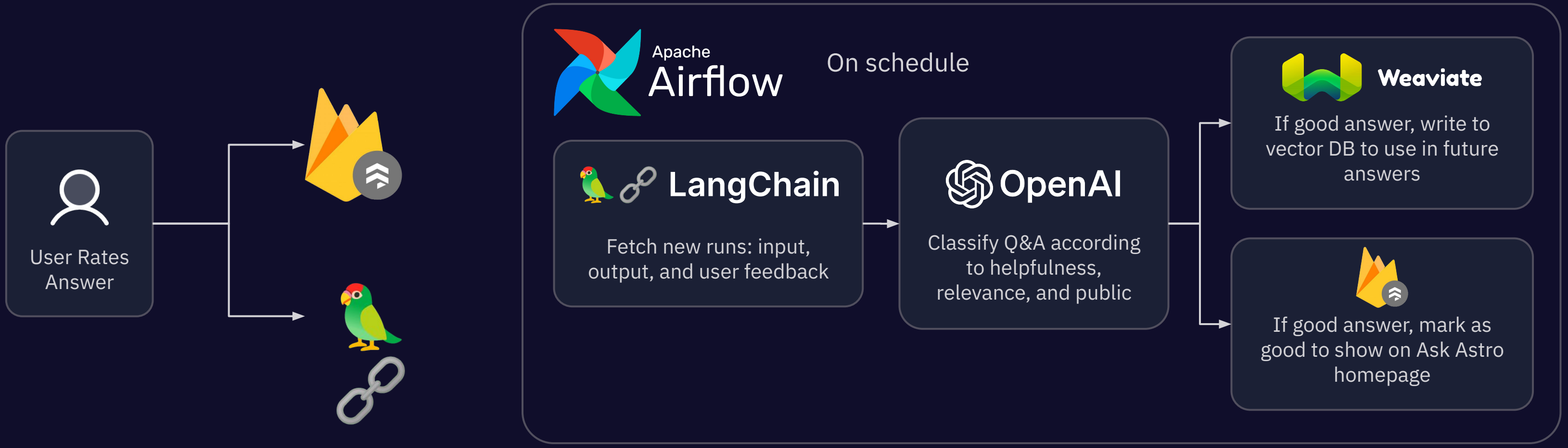
Prompt Orchestration and Answering



Users can interact with UI or Slack Bot; they both use the same API

- Original prompt gets reworded 3x using gpt-3.5-turbo
- Answer is generated by combining docs from each prompt and making a gpt-4 call
- State is stored in Firestore and prompt tracing is done through LangSmith

LLM & Product Feedback Loops



When a user submits feedback, it gets stored in Firestore and LangSmith for later use

- Airflow DAGs process feedback async to evaluate answers on helpfulness,, relevance, and publicness
- If answer is good, it gets stored in Weaviate and can be used as a source for future questions
- UI also shows the most recent good prompts on the homepage

Running this in production meant:

- Experimenting with different ***sources of data to ingest***
- Running the pipelines on ***a schedule and ad-hoc***
- Running the same workloads ***with variable chunking strategies***
- Needing to ***retry tasks*** due to finicky python libraries and unreliable external services
- Giving different parts of the workload ***variable compute***
- Creating standard interfaces to ***interact with external systems***

Running this in production meant:

