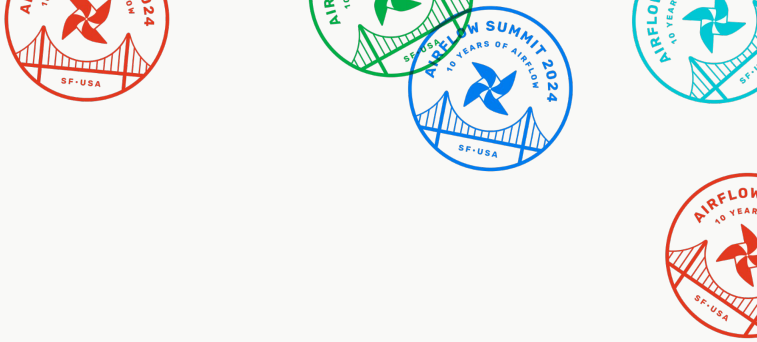
Scalable Development of Event Driven Airflow DAGs

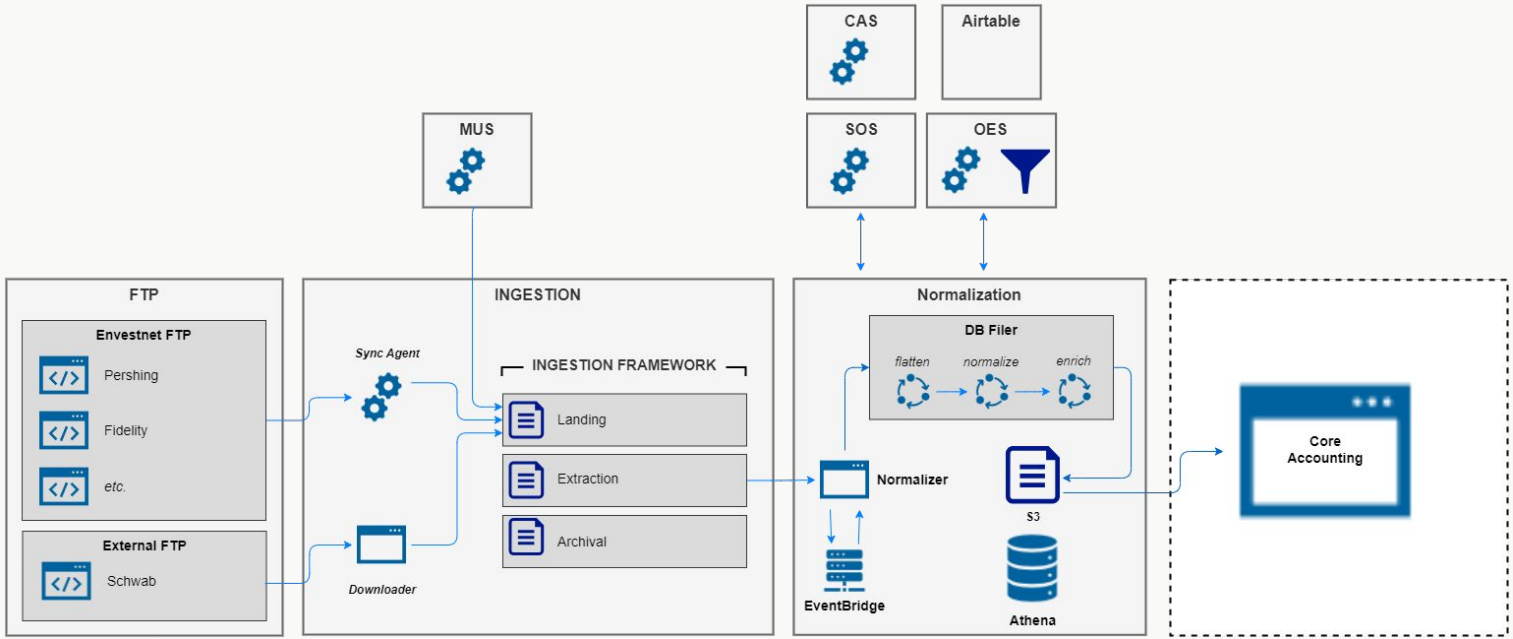
Subramanian Vellaiyan
Data Engineering
ENVESTNET

Ipsa Trivedi
Data Engineering
tekmetric
(Previously, Envestnet)



