General comments

This was an interesting challenge. I may have overcomplicated some steps. I've started simple but a few requirements got me and I needed to insert some tools and options.

The solution should be scalable to 100 million entries. It is encouraged to simplify the data by a data model. Please add proof that the solution is scalable.

Using the local FS we can "simulate" and "pretend" we are doing with a massive file(or multiples) to ingest. While moving this solution to production the ingestion process should change. We can benefit from some lambdas to process the csv, integrate spark with Glue, and have all that process together.

The Spark choice was having this scale in mind. Can't see how to process that quantity in a "simpler" batch or directly, in Database.

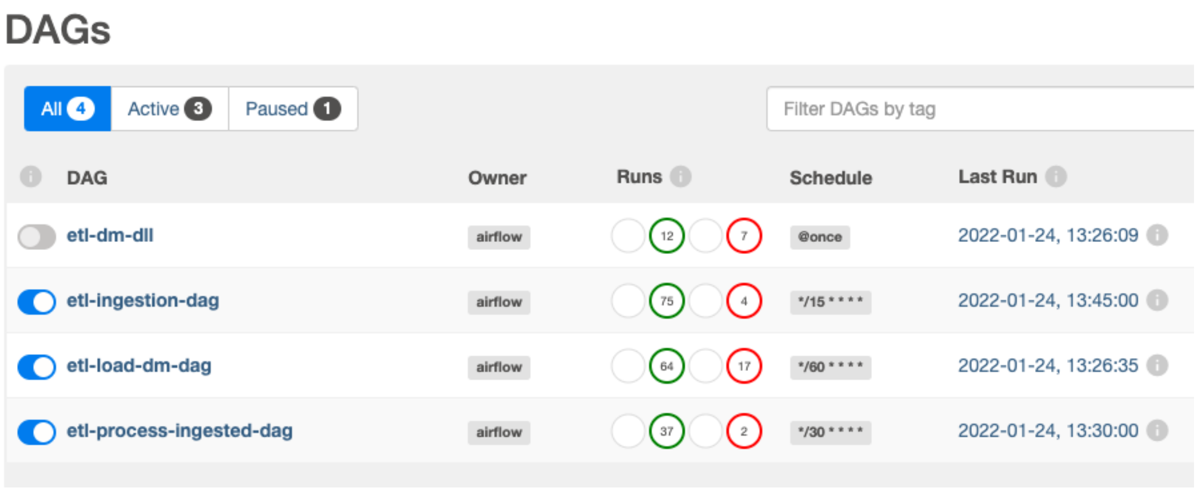
In the beginning, I thought to design the landing>staging>processed all in S3. This is why we have a terraform folder creating some buckets. Also, we have an option in the configs to choose between "S3" and "local" backends. I've dropped the first option so the solution relies only on the OS filesystem.

Develop a way to inform the user about the status of the data ingestion without using a polling solution.

My idea was to have Apache Airflow take care of the pipe executions. Airflow provides a nice UI to check for the execution and logs of the steps. We can even scale and add some cool integrations with Slack, Datadog, emails, etc. All of that to give visibility and clarity about the process.

Airflow dags

The DAGs are divided in the following:



DAGs

All 4 Active 3 Paused 1

Filter DAGs by tag

DAG	Owner	Runs	Schedule	Last Run
etl-dm-dll	airflow	12 7	@once	2022-01-24, 13:26:09
etl-ingestion-dag	airflow	75 4	*/15 * * * *	2022-01-24, 13:45:00
etl-load-dm-dag	airflow	64 17	*/60 * * * *	2022-01-24, 13:26:35
etl-process-ingested-dag	airflow	37 2	*/30 * * * *	2022-01-24, 13:30:00

All of them are using the DockerOperator which calls my project image.

etl-dm-dll

Will create five tables in a Postgres instance. It is a really simple “dm”. The dag has a task to do the truncate for all of the tables so, no SCD dimension or historic data are being kept. This can be changed by implementing some staging tables to then do the merge process. ***ddl queries are located in /dags folder***

etl-ingestion-dag

This dag will read the CSV file and convert it to parquet. The code executed by the docker image is in ***src/stream-csv-read-csv/read_convert.py***

