PulseAI Pipeline - Phase 1

BA 882 - Team 3

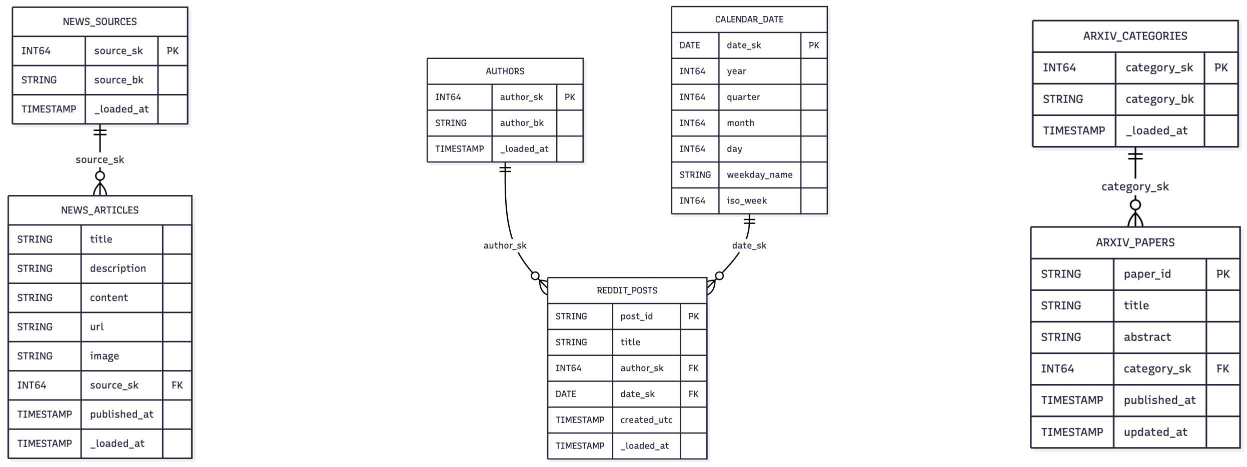
GitHub: PulseAI - GitHub

Jira: *Check Appendix PDF*

Objective: Our project aims to help users easily and conveniently stay up-to-date on AI news, research, and trends by leveraging an analytics pipeline that pulls relevant data from the internet and pushes this data through AI/ML workflows to produce curated summaries and dashboards. This will enable users to sift through the noise and digest curated outputs that get at the heart of the day's AI developments.

Data Model: Our data model brings together data from three sources: Reddit, GNews, and arXiv. Each of them follows a clear and consistent structure that makes it easy to analyze and connect information later.

The Reddit model focuses on posts, linking them with details about the authors and the dates they were created. The GNews model organizes articles and connects them with their respective news sources. The arXiv model captures information about research papers and groups them by category.



NEWS_SOURCES

INT64	source_sk	PK
STRING	source_bk	
TIMESTAMP	_loaded_at	

NEWS_ARTICLES

STRING	title	
STRING	description	
STRING	content	
STRING	url	
STRING	image	
INT64	source_sk	FK
TIMESTAMP	published_at	
TIMESTAMP	_loaded_at	

AUTHORS

INT64	author_sk	PK
STRING	author_bk	
TIMESTAMP	_loaded_at	

CALENDAR_DATE

DATE	date_sk	PK
INT64	year	
INT64	quarter	
INT64	month	
INT64	day	
STRING	weekday_name	
INT64	iso_week	

REDDIT_POSTS

STRING	post_id	PK
STRING	title	
INT64	author_sk	FK
DATE	date_sk	FK
TIMESTAMP	created_utc	
TIMESTAMP	_loaded_at	

ARXIV_CATEGORIES

INT64	category_sk	PK
STRING	category_bk	
TIMESTAMP	_loaded_at	

ARXIV_PAPERS

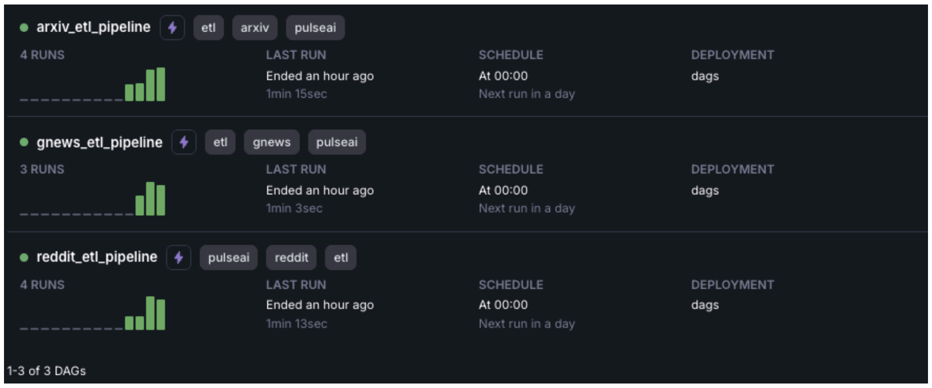
STRING	paper_id	PK
STRING	title	
STRING	abstract	
INT64	category_sk	FK
TIMESTAMP	published_at	
TIMESTAMP	updated_at	

Pipeline Flow: This step focused on establishing a robust, secure, and resilient data ingestion pipeline and implementing the necessary cron processes for scheduled execution. Our approach involved merging core API fetching logic into scalable Google Cloud Functions (GCF), which served as the primary, serverless execution environment for data acquisition.

To ensure enterprise-grade security, all sensitive API keys were externalized and securely managed within the deployment. This integration allowed the Cloud Functions to retrieve credentials at runtime, minimizing security risks and enabling efficient key rotation.

