CORE SCIENTIFIC

# ORDER BOOK TRIAL PROJECT DOCUMENT

Design and Implementation on Google Cloud

Durai Rajamanickam
02/25/2022
rdmurugan@yahoo.com

## 1. Requirements:

This document is one of the deliverables from Core Scientific trial project work. The below requirements were received for the trial project.

1. Create a data pipeline/ETL to ingest and persist order book data across two different exchanges for both BTC/USD and ETH/USD markets:

    a. Every 60 seconds poll each exchange API for the current order book and persist raw order book data.

    b. For each poll, extract $100k of bid and ask order book data for each exchange. i. The goal here is for each poll we should be able to reason about the price we could achieve when market buying or selling $100k of BTC or ETH.

2. The Business Intelligence team would like to consume the following data for their models in real time at 60 second granularity:

    a. What is the average mid-price per market?

    b. Which exchange would we prefer to execute a $50k buy or sell order on? At what price?

## 2. Architecture Considerations:

- ✓ Elastic Scalability
- ✓ Performance
- ✓ Cost of Operations (DevOps, DataOps)
- ✓ Ease of Development and Integration
- ✓ Ease of Maintenance
- ✓ Adaptable to future change

## 3. Design Considerations:

### 3.1 Data Processing approach:

1. Enable Live reporting – Real time analysis
2. Each exchange reports orderbook at various lengths, and fields so perform ETL to normalize incoming data from various exchanges.
3. Use Analytics warehouse to process large volume of data. (Considering 60 sec continued stream of data)
4. Ability to enable Batch processing for Historical/Timeseries analysis

### 3.2 Data Pipelines

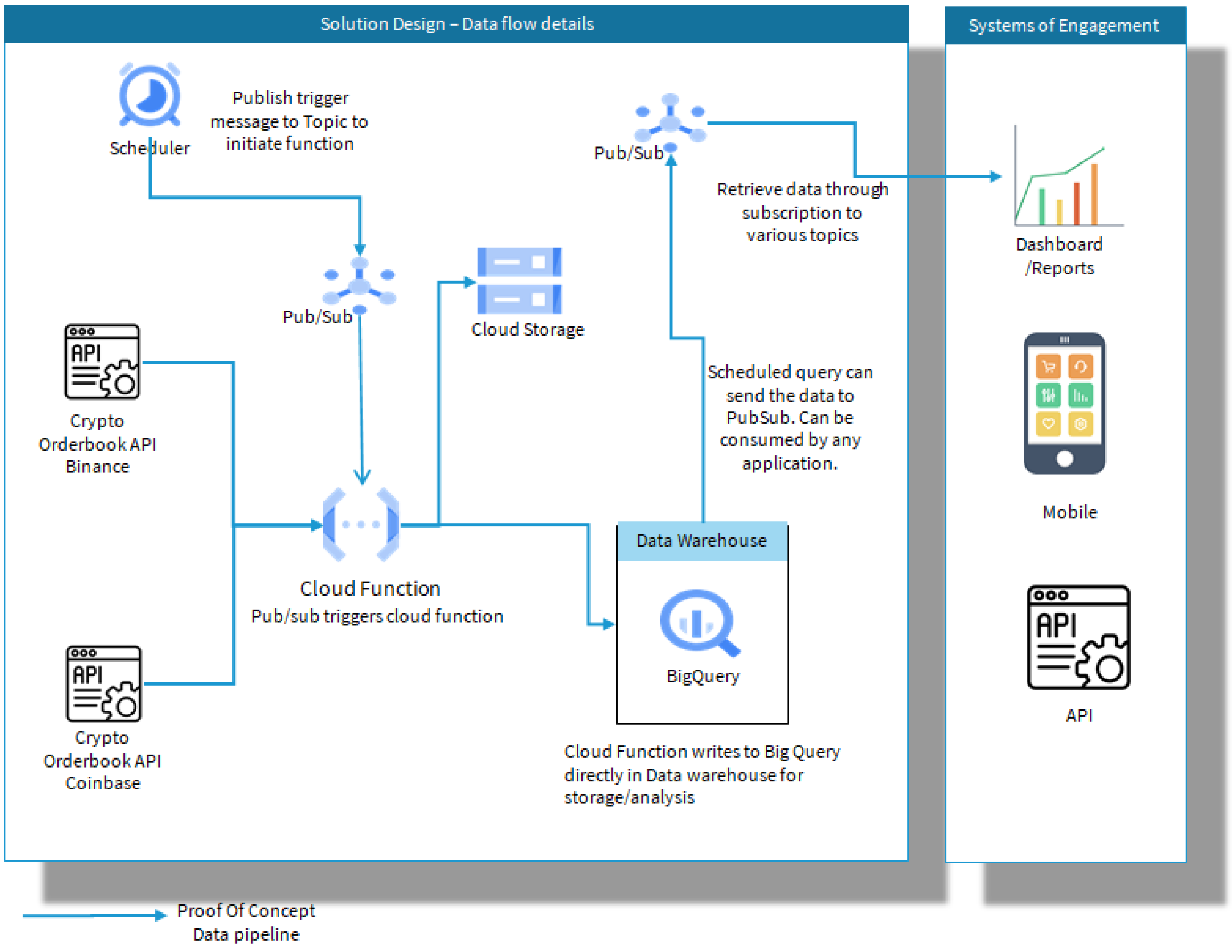
1. Use data pipeline that is scalable and real time (considering 60 sec data feed). High throughput, and dynamic scaling required to meet the requirements.
2. Consider two exchanges (Binance and Coinbase) for this PoC to retrieve order book.
3. Extract $40K to $100k of bid and ask order book data for each exchange
4. Use scheduler to poll the exchange's orderbook API end point every 60 seconds