Use Case

Data Source Onboarding



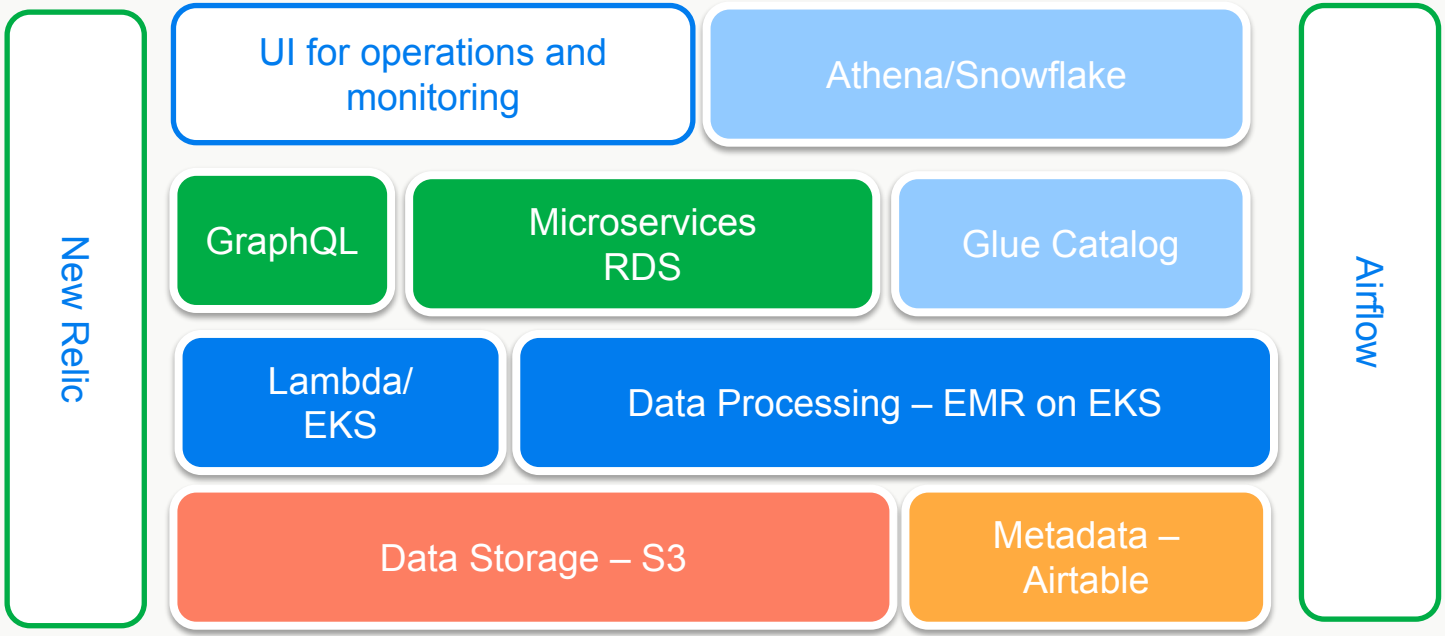
Data Source Onboarding

- Onboard data from different sources
  - Batch based and event-based sources
  - Files of different formats, types, size, layout, arrival schedules
- Standardize into a common data model
  - Source specific and source agnostic business logic
  - Metadata management
  - Integration with internal systems
  - Data mapping and transformation
- Deliver data to stakeholders within SLA
  - Filter out non-specific data
  - Process and deliver within 20 minutes of arrival
  - Data validation and reconciliation

Requirements and Complexity

- Variety:
  - Sources have different file formats, types, layouts, sizes, file arrival patterns
- Volume:
  - 200 sources, 100k's of files.
  - Complete daily processing terabyte scale
- Domain Expertise:
  - Source data needs to be mapped and transformed to standard data model. Analysts have gained knowledge over the decade.
- Daily Ops:
  - Accurate and timely monitoring to run, debug & rerun before SLA
- Processing:
  - SLA of 20 mins. Complex multi-step processing as soon as any file dependency is met.