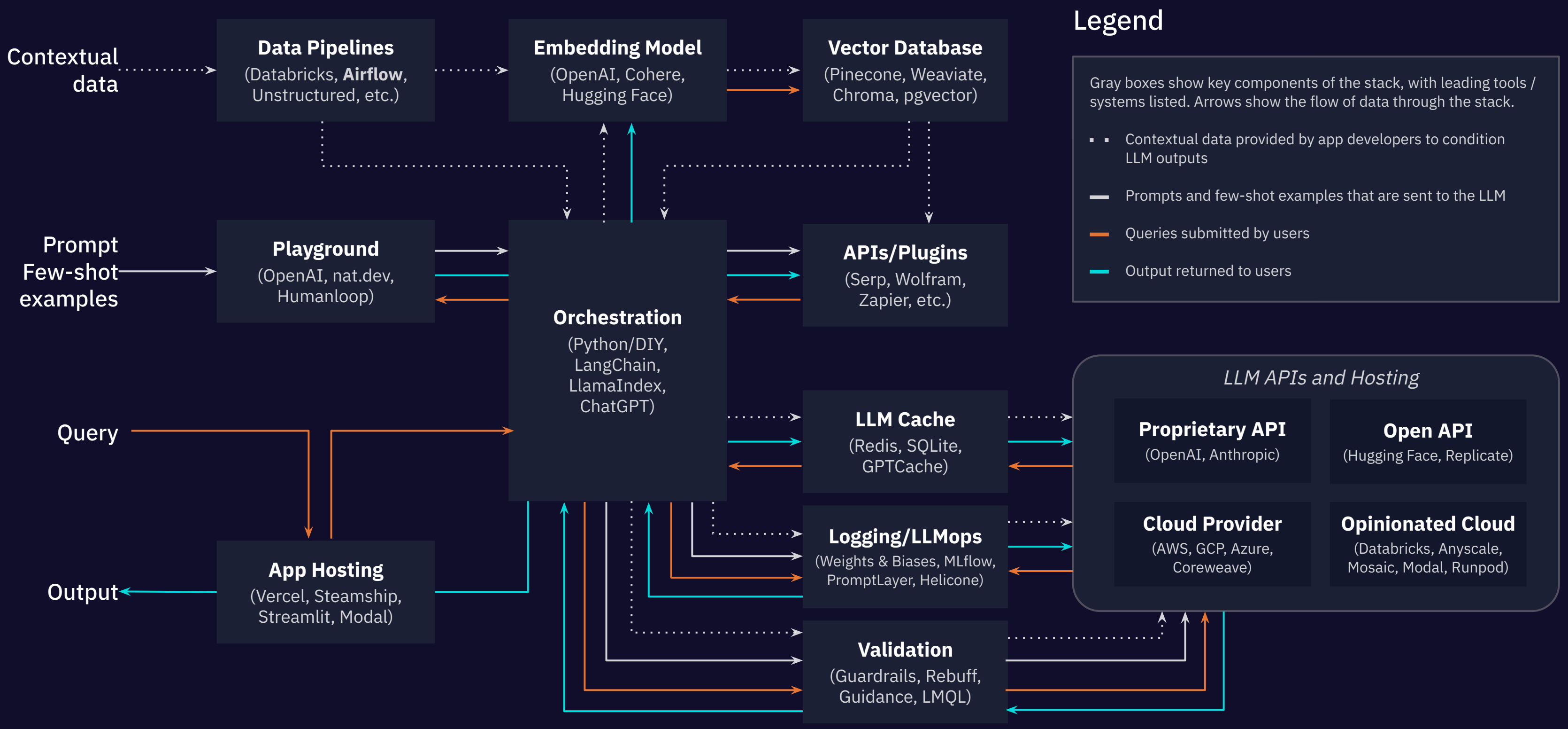
- Experimenting with different ***sources of data to ingest***
- Running the pipelines on ***a schedule and ad-hoc***
- Running the same workloads ***with variable chunking strategies***
- Needing to ***retry tasks*** due to finicky python libraries and unreliable external services
- Giving different parts of the workload ***variable compute***
- Creating standard interfaces to ***interact with external systems***

Which is what Airflow's great at!