### 3.3 Live Reporting

1. Enable analytics warehouse to meet live reporting requirements to find the average mid-price.
2. Provide granularity at exchange and type of trade level between asks and bids. [Persist data at granular level]

## 4. Functionality and Design Decision

1. Core Scientific uses GCP and AWS, The PoC is leveraging GCP managed services.
2. Use Orderbook API from Binance and Coinbase exchanges for the PoC
3. Each exchange sends a slightly different structure of order book. Build modularized ETL framework to clean up the order book and standardize the data (Embedded in Cloud Function for PoC purpose)
4. Use Cloud Scheduler (GCP component) to trigger the cloud function. The scheduled sends a message in a 60-sec interval to topic that triggers the function to pull data from exchanges.
5. Use on-demand scalable cloud function. Build framework to extract the data to add more exchanges in the future.
6. Once the function is triggered by pubSub, the process will retrieve order book data through API.
7. Send standardized "order book" with the following data structure [side [bid, asks], price, quantity, cost, timestamp, exchange] to BigQuery (Data Warehouse) and Cloud Storage. (This program extracts only trade values between $40K to $100 K for PoC purpose)
8. From Big query we can analyze and extract mid-price ranges and other questions like when we can purchase 50k crypto tokens.


Solution Design – Data flow details
Systems of Engagement
Scheduler
Publish trigger message to Topic to initiate function
Pub/Sub
Retrieve data through subscription to various topics
Dashboard /Reports
Pub/Sub
Cloud Storage
Crypto Orderbook API Binance
Scheduled query can send the data to PubSub. Can be consumed by any application.
Mobile
Cloud Function
Pub/sub triggers cloud function
Data Warehouse
BigQuery
API
Crypto Orderbook API Coinbase
Cloud Function writes to Big Query directly in Data warehouse for storage/analysis
Proof Of Concept Data pipeline

Architecture Diagram for the PoC

## 5. Implementation Steps for Real time Orderbook Extraction

The following pre-requisites should be available to deploy the code in Google Cloud for account level permissions and security. Also, this will enable turning on the components quickly.

1. Set up your organization resource
2. Add users and groups to your organization
3. Set up administrator access to your organization
4. Set up billing
5. Set up the resource hierarchy
6. Set up access control for your resource hierarchy
7. Set up support
8. Set up networking configuration