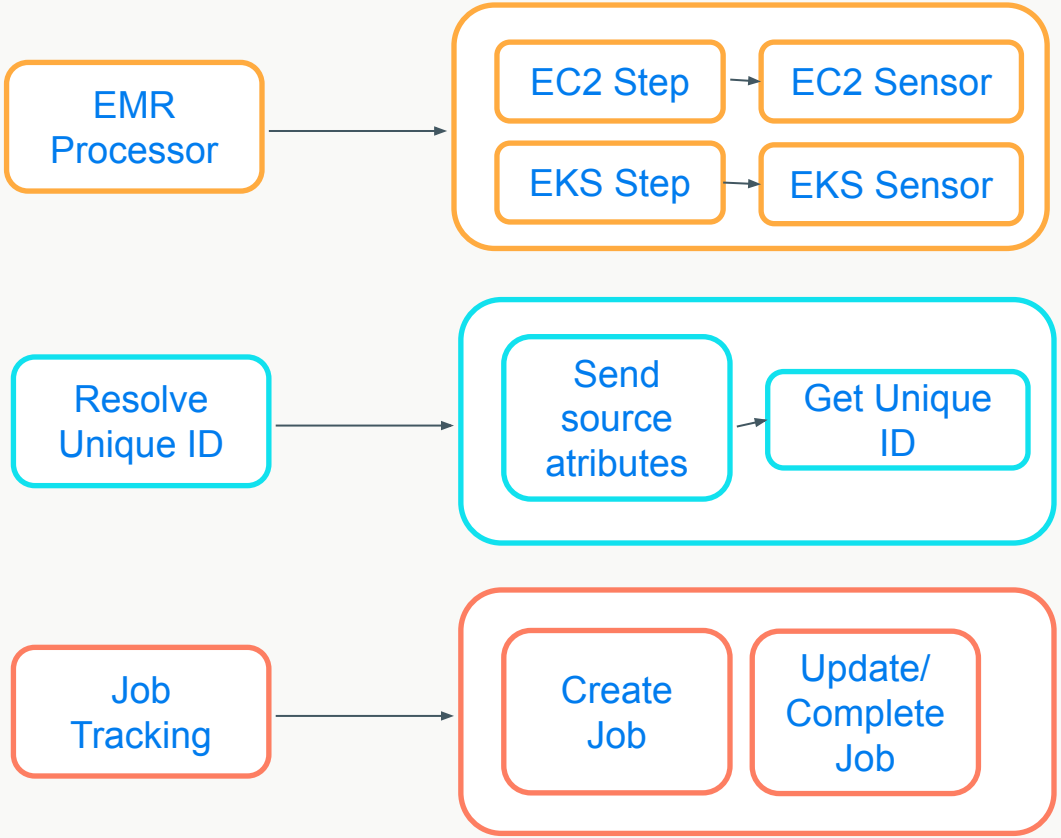
Technology Stack

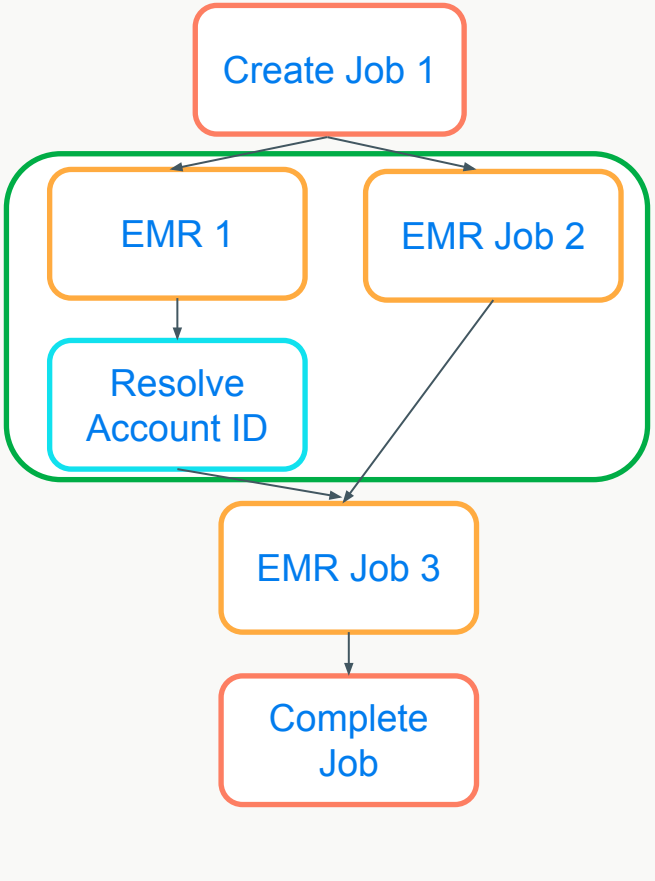


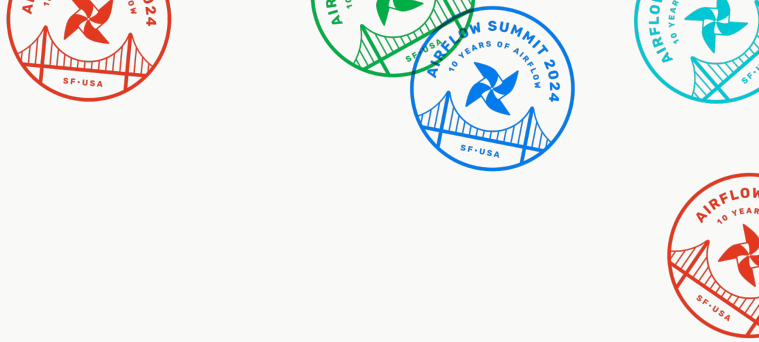
Why Airflow?