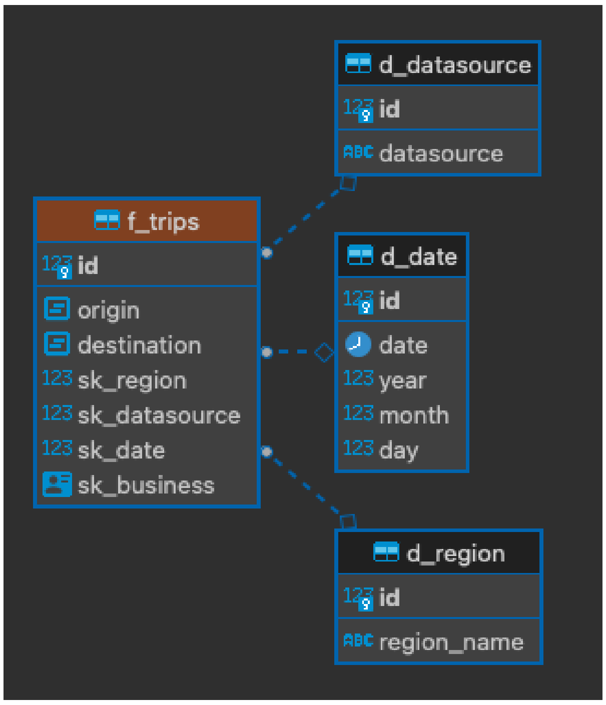
etl-process-ingested-dag

The step will consume the previous parquets and do some enhancements to it. Like adding new columns (year, month, date) breaking coordinates.. and saving to a new parquet file. ***src/process/spark_process.py***

etl-load-dm-dag

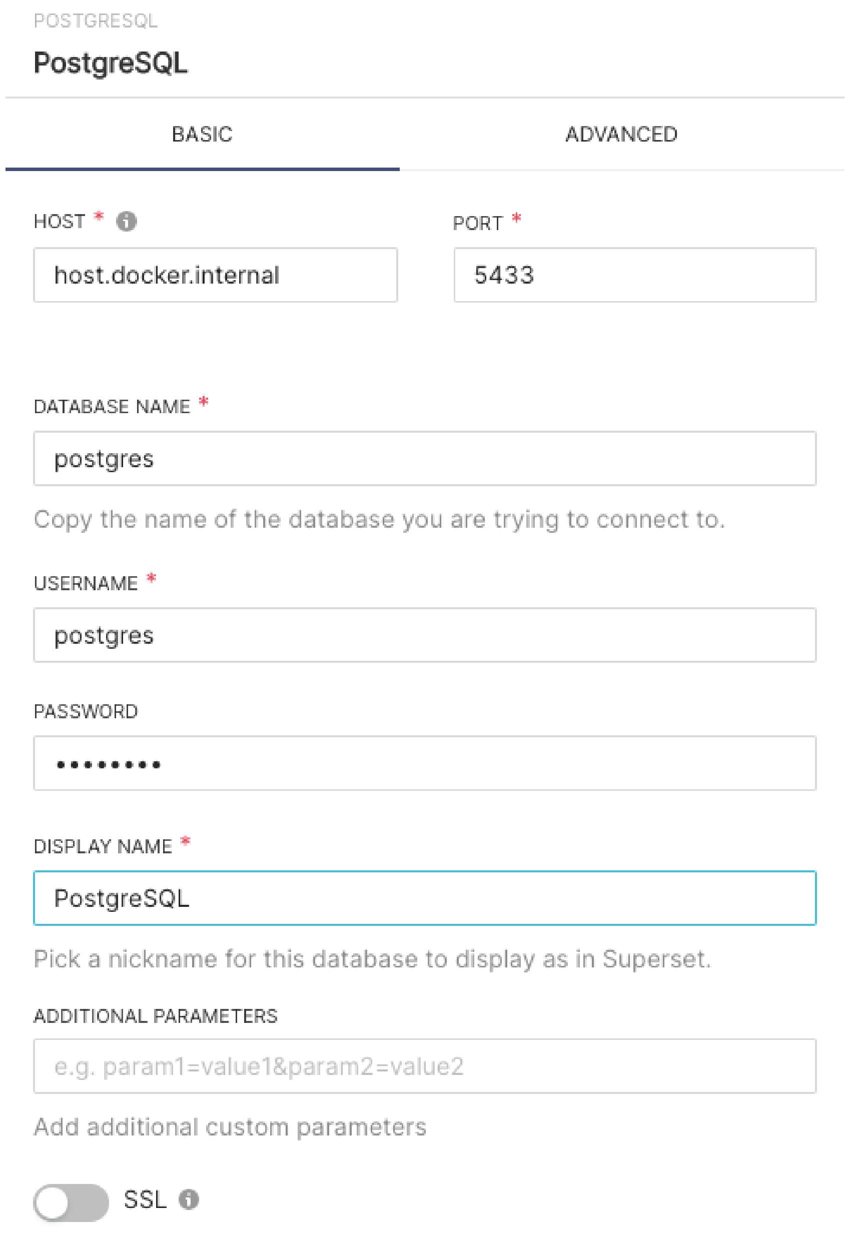
The final step. I've made some decisions here that probably won't make it in a production release. I'm consuming the previous parquet files and converting to CSV to use the "copy" command from psql Postgres. All the data from the current day will be converted.

