Airflow and Spark for Contact Center AI

Turbocharging MLOps for Generative AI at ASAPP

Udit Saxena
ASAPP AI Engineering

ASAPP

Udit Saxena

Some background

- Building ML/AI platforms at ASAPP
- Working with Airflow for about half a decade now
- Previously at SumoLogic, on the modeling team
- AI/NLP (MS) from UMass Amherst, BE (CS/Math) from BITS Pilani
- @saxenaudit on Twitter/X
- Find this work on the official Apache Airflow blog

Agenda

01 A little bit about ASAPP

02 MLOps at ASAPP

03 Airflow for MLOps at ASAPP

04 ASR workflow

05 Spark Integrations

06 LLM-based solutions

07 Recap

ASAPP
Some background

- Building AI solutions for Contact Centers, for over a decade
- Generative AI has been disruptive in the Contact Center space
- AI-powered tools:
    - Generative Agent
    - AutoSummary
    - AutoTranscribe
    - AutoCompose
- Big proponents of Airflow for MLOps for our tools

MLOps at ASAPP

- Continuous Improvement
- Diverse Data Processing
- Scalability for Changing Technologies

This is where Airflow comes in …

Airflow at ASAPP

Data Engineering

Data Ingestion and Preprocessing

DataOps

Managing data across multiple Machine Learning workflows

MLOps

Development, Evaluation and Monitoring of ML models

Airflow at ASAPP: Over time

Pre-version 2.x

Highly customized Airflow for stability, coupled DAGS, deployment not scalable

Version 2.3.x

Improved out-of-the-box stability, reliability, RBAC, User management

Fewer custom solutions

Git-Sync for DAGs

Allows scaling up deployments, faster dev iterations, deeper K8S integration

Strong MLOps support

Supporting operations around ML model lifecycle, development, evaluations, deployment

Multiple Deployments

Support different Airflow clusters for different applications

Spark Integrations

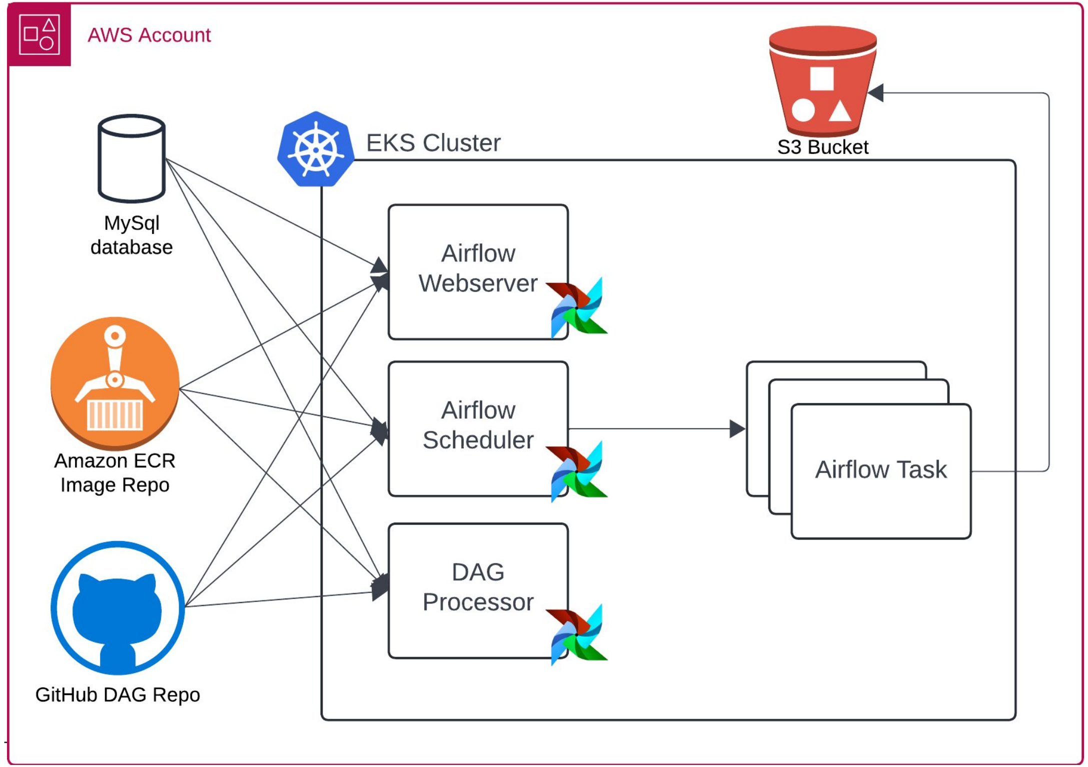
Adding Spark based workflows for improving processing times of ML pipelines