- Flexible Development
  - Scalable DAG building
  - Programmatic approach, developer friendly
  - Dynamic pipeline creation
- Easy Integration
  - Easily integrate with any system
  - Ability to develop custom functionality by extending existing functionality
- Intuitive Monitoring
  - Easy to execute, debug and monitor pipelines
  - Restarting a DAG from failure point from middle of a pipeline in case of errors
- All in One Visualization
  - Convenient dependency management between tasks
  - Visualize complex orchestration in a single view. Consolidate multiple tasks in groups.

Scalable DAG: Framework





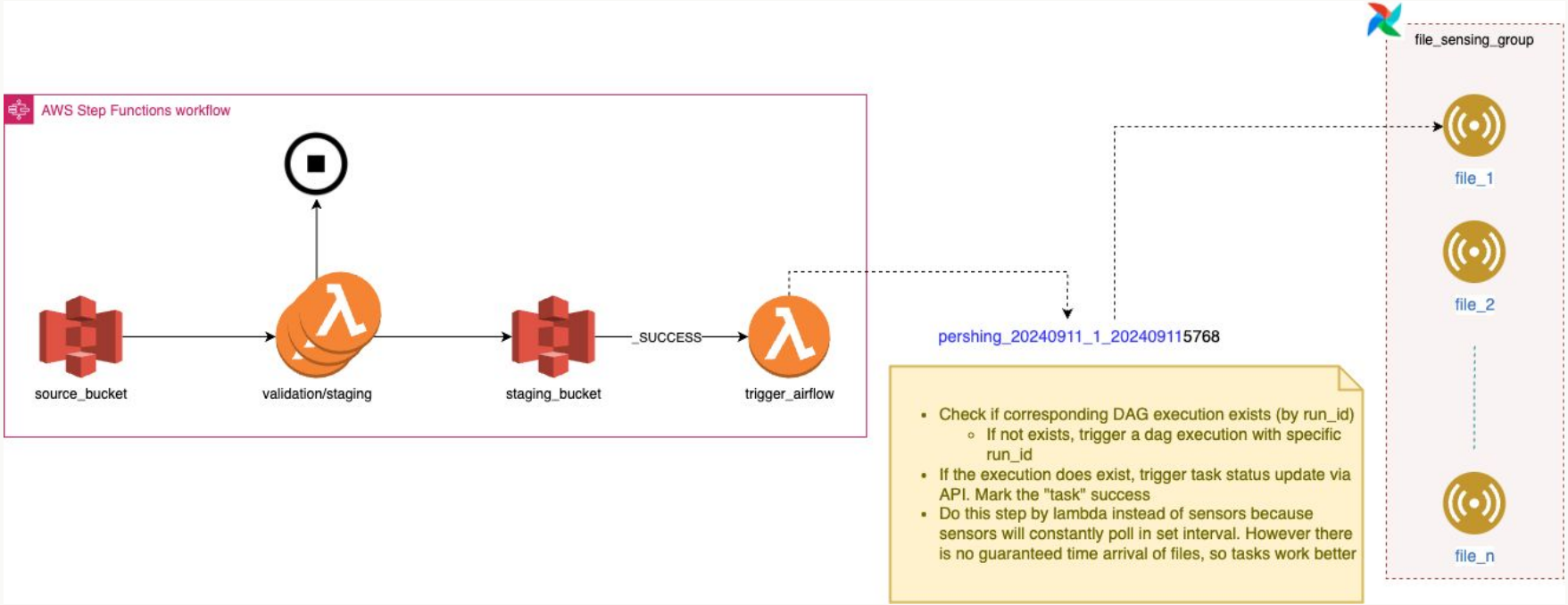


Event Driven DAG Handling

Creating Event Driven DAGs

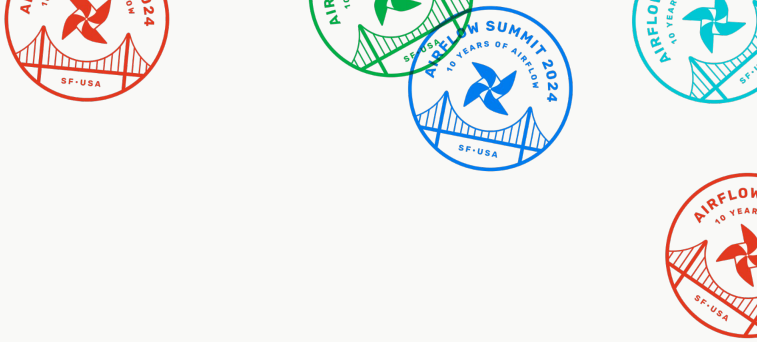
- One DAG per source to allow monitoring per source
- Setup:
  - A configuration driven AWS Step Function that validates the files, stages them and triggers Airflow of the respective source in an event-based manner
  - Source specific configs driven by custom DAG creation modules to create DAGs automatically
  - Source and file specific & agnostic configs driven by custom spark application that contain business logic
- Airflow:
  - file_sensing task group:
    - Tasks corresponding to each file in a source
  - file_processing task group:
    - Tasks to handle file processing via API calls and spark jobs

