Requirements

1. The architecture may process some data as real time streams where applicable and some in batches;
2. Assume that real time data from different RabbitMQ Queues/Kafka topics may need to be consolidated and aggregated before being consumed by end applications (assume that a single process may be represented by multi-topic events, all of which are related by a single ID);
3. Ensure that access to sensitive data is limited to specific roles (eg. data scientists may need access to the raw data, but business analysts only require access to aggregate data);
4. The architecture can be designed using any tool or technology, with preference to open source code and cloud services;

Solution

The solution below describes an outline of a data infrastructure architecture project that can solve and meeting the requirements specified by challenge 1.

In a data engineering process, there are some important steps that need to be followed, in order to build consistent and efficient ingestion pipelines and data structures. Through data engineering, we can add value to the business by delivering consolidated data sources to data analysts, data scientists, BI analysts, etc., helping in data-driven decision making.

To build this data infrastructure architecture, we will summarize the steps in **acquisition**, **modeling, processing/transformation**, **storage**, **visualization**, and **access monitoring/control**.

The following figure illustrates the architecture proposed in this solution.

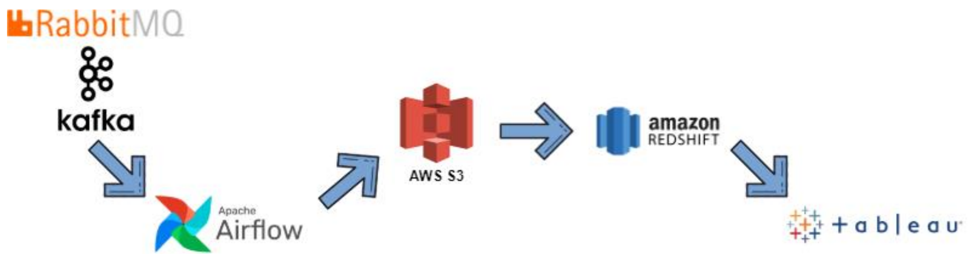


Figura 1 - Data infrastructure architecture example.

Figure 1 showed the technologies used in each step of the data infrastructure and ingestion pipeline, from acquisition to visualization. We can highlight:

1. **RabbitMQ/Kafka:** Data messaging / streaming tools that exchange data between the **producer** (the company's back-end applications) and the **consumer** (acquired by Apache *Airflow*) through the different topics mentioned. Using a Kappa architecture, we will already be able to fulfill Requirement 1, making it possible to process data in batch or streaming way.
2. **Apache Airflow**: Open-source platform for flow management. There, it is possible to build the entire data pipeline. In this project proposal, *Airflow* will be responsible for the acquisition, processing, and monitoring of the data.
3. **AWS S3**: Also called Simple Storage Service, it is a service provided by AWS that provides the storage of objects of the most varied types. In this project, he will be responsible for storing the data files consumed from the *Kafka topics*.
4. **AWS Redshift**: Data Warehousing service offered by AWS. Its focus on building OLAP solutions. In Redshift, data coming from AWS S3 will be stored in a predefined data structure, so that it meets the business and its needs.
5. **Tableau**: It is a very intuitive data visualization tool. It is possible to create the most varied types of dashboards, giving the analytics teams a simple and complete solution. Tableau enables integration with different data sources, making it a very good option for consolidating data stored on Redshift. An open-source alternative to Tableau would be the ***MetaBase*** tool, which has features very close to what *Tableau* delivers.