Batch processing

Support offline batch pipelines for increasing throughput of ML pipelines at efficient cost

Airflow at ASAPP

- Primary MLOps orchestration tool since 2020
- Git-sync deployment pattern
- Airflow scheduler uses KubernetesExecutor to schedule task pods
- Task pods are almost always use KubernetesPodOperator
- All images are stored in AWS ECR



Data Ingestion and Preprocessing

- Data is first ingested into data lakes via real-time Spark applications and Golang applications
- Processed, computed, cleaned, and sorted before being distributed to various sinks
- Over a million Airflow tasks daily across more than 5,000 Airflow DAGs

DataOps: Managing data lifecycle

Data Retention Policy Enforcement

- Periodically check and enforce retention policies across diverse data sources
- Makes it easier and less error-prone over manual management

Production Data Sampling:

- Scheduled DAGs run periodically to collect and process production data for various downstream applications eg transcribing audio for MLOps workflows

MLOps with Airflow

Model training/fine tuning

Support a variety of model types (small/ medium sized models, LLMs) across applications – Speech, NLP.

KubernetesExecutor + KubernetesPodOperator is the default task pattern here.

Orchestrate experiment pipelines, periodic updates.

Model monitoring/ evaluation

Frequent monitoring of models in production or currently being evaluated through S3 triggers, API calls, scheduled jobs.

Subsample production data.

Deep integration with Observability tooling through Prometheus and Grafana.

ML Engineering

Complex pipelines used for internal development use cases for offline application support.

Used both by our NLP and Speech team.

~~MLOps~~ LLMOps with Airflow

Popular LLM provider support

Pipelines which use AWS Bedrock, OpenAI, Anthropic, internal LLMs (hosted through vLLM and TGI endpoints)

Redaction-as-a-service for privacy

LLM monitoring/ evaluation

LLM-as-a-judge for evaluations

Pipelines for custom, quick “vibe” based evaluations

Monitoring of critical LLM request metrics via Grafana: cost, latency, retries, other evals/assertions, token counts

~~ML~~ LLM Engineering

Support LLM artifacts - Prompt management and versioning

Prompt optimization iteration through retroactive testing and analysis

One-off workflows to support a diverse LLM dev/ engineering landscape

Airflow at ASAPP: Key learnings

01

Airflow doesn't stand alone

A team well versed in Infrastructure, Platform and Machine Learning can take it far

02

Airflow abstractions are powerful

Use Airflow as an orchestrator. Abstractions and integrations allow Airflow to scale well

03

Git-Sync allows great DevEx

Fast development iterations, great UX.

04

Improved stability after v2.3.x

Strong and mature open source community has made steady progress.

Agenda

01 ~~A little bit about ASAPP~~

02 ~~MLOps at ASAPP~~

03 ~~Airflow for MLOps at ASAPP~~

04 ASR workflow

05 Spark Integrations

06 LLM-based solutions

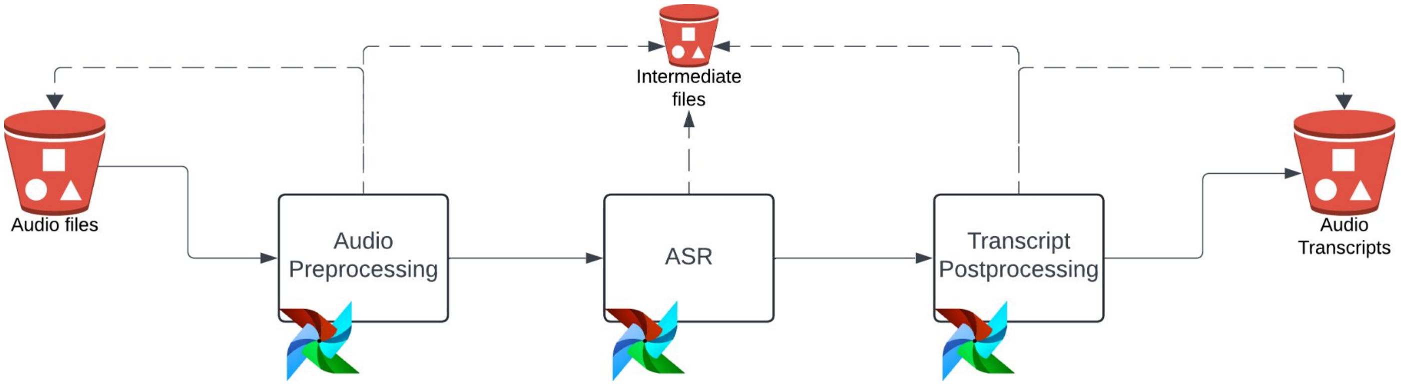
07 Recap

Case Studies

- An Automatic Speech Recognition workflow
- Integrating Spark for improving ASR processing times
- (quick aside) LLM workflows

