Individual assignment

Flow of data and data transformation

I would define our API endpoints (Quotes and Listings form coinmarketcap), producer.py, and Kafka servers as upstream components. As downstream components I would define our connect_api.py, currencies.py, consumer.py, PostgreSQL server and dashboard.py as parts of the downstream. First, our main data “quotes/latest” we use for the dashboard are retrieved in the file connect_api.py by the function get_latest_coin_data. When this data are imported into producer.py, we then use get_latest_coin_data again in the main function to serialize data where symbol = “BTC, SOL”, to transform it into a Kafka message in json format that we send into our topic that we named “crypto” inside our Kafka container named “broker”.

When the data are then handled in consumer.py, where we use def main function to retrieve data from our topic “crypto” by using app -> crypto_topic to transform the data from json format to a python dictionary and then creating a streaming dataframe enabled by having imported the Quikstreams module. In the next step the data is filtered by variables we wanted to include in our dashboard by the retrieve_coins_info function. Thereafter we sink the data into PostgreSQL via the table “quotes_coins” by using .sink and the create_postgres_sink function.

In the next step data are loaded into dashboard.py, by using SQLAlchemy we are able to use the create_engine function to connect with PostgreSQL, and import the quotes_coins table that we run a sql statement on and transform into a dataframe. Finally we use use the Streamlit module to enable our dataframe to present streaming data for our dashboard.

In our file listings_api.py, we retrieve API data “listings/latest” in the get_listings_data function and transforms it to sql format in tables “cap_top_10” and “vol_top_10” in the main function and sends it into PostgreSQL, thus skipping the need for a producer file, consumer file and Kafka broker container.

Considerations of the dashboard, purpose of different parts in the architecture and technologies used

A key consideration was to create a dashboard where we provide data that’s in the interest of a crypto trader, thus the focus to create the 4 graphs that shows percent change for different time formats.
Two other simple considerations we made to increase the attractiveness of the dashboard was to first create our “Moon platoon” logo and that the colors used for the piechart was used as well for the lines in our linecharts to give the dashboard a more cohesive impression.

We found inspiration for the base of our architecture of the project from the lectures in this course for how our teacher organized a streaming data pipeline. Our intent was to first copy his structure with only one currency to test that this would work, and thereafter test with more currencies and structure for processing API data. So after we managed this task we then experimented and tested a more efficient version in listings_api.py where could create up/down stream functions in one single file that could be imported to Postgres.

In terms of technologies we used Coinmarketcap as API endpoint as source/to collect data, python for programming language for the project, docker to store our Apache Kafka containers locally where we then used our broker container to enable real-time streaming data into our consumer.py and using Quik streams library to facilitate this connection between Kafka to our consumer.
We used a PostgesSQL server locally store our transformed cryptocurrency data and SQLAchemy and Psycopg2 to connect and interact with PostgreSQL when importing our tables into dashboard.py.
For data visualization with real-time streaming data in our dashboard we used Streamlit and Streamlit Autorefresh to enable continuing updates. Pandas was used to create the dataframe used and for data manipulation e.g. converting currencies. Finally, we used matplotlib to create the linecharts and pie diagrams used in the dashboard.

Possible developments of the project and reflections of the team groupwork

A first main task would be to update our table "Current Price" to add values for every hour on the X axis and make it possible to interact with the line graphs with inspiration from our classmates. Other additions could be to create a selectbox for all other currencies and when clicked different percent change graphs would appear with more compressed information in a single window frame, e.g. circulating supply and market cap dominance.
I would as well like to try copy the compression of the source data collection, consumer and producer for the whole project like we did in listings_api.py, if possible.

I found that the best way to collaborate in our group was to meet up IRL as solving issues with the code for instance went much faster compared with trying to do this remote. We divided most of the coding of the different parts of the project between us, but I found it more productive and inspiring when we tried pair programming as we learned from each other and because as you had to verbalize your thought process for the given syntax to your groupmate. So, for future group projects I would prefer more IRL work and pair/mob programming.