Project infrastructure

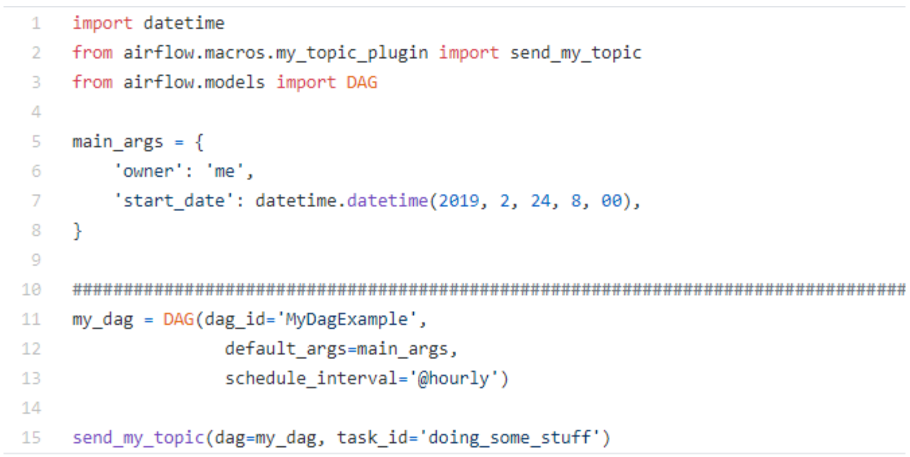
For the development of this project, it will be necessary to run Apache *Airflow* on a server, create a bucket on AWS S3 and a cluster on Redshift. For the last two, just create an account on Amazon Web Services (https://aws.amazon.com/) and make the settings directly on the console of each service. For the execution of *Airflow*, it will be necessary to install it and, in this project propose, we can use ***Docker*** and ***Docker Compose*** (https://www.docker.com/) to provide the necessary environment for its execution. If the visualization tool chosen is ***MetaBase***, we can also install and configure it over Docker. On the other hand, if the tool chosen is Tableau (https://www.tableau.com/), it will be necessary to obtain a Tableau Desktop license according to the needs of the analysts.

The first step before starting the data acquisition step is to perform the installation of *Docker* and the configuration of *Docker Compose* for *Airflow* and *Kafka*. If you choose to use MetaBase, you will also need to configure the specifications for the tool in Docker Compose.

Data acquisition

As mentioned in the challenge description, there are back-end applications that post messages for Kafka topics and in this project, these applications will act as **Producers** for *Kafka topics*. Therefore, our goal will be to build the **Consumer** of the application with the help of *Kafka-Python API*.

This consumer consists of 1) a python function that performs sequential readings of messages from *Kafka topics*, groups and converts the data into a *numpy array* and saves the result in a chosen format (for example: csv, json, parquet, etc.); and 2) An *Airflow Directed Acyclic Graphs* (DAG) containing the instruction for executing the consumer function. Through *Airflow*, we can define the rules and frequency of execution of this DAG. The following image is a simple example of the structure of an Airflow DAG.



```python
import datetime
from airflow.macros.my_topic_plugin import send_my_topic
from airflow.models import DAG

main_args = {
    'owner': 'me',
    'start_date': datetime.datetime(2019, 2, 24, 8, 00),
}

##################################################################################
my_dag = DAG(dag_id='MyDagExample',
               default_args=main_args,
               schedule_interval='@hourly')

send_my_topic(dag=my_dag, task_id='doing_some_stuff')
```

Figura 2 - dag_example.py

Data modeling

Before starting the data processing and storage, it is necessary to define what the modeling/structure will be and what visualizations are desired by the Analytics teams. This is an extremely important stage, as it will be directly involved in strategic decision making of the company. Therefore, to fulfill these definitions, **data scientists**, **data analysts** and **managers** must always be involved, enabling an uncomplicated modeling that achieve all the desired purposes.