Triggering Event Driven DAGs



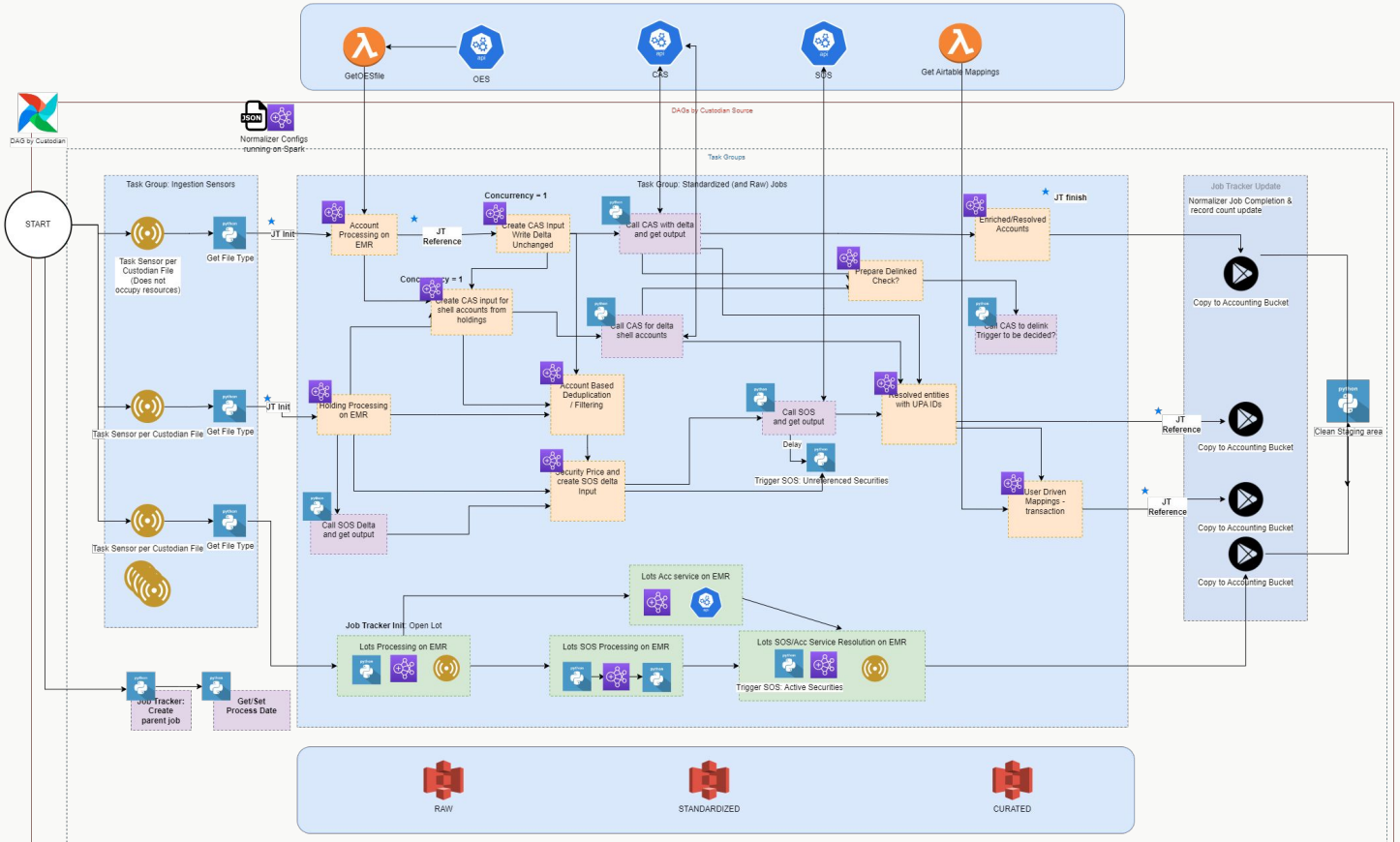
Triggering Event Driven DAGs

- Run IDs of the type: <source>_<file-date>_<set_code>_<run-ts>
- Check if an execution of that source, file_date, set_code, run_date exists or not
- If exists:
  - Mark the task for that file success via API call
- If not exists:
  - Create an execution with defined run_id of <source>_<file-date>_<set_code>_<ts>
  - Mark the task for that file success via API call
- All the “source” level information is passed to the DAG at conf level when triggering the DAG
- All the file level information is passed to the DAG via _SUCCESS file



Scalable Development