If in AWS, I would make a bucket available through Redshift Spectrum so we don't need to do explicit insert/copy commands. Or, at least, do the copying S3 > Physical Redshift tables only for some tables. Master data could stay in S3 only and be consumed through Spectrum. Need to dedicate more time thinking about it. ***src/fill-dm/generate-import.py***



Visualizations

The solution docker-compose file should deploy Superset as well. Superset is an open-source visualization tool. I chose that tool because it offers a lot of flexibility and integrates well with other Apache tools. To be able to connect to the running postgres instance and do your own analysis, you would need to create a Database connection.


POSTGRESQL

PostgreSQL

HOST * host.docker.internal

PORT * 5433

DATABASE NAME * postgres

Copy the name of the database you are trying to connect to.

USERNAME * postgres

PASSWORD •••••••••

DISPLAY NAME * PostgreSQL

Pick a nickname for this database to display as in Superset.

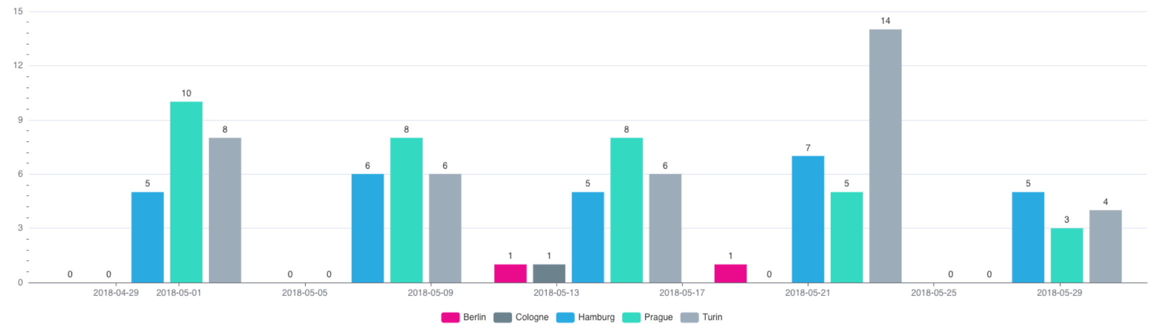
ADDITIONAL PARAMETERS e.g. param1=value1&param2=value2