As this is a hypothetical project proposal and the Redshift database is for objective analytical functions, we can adopt the **Star Schema modeling** due to its simplicity, using fact and dimension tables. The following image is a basic example of this type of modeling:

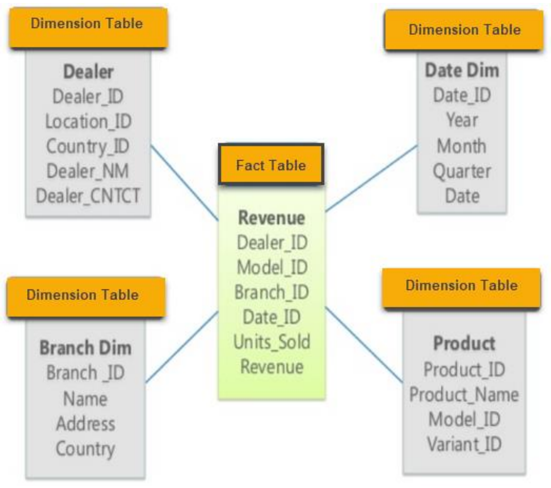


Figura 3 - Star Schema Example

Data processing

As we are working with *Airflow*, the ETL process will be encapsulated in a DAG composed of several tasks for data transformation. The transformation python files must be built based on the modeling already studied and predefined. In this way, data obtained from different topics can be consolidated, aggregated, and correlated, fulfilling Requirement 2.

To exemplify a transformation, suppose that the data brought in the **data acquisition** step are, in the great majority, *Boolean* (1 or 0, True or False). However, in consolidation the data for manipulation and visualization, the data needs to be categorized (Buy/Not-Buy, Deny Credit/Allow Credit etc.). In this case, a transformation script will be needed to perform this treatment.

A very important observation here is the storage location of each of the files mentioned in this proposal (DAGs, data files from Kafka topics, configuration files, transformation results, etc.). If the project is to be implemented, it is necessary to carry out the entire configuration of *Apache Airflow* directories inside the corresponding container, all mapped inside Docker Compose.

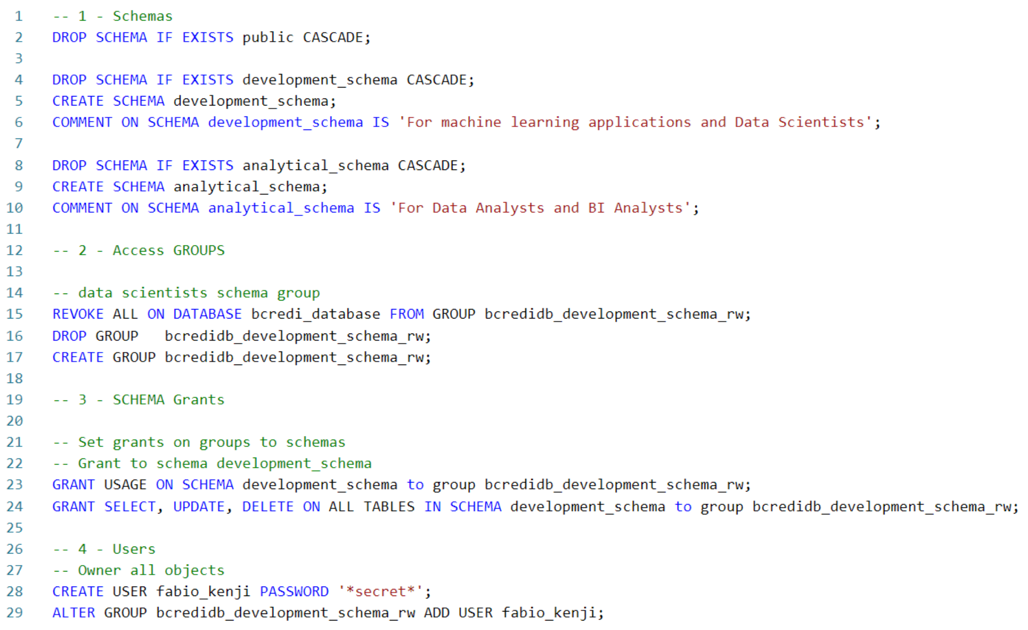
Data storage

In this step, we need to store the extracted and properly transformed data into Redshift. For this we will need to follow the following steps:

1. In this step, schemas and table structures should be created according to the business needs (defined in the data modeling step) and segmentation of users by function (defined by the challenge – data scientists and data analysts) into the data warehouse. Redshift has the possibility of creating **user groups**, guaranteeing **personalized permissions** for each existing group. In this way, we can deny or grant different permission levels (SELECT, INSERT, UPDATE, DELETE) for each group,

segmented by schemas (***development_schema, analytical_schema, admin_schema, data_visualization_schema etc***), ensuring compliance with Requirement 3. In Figure 4 we have an example of an SQL script for these purposes.