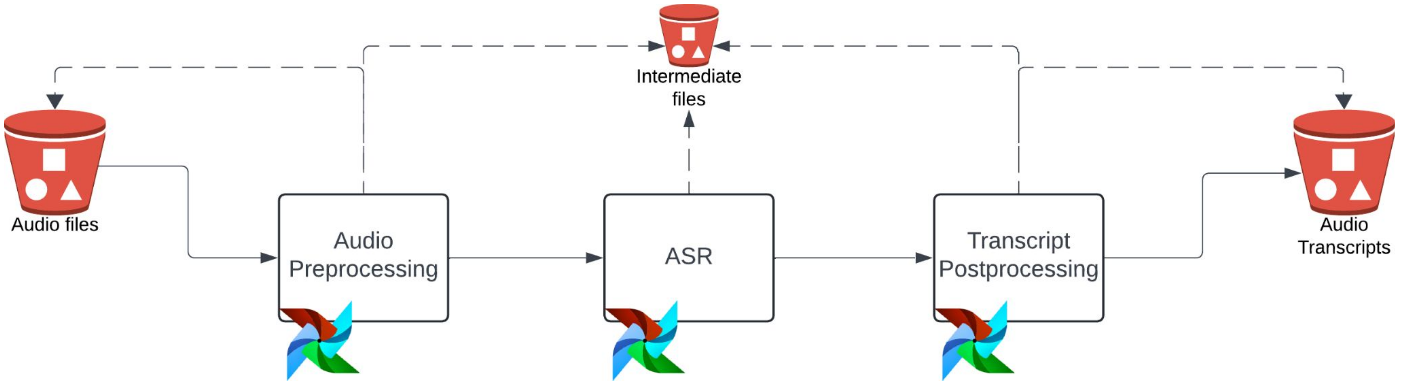
Automatic Speech Recognition

- Our Speech team uses Airflow to transcribe current and historical audio data for offline downstream pipelines.
- A sample workflow consists broadly of three groups of sub-tasks:
    - Preprocessing the audio files
    - Transcription: Automatic Speech Recognition (ASR) using proprietary models
    - Post Processing the resulting transcripts



Automatic Speech Recognition: In depth

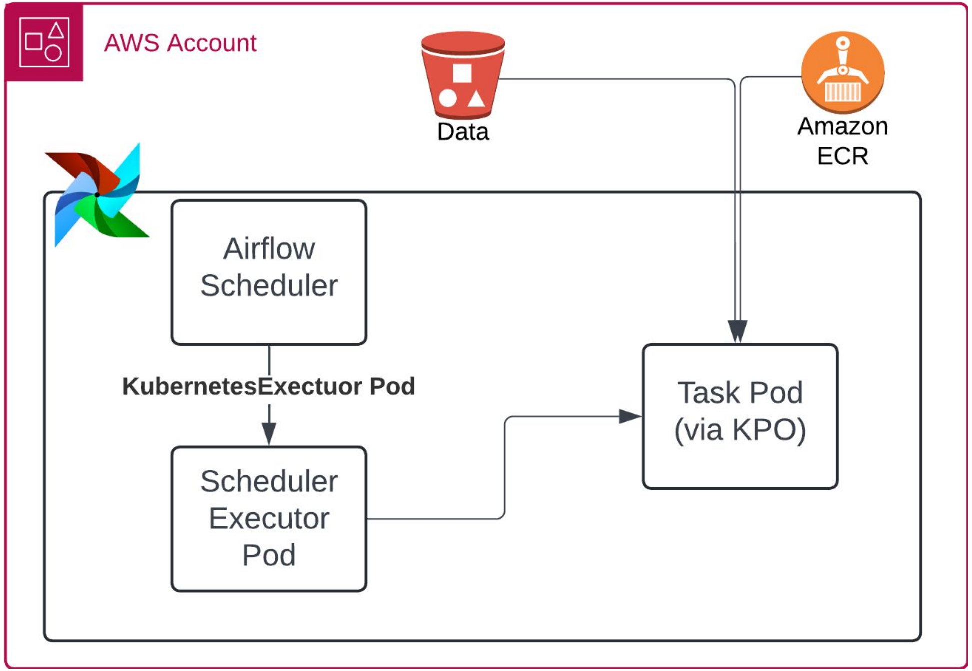
- Each task maps to an Airflow task using a KubernetesPodOperator
- Data plane: S3 bucket with audio files, transcripts, intermediate data
- We need to scale for increased audio corpora, different audio input formats, different ASR runtimes, and different output transcript schema
- Current workflow processing times won't scale, eg a single ASR workflow takes 40+ hours to complete for a reasonable workload