Add additional custom parameters

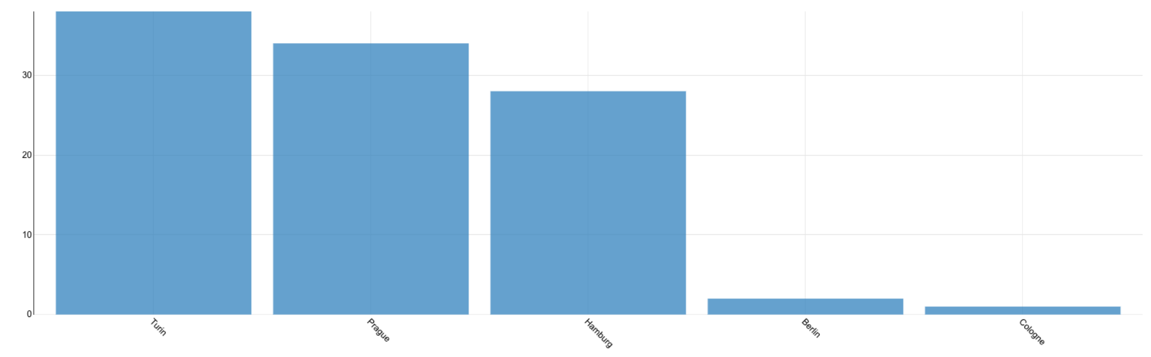
SSL

password: password

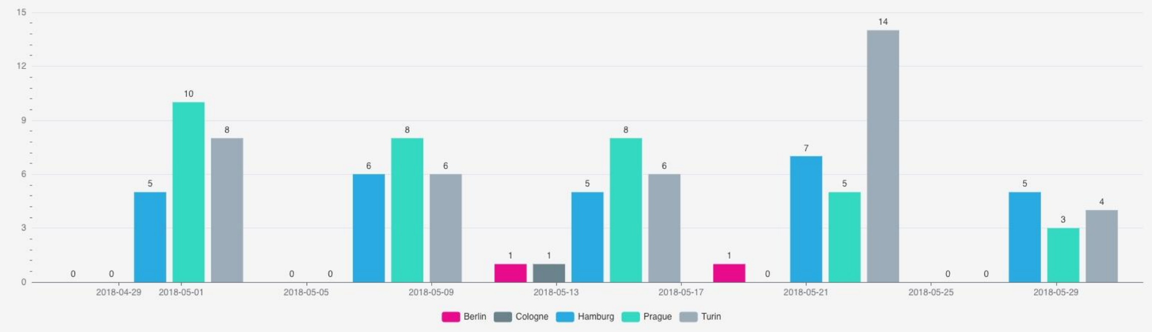
Trips by Region and Week



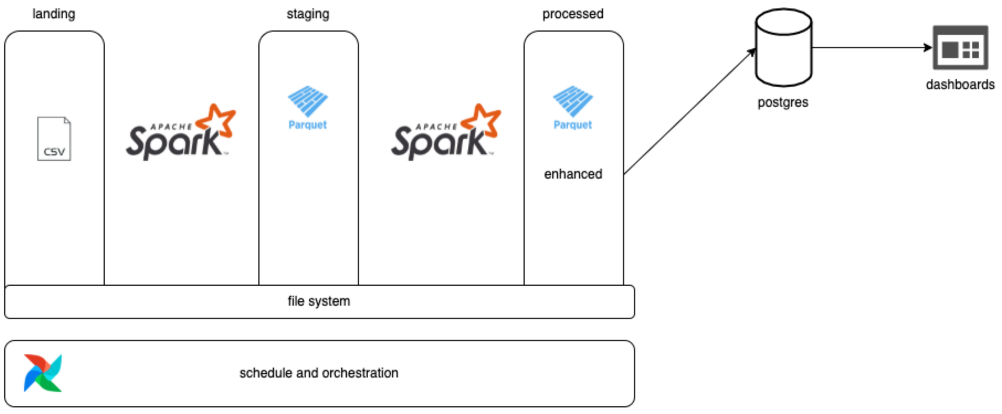
Trips by Region



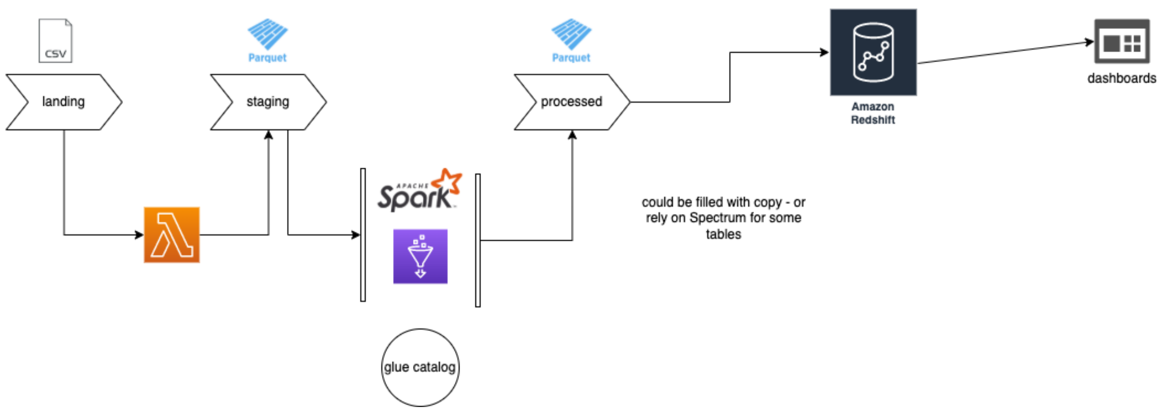
Trips by Region and Week



Current solution



Possible cloud solution


csv
Parquet
Parquet
landing
staging
processed
Amazon Redshift
dashboards
could be filled with copy - or rely on Spectrum for some tables
glue catalog