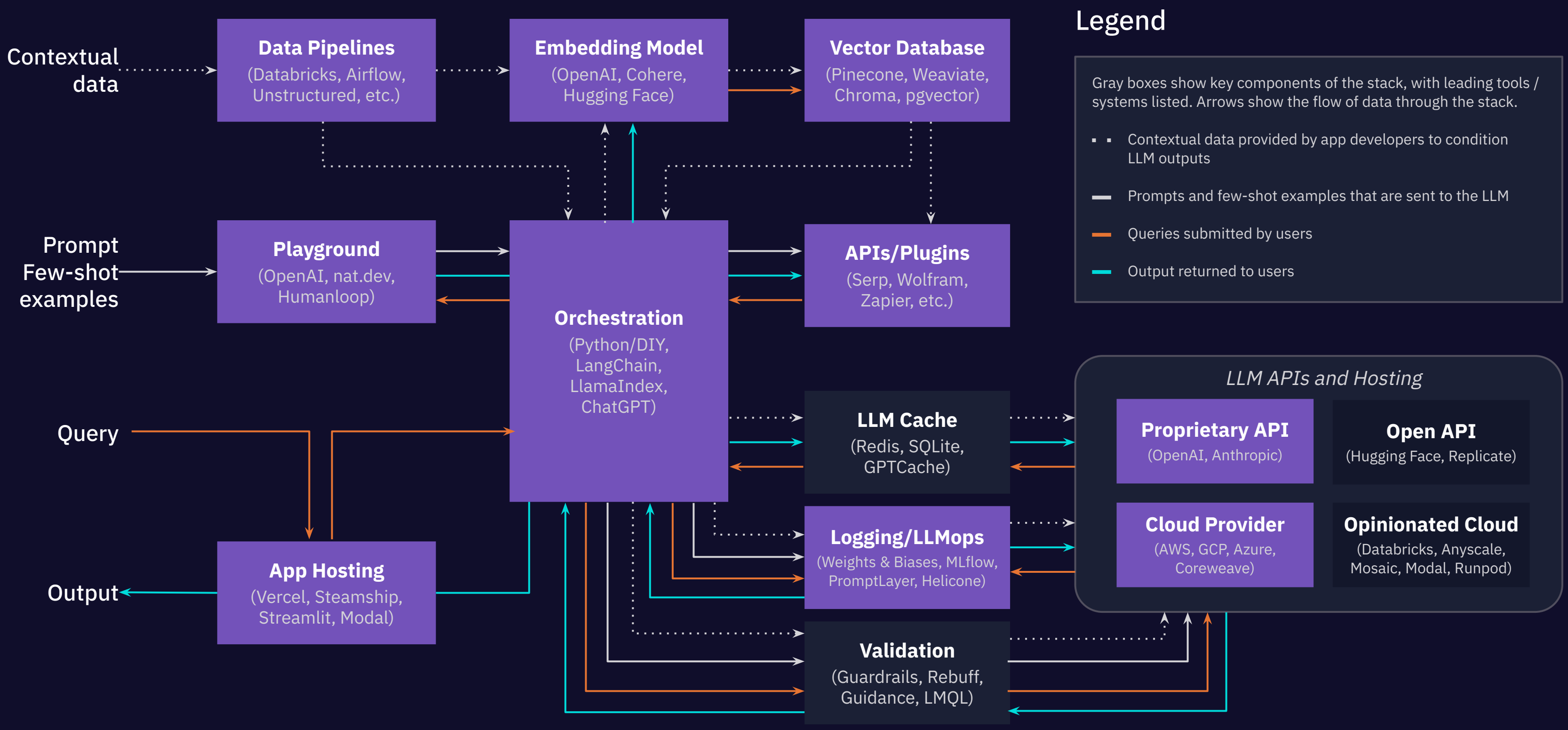
ask.astronomer.io

github.com/astronomer/ask-astro

a16z's Emerging LLM App Stack



AskAstro has a few parts of this...



…but there's even more to consider.

Data Governance

- How do you account for private data?
- How do you provide transparency into data lineage?

Fine Tuning

- Does it improve results?
- How much does it cost?

Feedback Loops

- Semantic cache for correct responses
- Ranking sources based on accuracy and ranking accordingly
- Prompt clustering – what are people asking?

Airflow is foundational to best practices for all of this.

Thanks to the AskAstro Team:

Philippe Gagnon

Michael Gregory

Community Collaboration

Providers

Interfaces

Patterns and Use Cases

What are all the providers the ecosystem needs?

Weaviate

pgvector

OpenAI

Dolly

Pinecone

What's the interface that feels right for LLMOps?

```
create_embeddings = OpenAIEmbeddingOperator(
    task_id="create_embeddings",
    conn_id="openai_prod",
    source_data="/usr/local/airflow/dags/github.pdf",
    output_file="/usr/local/airflow/data/embeddings.txt",
    model="text-embedding-ada-002",
    encoding="cl100k_base",
)

store_embeddings = WeaviateOperator(
    task_id="check_schema",
    conn_id="weaviate_prod",
    embeddings="/usr/local/airflow/data/embeddings.txt",
)
```

```
LlmOperator(
    task_id="openai_task",
    embedding="OpenAI",
    source_dataset=Dataset("/usr/local/airflow/dags/data/github.pdf"),
    target_dataset=Index(uri="pgvector://postgres", name="airflow_summit_test"),
    embedding_params={
        "embedder_model": "text-embedding-ada-002",
        "encoding_name": "cl100k_base"
    },
)
```

ASTRONOMER

What’s the interface that feels right for LLMOps?

```python
@task
def generate_and_store_embedding(data_path):
    import os

    from langchain.document_loaders import PyPDFLoader
    from langchain.embeddings import OpenAIEmbeddings
    from langchain.text_splitter import CharacterTextSplitter
    from langchain.vectorstores import Chroma

    assert os.environ["OPENAI_API_KEY"]
    loader = PyPDFLoader(file_path=data_path)
    pdf_docs = loader.load()
    text_splitter = CharacterTextSplitter.from_tiktoken_encoder(
        chunk_size=1000, chunk_overlap=200
    )
    documents = text_splitter.split_documents(pdf_docs)
    Chroma.from_documents(documents=documents, embedding=OpenAIEmbeddings())
```

Patterns

What are the best practices for building pipelines for LLM Apps?

- Do you use one task to ingest and write?
- Can you use dynamic task mapping to break it out?
- Do you write to disk?
- Can you store embedding values in XCOMs?
- How do you reconcile Airflow orchestration with prompt orchestration?

Let’s do this all in the open source!