Automatic Speech Recognition: In depth

Looking at a single Airflow task

- The Airflow Scheduler runs an ASR Task pod using the KubernetesPodOperator
- The task pod pulls an image from ECR
  - Each stage has its own image
- In an ASR pipeline, data could be audio files, chunked audio clips, text transcripts etc



Why Spark?

Scalability

- Processing capabilities can grow as data demands increase.
- Easy to configure parallel workers (and the resources for each).

Development Experience

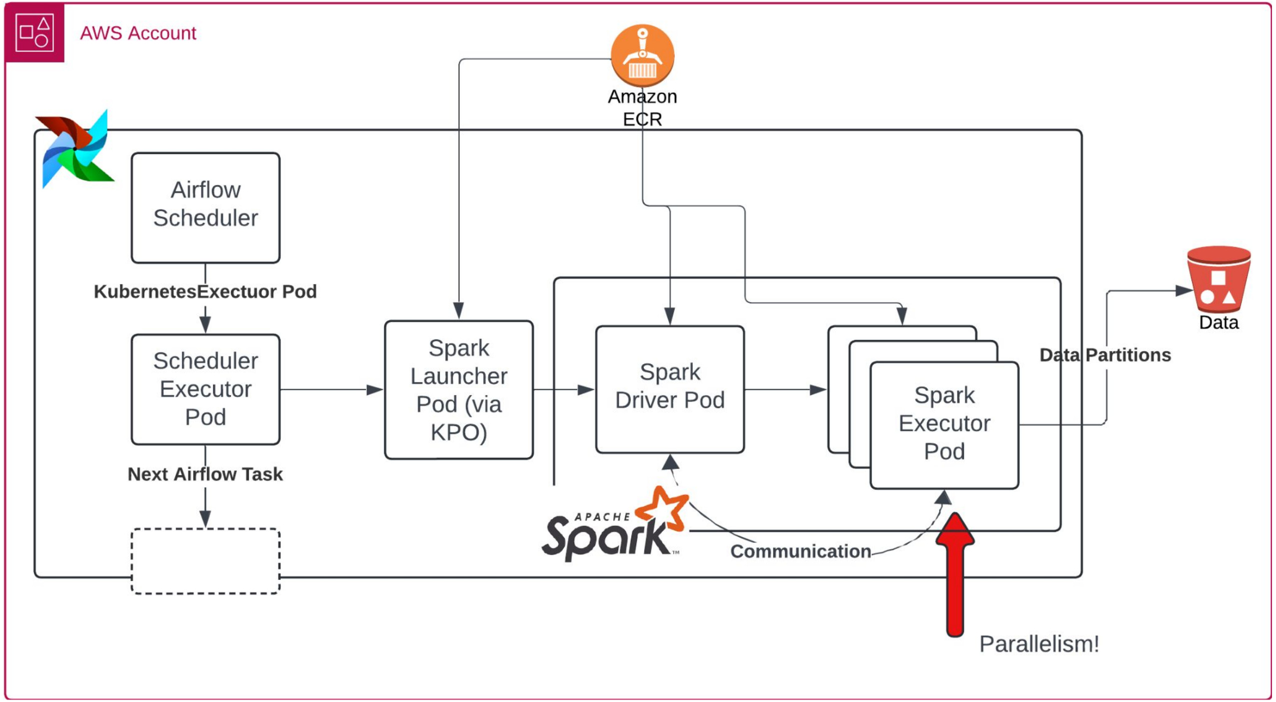
- Intuitive development of data pipelines: dataframe centric. Not just running scripts.

Context

- We already have Airflow DAGs, offloading heavy data processing to Spark was a reasonable transition.