Astronomer (Apache Airflow) DAGs handled pipeline orchestration and scheduling. We designed the DAGs to manage the flow control, execute the GCF extraction tasks on a fixed cron schedule, and monitor task dependencies. The raw output from the ingestion process was then reliably staged within GCP Buckets, providing an immutable storage layer for all source files. The ingestion DAG pulls data twice a day, ensuring optimal data freshness and error-free for newsletter deployment.

Finally, the raw data was seamlessly integrated into BigQuery, where it was structured and optimized for high-performance querying and served as the single source of truth for all downstream tasks, including complex Natural Language Processing (NLP) and Machine Learning (ML) model training. Rigorous testing was performed on the pipeline and its resilience to ensure reliable, error-free operation under various load conditions.



```
● arxiv_etl_pipeline   etl  arxiv  pulseai
4 RUNS        LAST RUN             SCHEDULE             DEPLOYMENT
              Ended an hour ago    At 00:00             dags
              1min 15sec           Next run in a day

● gnews_etl_pipeline   etl  gnews  pulseai
3 RUNS        LAST RUN             SCHEDULE             DEPLOYMENT
              Ended an hour ago    At 00:00             dags
              1min 3sec            Next run in a day

● reddit_etl_pipeline   pulseai  reddit  etl
4 RUNS        LAST RUN             SCHEDULE             DEPLOYMENT
              Ended an hour ago    At 00:00             dags
              1min 13sec           Next run in a day

1-3 of 3 DAGs
```

<u>BQ & GCP Reporting/Analytics Capabilities:</u> For our project, we are utilizing *Google BigQuery* as our cloud data warehouse and are more broadly leaning on the Google Cloud (GCP) ecosystem of tools to execute our workflows. The usage of these tools can be broken into four main layers:

1. **Ingestion Layer**
   - *Google Cloud Functions*: We use a single "extract" function for each of the three APIs that pulls data from the source and lands it in our GCS bucket in its raw format.
   - *Google Cloud Storage (GCS)*: Currently, GCS is being used as the place where we store raw data before it undergoes light transformation prior to landing in BQ. Since this bucket will be updated daily in accordance with our pipeline scheduling, we chose a "Standard" configuration to minimize latency and costs.
2. **Loading Layer**
   - *Google Cloud Functions*: A "setup" function is the first to run in our pipeline, even before "extract," but "setup" is a component of the loading layer, as it creates the schema, dataset, BQ table, and performance settings that are expected by the "load" function. Subsequently, this load function reads the JSON files from our GCS files, assures no duplicate instances are included, and lands the new data in the BQ table created by "setup." Given that the setup function performs minimal transformations to make the data "BQ-ready," our pipeline can be considered one of EtLT.
   - *Google BigQuery*: BigQuery is our cloud data warehouse. BigQuery enables us to store and interact with our lightly transformed data, allowing it to be cleaned, altered, and enriched using SQL, which optimizes our data for subsequent downstream tasks in the pipeline.
3. **Transformation Layer**
   - *Google BigQuery*: BigQuery will function as our main transformation engine. While BQ is not a full-stop transformation tool, it will allow us to execute joins, handle nulls, and enrich data with summary and calculated columns. This initial cleaning can serve as a feeder to other tools, such as Vertex AI and

Dataproc clusters, whose outputs can be fed back to BQ and then pushed to BI tools, displaying our final dashboard/summaries.

4. **Analytics & Reporting Layer**
    - AI/ML: For AI and ML efforts, we will either use Vertex AI's off-the-shelf models, instantiate our own models in Dataproc, or a combination of the two based on our requirements. As we continue to optimize our initial pipeline flow, our exact approach to ML/AI implementation remains to be solidified. Our path will become clearer with further data transformation, enabling us to understand better how to balance team capabilities, cost, and project requirements.
    - BI Applications/Reporting: We are currently leaning towards using *Tableau* as our reporting engine, given our team's broad comfort level with the software and BQ's compatibility with a wide variety of BI tools via its API. However, this is another area where our approach is not yet solidified, as we may find that a GCP tool like Looker or another third-party tool like Power BI is more suitable for our needs.

Design Choices: Our reasoning for creating three separate pipelines (One per API) in Phase 1 was primarily for the purposes of practicality and traceability. Specifically, isolating the sources allowed us to write code unique to each one that was optimized for their unique structures and access requirements. This simplified the process of trying to ultimately land the data in BQ, a task that would have been confusing had these efforts and the three cloud functions per API been combined into a single DAG. In line with this, isolating the sources enables tracing pipeline errors more easily, and failed runs can have their cause quickly narrowed down without needing to question whether interactions between functions linked to different APIs caused the errors.

An ancillary benefit of this isolated structure is that it is logical within the context of our data. Given that one of the key qualitative focuses of this project is to understand how AI research, news, and public discourse compare on a day-to-day basis, this initial separation allows us the flexibility to conduct individual analysis for each source type. Although this initial pipeline lacks a central fact table to link our three sources, we plan to create one in subsequent stages, utilizing universal categories to facilitate cross-source groupings.

For project management, we utilize Jira for task assignment, progress tracking, and time management, and *GitHub* for version control. Given the test-heavy nature of the pipeline construction process, our team members' ability to create GitHub branches visible to the team but not impacting the main code base has been integral to our iteration and implementation.

Looking Ahead: Looking ahead, we would be working on the following:

- Using ML and NLP techniques, we will be extracting the keywords from the data that we have collected and creating classification categories
- Using these classification categories, we will unify our data model and add this classification to the EtLT pipeline
- Furthermore, we will be working on the AI part of it, using generative AI to create the AI-curated newsletter