Simplified Architecture Diagram

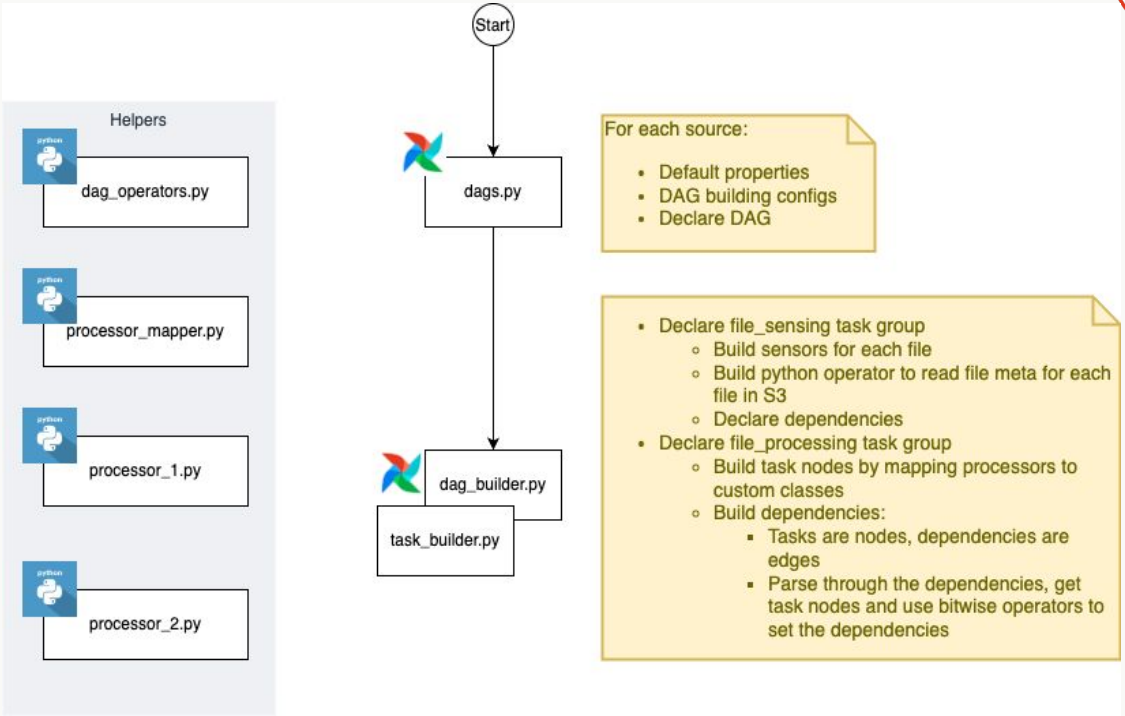


Scalable DAG: Config Building

Dependency Management Config

```
"Call API 1": {
      "processor": "call_api_1",
      "label": "Call API to get output in S3 bucket",
      "task_group": "file_processing",
      "dependencies": {
         "file_sensors": [
            "file1",
            "file2"
         ]
      }
   },

   "Submit Spark Job": {
      "processor": "submit_spark_job",
      "label": "Submit Spark Job for <source>",
      "task_group": "file_processing",
      "dependencies": {
         "file_processing": [
            "call_api_1"
         ]
      }
   }
```