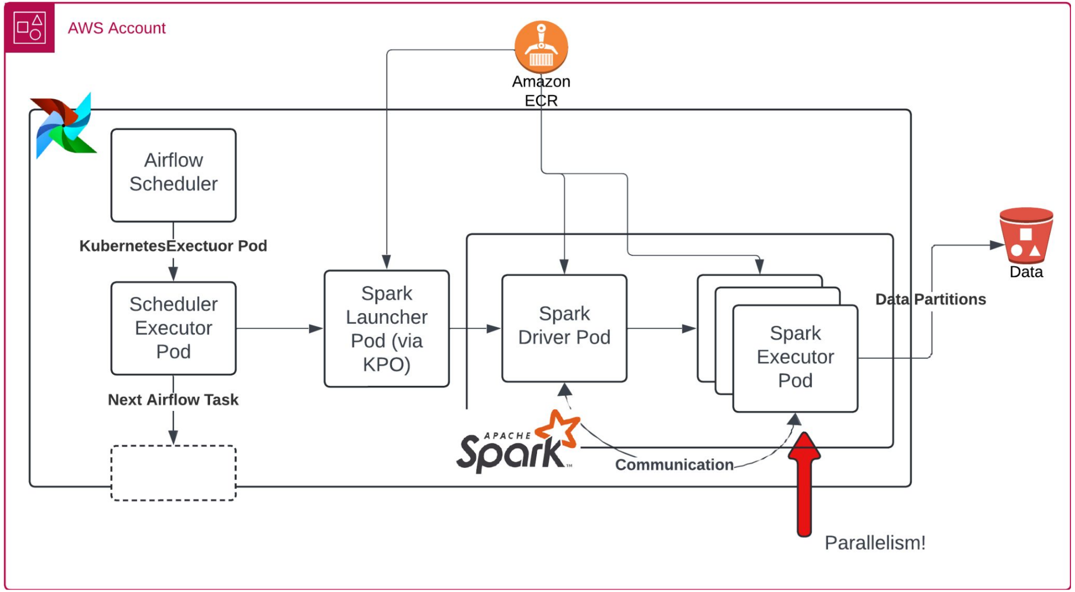
Integrating Spark with ASR

- Instead of managing one heterogenous Spark cluster, use Airflow to launch an on-demand spark cluster for each DAG!
- Leverage Spark's integration with Kubernetes
- The driver pod interacts with the Airflow Task Pod as well as the executor pods.
- The spark executor pods work on the partitioned data
- The cluster uses the K8S API to communicate



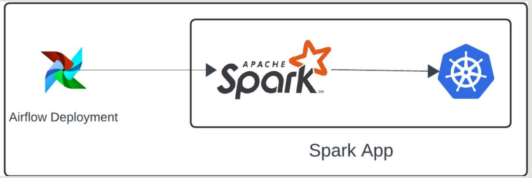
Integrating Spark with ASR

- **Advantages:**
  - Simplified management: Airflow is only orchestrating; Spark is managing the cluster
  - Flexibility: The executor pods can be customized for different workloads and are only bounded by the underlying manifest
  - Scalability: Spark does the heavy-lifting
- **Disadvantages:**
  - Initial setup cost; better after defaults are set up
  - More involved troubleshooting; you need to know your application requirements well
  - DAG writing for high performance workflows now requires knowledge of Spark



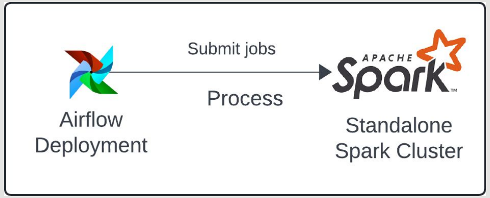
What's the difference?

Our operator



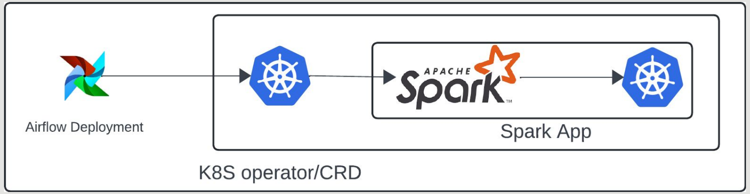
- Spark Native using Spark scheduler
- One level of indirection
- Extremely flexible

KubernetesSpark Operator



- Need to manage a standalone spark cluster
- Heterogeneous workloads make this unwieldy