2. In this step, we need to save the previously processed data in an S3 bucket. For that, we will need to create a function that has as input parameters the name of the processed data file. This will retrieve the file and then save it in the bucket.
3. Just like the previous step, we will need to build another function that takes as a parameter the same file name, and the name of the destination table that will be loaded with the data. For this, Redshift has the ***COPY command***, which is capable of loading DW tables according to files saved in S3.
4. The last step will be the construction of an SQL script that, based on the DW tables, should build a set of **Materialized Views** in Redshift. The views will consolidate the tables and, in addition to facilitating the connection with data visualization tools, it also optimizes the reading process in the data warehouse.



```sql
-- 1 - Schemas
DROP SCHEMA IF EXISTS public CASCADE;

DROP SCHEMA IF EXISTS development_schema CASCADE;
CREATE SCHEMA development_schema;
COMMENT ON SCHEMA development_schema IS 'For machine learning applications and Data Scientists';

DROP SCHEMA IF EXISTS analytical_schema CASCADE;
CREATE SCHEMA analytical_schema;
COMMENT ON SCHEMA analytical_schema IS 'For Data Analysts and BI Analysts';

-- 2 - Access GROUPS

-- data scientists schema group
REVOKE ALL ON DATABASE bcredi_database FROM GROUP bcredidb_development_schema_rw;
DROP GROUP   bcredidb_development_schema_rw;
CREATE GROUP bcredidb_development_schema_rw;

-- 3 - SCHEMA Grants

-- Set grants on groups to schemas
-- Grant to schema development_schema
GRANT USAGE ON SCHEMA development_schema to group bcredidb_development_schema_rw;
GRANT SELECT, UPDATE, DELETE ON ALL TABLES IN SCHEMA development_schema to group bcredidb_development_schema_rw;

-- 4 - Users
-- Owner all objects
CREATE USER fabio_kenji PASSWORD '*secret*';
ALTER GROUP bcredidb_development_schema_rw ADD USER fabio_kenji;
```

Figura 4 - schemas_groups_users_tables_script.py

DAG configuration in *Airflow* and Data Visualization

The penultimate step of this data infrastructure proposal will be to build the DAG execution logic with all the necessary tasks for the ETL process. First, all the functions related to file processing such as the steps of **data acquisition** and **data processing**. We can also create a task that sends all the transformed files to S3, all in parallel. Then, a task should be performed to load the data for the dimension tables in parallel. It is important to highlight that the dimension tables must be loaded before the fact table, as it depends on the data contained in the other tables. After carrying out all loads on the dimension tables in parallel, we can load the fact table. Finally, we can run the process that builds all Redshift materialized views for the data visualization tool and analytical needs.

The last step of this data architecture proposal is the installation, configuration, and connection of Tableau with Redshift views. In this step, after creating an account and downloading Tableau Desktop, we can connect with the Redshift data source and choose the schema/table to be used as the data source. Finally, dashboards can be developed in a way that best meets the demands of management, helping an intelligent and data-driven decision making.

Conclusion

This proposal aimed to present a data infrastructure architecture that enables the acquisition, modeling, processing, storage, visualization, and control of data access for different stakeholders. The project has a simple structure and seeks to bring a low-cost implementation solution, allowing its development in any context.

It is hoped that the presented solution can bring powerful insights and future applications for the company.