Scalable DAG: Config Building

Dependency Management Config

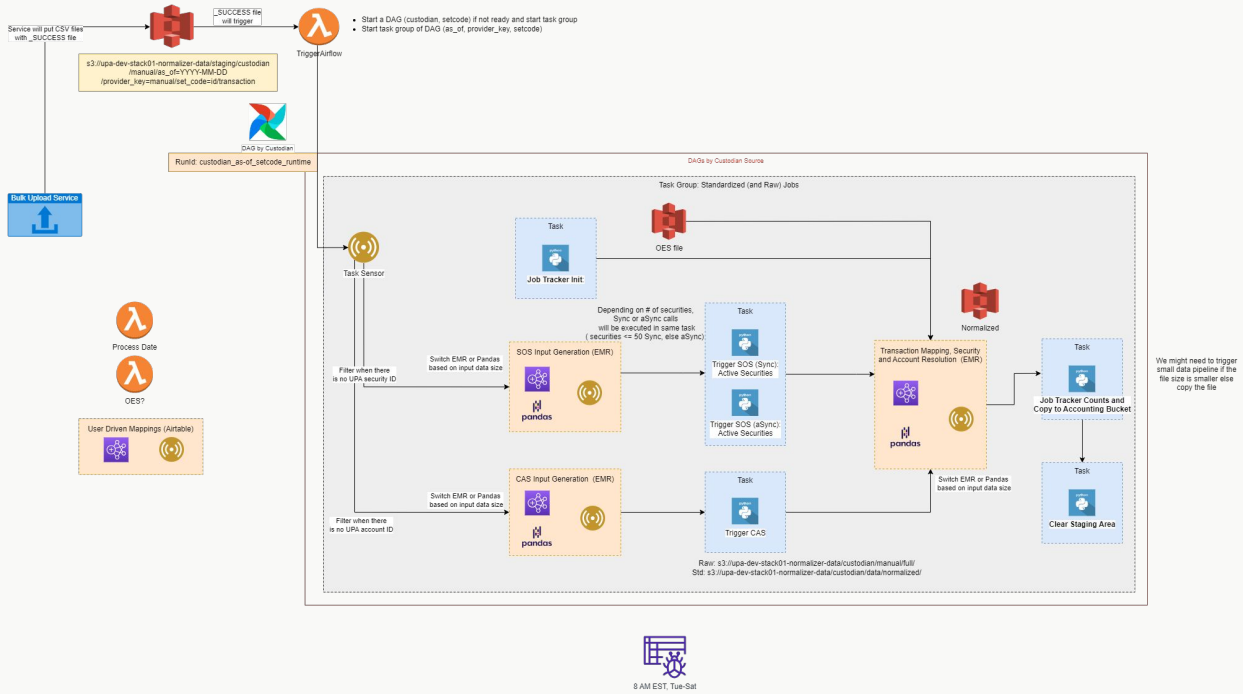
```
" call_api_1 ": {
      "processor": “API",
      "label": "Call API to get output in S3 bucket",
      "task_group": "file_processing",
      "dependencies": {
        "file_sensors": [
           "file1",
           "file2"
        ]
      }
   },

   " submit_spark_job ": {
      "processor": “EMR",
      "label": "Submit Spark Job for <source>",
      "task_group": "file_processing",
      "dependencies": {
        "file_processing": [
          "call_api_1"
        ]
      }
   }
```