9. Set up logging and monitoring
10. Configure security settings for apps and data

**Command – Line Setup Instructions:**

- Install and initialize the Google Cloud CLI.
- gcloud init
- export PROJECT_ID=coredata-trial
- gcloud projects create $PROJECT_ID
- export SERVICE_ACCOUNT_NAME=CoreDataSrvc
- gcloud iam service-accounts create SERVICE_ACCOUNT_ID --description="service account for coredataservices" --display-name="coredatasrvc"
- export PROJECT_ID=$(gcloud config get-value project)
- export IAM_ACCOUNT=$SERVICE_ACCOUNT_NAME@$PROJECT_ID.iam.gserviceaccount.com
- gcloud projects add-iam-policy-binding $PROJECT_ID --member serviceAccount:$IAM_ACCOUNT --role roles/owner
- gcloud iam service-accounts add-iam-policy-binding SERVICE_ACCOUNT_ID@PROJECT_ID.iam.gserviceaccount.com --member="user:USER_EMAIL" --role="roles/iam.serviceAccountUser"
- export GOOGLE_APPLICATION_CREDENTIALS=path/to/your/credentials.json
- export BUCKET_ID=coredatastore001
- gsutil mb gs://$BUCKET_ID

**Enable following API**

- Dataflow API
- Compute Engine API
- Cloud Logging API
- Cloud Storage
- Google Cloud Storage JSON API
- BigQuery API
- Cloud Pub/Sub API
- Cloud Datastore API
- Cloud Resource Manager API

**Create Topics to Trigger Function:**

The below topic triggers “PushOrderBook” function. Message can be anything. This topic is exclusively to trigger the function.

- gcloud pubsub topics create projects/coredata-trial/topics/GetOrderBook

**Deploy the Cloud Function:**

- Create a cloud function with Pub-sub trigger.
- Deploy the function “PushOrderbook”
- Use main.py code from Github
- Check if function is running successfully.

**Testing and Running the Function:**

- gcloud pubsub topics publish GetOrderBook --message="Hello World!"

**Scheduling for 60 sec data retrieval:**

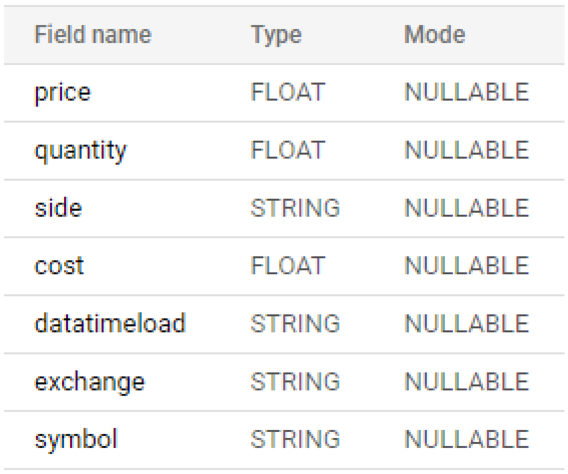
- gcloud scheduler jobs create pubsub publisher-job --schedule="* * * * *" --topic=GetOrderBook --location=US --message-body="Hello!"
- gcloud scheduler jobs run publisher-job

**BigQuery Setup:**

Create following project, dataset and table in BigQuery.

coredata-trial->orderbookdataset->orderbookcrypto

Create below table structure to store the orderbook data.



| Field name | Type | Mode |
|---|---|---|
| price | FLOAT | NULLABLE |
| quantity | FLOAT | NULLABLE |
| side | STRING | NULLABLE |
| cost | FLOAT | NULLABLE |
| datatimeload | STRING | NULLABLE |
| exchange | STRING | NULLABLE |
| symbol | STRING | NULLABLE |

CORE SCIENTIFIC

The above arrangement will setup complete end to end suite of data pipeline for the given requirements of 60sec orderbook interval data collection from Crypto Exchanges.

## 6. Business Intelligence requirements:

The below requirements can be achieved by running following queries at any given time interval.

a. What is the average mid-price per market?

The below SQL would help extract the mid-price per market.

```
select
    exchange,
    symbol,
    cast ((high_price + low_price)/2 as float64) as buy_price
    from (select
    exchange,
    symbol,
    max (case when side='bids' then price end) as high_price,
    min (case when side='asks' then price end) as low_price,
from coredata-
trial.orderbookdataset.orderbookcrypto group by exchange, symbol)
```

b. Which exchange would we prefer to execute a $50k buy or sell order on? At what price?

The below SQL would help extract the exchange and price per market.

```
select
    exchange,
    symbol,
    cast ((high_price + low_price)/2 as float64) as buy_price
    from (select
    exchange,
    symbol,
    max (case when side='bids' then price end) as high_price,
    min (case when side='asks' then price end) as low_price,
from coredata-
trial.orderbookdataset.orderbookcrypto where cost between 49500 and 50500  g
roup by exchange, symbol)
```