READ ME

Here is the streaming pipeline design for getting the top 3 highest average stock price per 1 min / 5 mins / 15mins.

Attachment: publish.py,top\_3.py,top\_3\_change\_most.py

1. At first, I registered and used Quote endpoint to retrieve data of ten similar price stocks:

<https://sandbox.iexapis.com/stable/stock/market/batch?symbols=futu,aapl,pdd,msft,fb,bidu,gme,baba,bili,kod&types=quote&token=Tpk_45a453752ec8497f86a9527198cc7af2&chartLast=1>

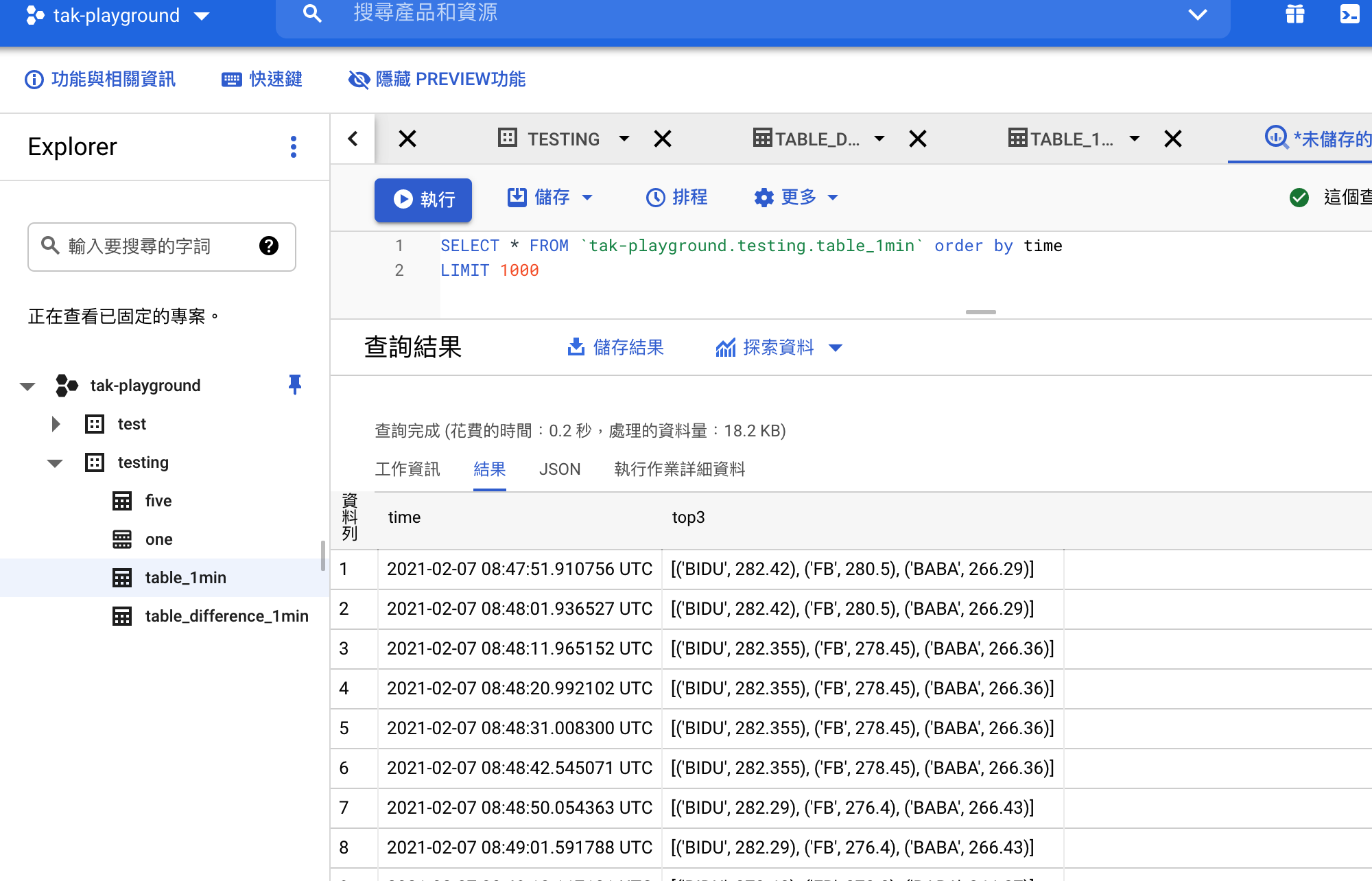
1. Then I created a python script(publish.py) to call the above api and then publish the data to a pub sub topic. And then attach it to Cloud Function.
2. Create tasks in cloud scheduler to trigger (publish.py) every 10 seconds to publish data to pubsub topic.
3. Create a pubsub topic to receive data from publish.py
4. Use dataflow (apache beam) to do transformation of streaming data from pubsub and publish to bigquery. Here I use three hopping windows (window size = 60s/300s/900s) to aggregate data to get the average and filter the top 3 highest prices of stocks.