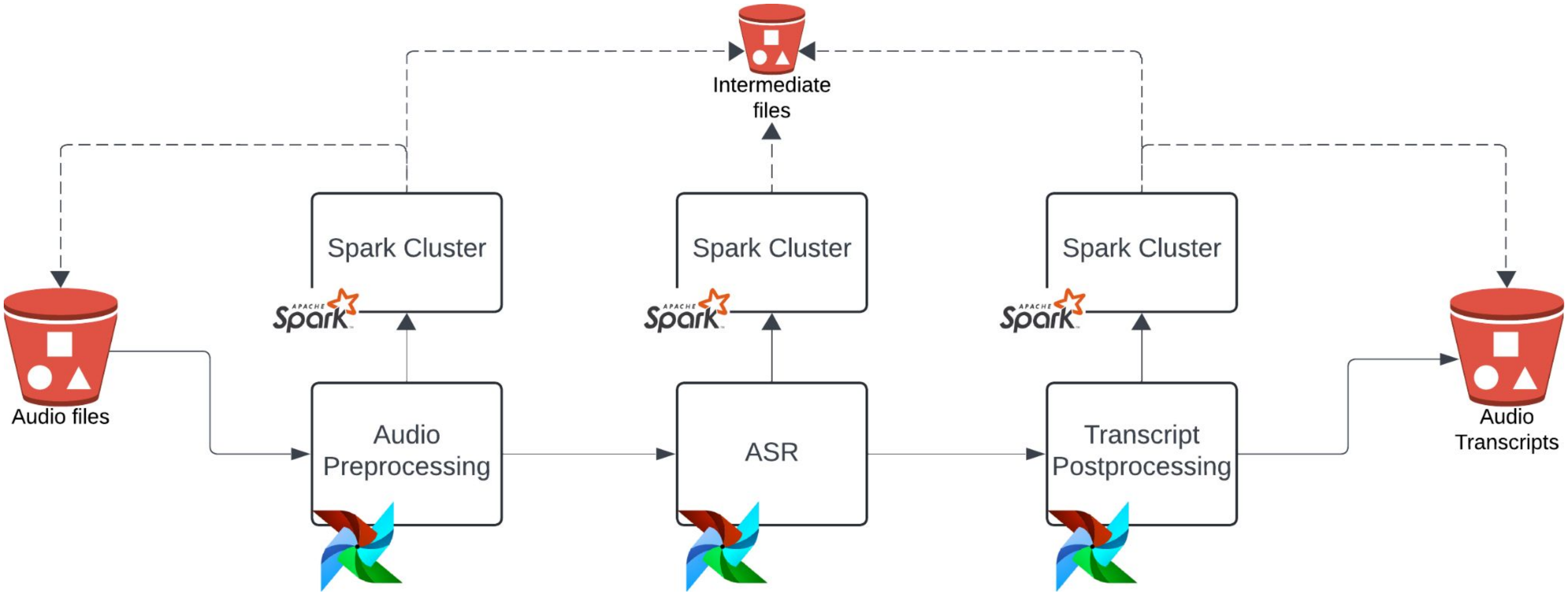
SparkOperator



- Kubernetes Native using K8S operator
- Many levels of indirection
- Not flexible

Integrating Spark with ASR

- In the new workflow, each Airflow task can now launch its own Spark cluster



Results

- We see an **order of magnitude improvement in runtime durations** for this workflow
- Earlier it took **40+ hours**; now the workflow takes **<5 hours**
- Further optimizations possible
- With Airflow orchestrating this workflow, we can explore pareto improvements with a task-appropriate optimal Spark configuration.

LLMs with Airflow

- Similar workflows can be used for LLMs with Airflow tasks
- If LLMs are consumed through endpoints, use API calls through the KubernetesPodOperator
- If consuming LLMs by manual deployment and custom inference, consider scheduling Spark executor pods on GPU nodes with models hosted locally

Spark with Airflow: Key learnings

01

Airflow doesn’t stand alone

Multiple perspectives: Spark as an ML platform, deep Infrastructure and Airflow integration

02

Airflow abstractions are powerful

Just an orchestrator here; heavy lifting done through the KubernetesPodOperator and Kubernetes API integrations

03

S3 is the great equalizer

Boring, reliable, and scalable storage can go a long way before you hit performance plateaus

04

Improved stability after v2.3.x

Can use a lot of the new features in conjunction; eg TaskPools, Slots to control load on cluster

Recap and closing remarks

Airflow as a versatile MLOps/LLMOps tool

Synergy between Airflow, Spark, and LLMs

Continuous innovation in AI workflow management

Thank You