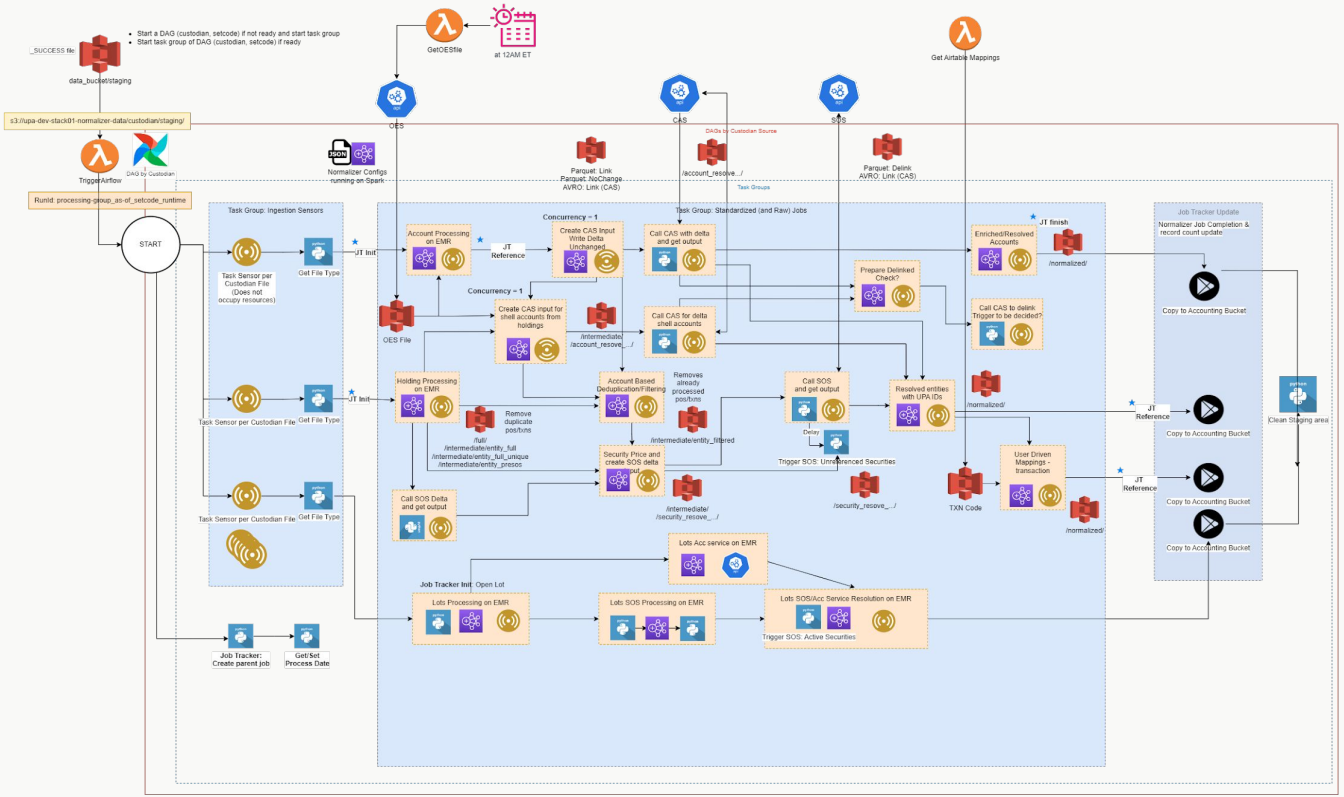
```python
for parent in dependencies:
    first_task = task.steps[0]
    if stage == 'root':
        dag.get_task(parent) >> Label(edge_label) >> first_task
    elif stage == destination_stage:
        tasks[parent].steps[-1] >> Label(edge_label) >> first_task
    else:
        if dag.has_task(f"{parent_task_group.group_id}.{parent}{parent_task_suffix}"):
            parent_task_group.get_child_by_label(
                f"{parent}{parent_task_suffix}") >> Label(edge_label) >> first_task
```

Simple DAG

MUS-Normalizer
Pipeline Architecture



Complex DAG


_SUCCESS file
data_bucket/staging
s3://upa-dev-stack01-normalizer-data/custodian/staging/
TriggerAirflow
DAG by Custodian
RunId: processing-group_as-of_setcode_runtime
START
Start a DAG (custodian, setcode) if not ready and start task group
Start task group of DAG (custodian, setcode) if ready
GetOESfile
at 12AM ET
OES
CAS
SOS
Get Airtable Mappings
Normalizer Configs running on Spark
Parquet: Link Parquet: NoChange AVRO: Link (CAS)
/account_resove.../
Task Groups
Parquet: Delink AVRO: Link (CAS)
Task Group: Ingestion Sensors
Task Sensor per Custodian File (Does not occupy resources)
Get File Type
JT Init
Task Group: Standardized (and Raw) Jobs
Concurrency = 1
Account Processing on EMR
JT Reference
Create CAS Input Write Delta Unchanged
Call CAS with delta and get output
Enriched/Resolved Accounts
JT finish
/normalized/
Prepare Delinked Check?
Call CAS to delink Trigger to be decided?
Create CAS input for shell accounts from holdings
OES File
/intermediate/account_resove.../
Call CAS for delta shell accounts
Holding Processing on EMR
Remove duplicate pos/txns
/full/ /intermediate/entity_full /intermediate/entity_full_unique /intermediate/entity_presos
Account Based Deduplication/Filtering
Removes already processed pos/txns
/intermediate/entity_filtered
Call SOS and get output
Delay
Trigger SOS: Unreferenced Securities
Resolved entities with UPA IDs
Security Price and create SOS delta
/intermediate/ /security_resove.../
/security_resove.../
Call SOS Delta and get output
TXN Code
User Driven Mappings - transaction
Lots Acc service on EMR
Job Tracker Init: Open Lot
Lots Processing on EMR
Lots SOS Processing on EMR
Lots SOS/Acc Service Resolution on EMR
Trigger SOS: Active Securities
Job Tracker: Create parent job
Get/Set Process Date
Job Tracker Update
Normalizer Job Completion & record count update
Copy to Accounting Bucket
(Clean Staging area)





Questions?

Email:

ipsa.trivedi@gmail.com
connect.subramanian@gmail.com

LinkedIn:

https://www.linkedin.com/in/ipsatrivedi
https://www.linkedin.com/in/svellaiyan