Reference: top\_3.py

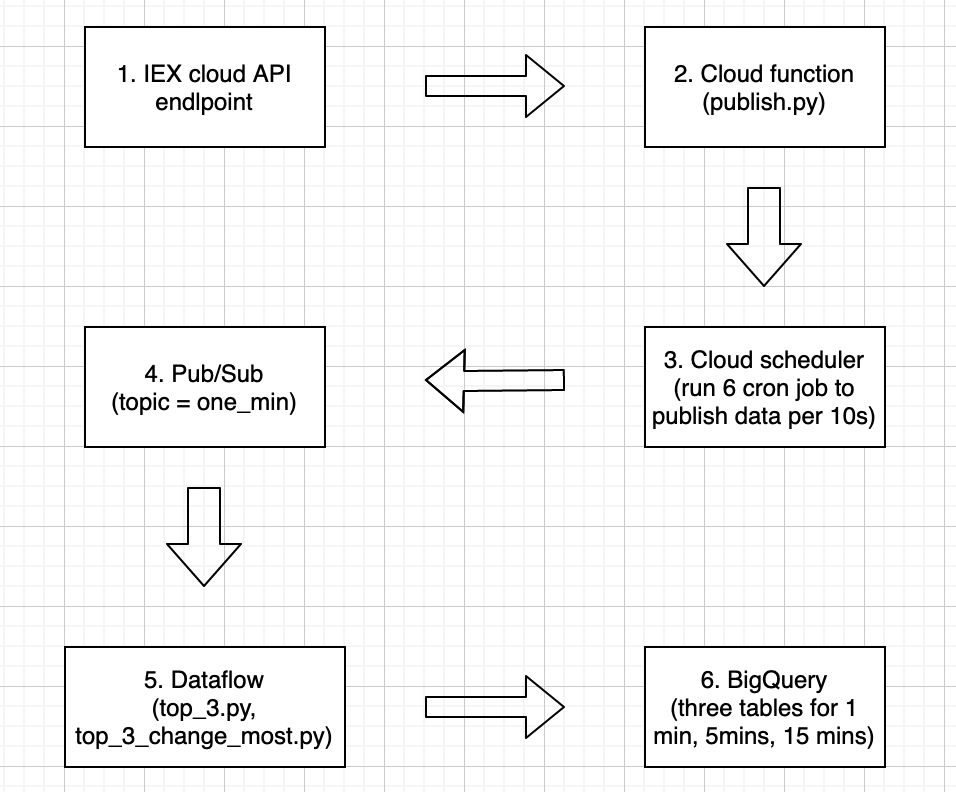
I created another dataflow pipeline to get top 3 max change the difference of stock price within a period (window size = x)

Reference: top\_3\_change\_most.py

1. Finally we can read data in bigquery table



Streaming data pipeline flow chart:



Remarks:

1. For normal practice, finalized data should be sent to Pub Sub then inserted to bigquery. However,for this task, since bigquery is the only one consumer so that I skip this.
2. To implement the publish data action to pubsub, I can opt cloud run/ cloud function and app engine. For this task, app engine is too heavy so I choose between cloud run/ cloud function. Eventually for simplicity, I adopt cloud functions as the differences between them are not significant in this case.
3. For the purpose of backfill, a staging place is needed. We can add logic in step 2(cloud function) to store staging data to google cloud storage if needed.
4. I will suggest using cloud composer(apache airflow) to wrap the whole data pipeline for a more complicated case. However, for this task the logic is simple and not many dependencies are needed so I did not opt cloud composer.