Streaming Data with AWS EC2

CIS 9760 - Big Data Technologies
Professor Ecem Basak

Mariya Mithaiwala - mariya.mithaiwala@baruchmail.cuny.edu

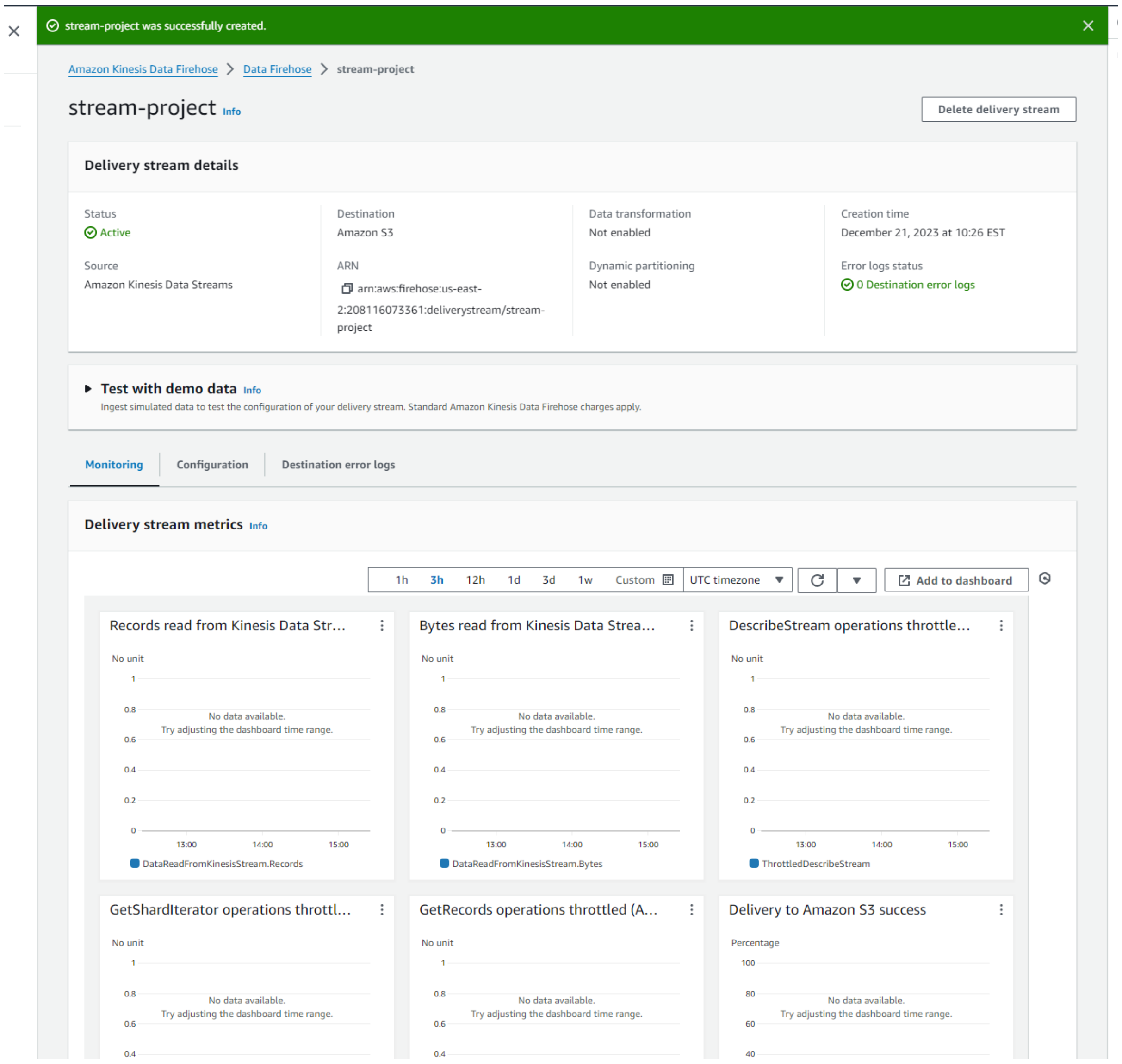
Table of Contents

Overview

The streaming finance data project involves the creation of a real-time data pipeline for interactive querying, leveraging AWS EC2 and related services. The project encompasses four key infrastructure components: an EC2 application (DataTransformer) for data collection, a Kinesis stream (DataCollector) to hold the data, a serverless process (DataAnalyzer) for querying S3 data using AWS Athena, and visualizations (DataVisualization) of the query results. The initial phase involves setting up a Kinesis Delivery Stream, developing an EC2 application to collect actual finance data (stock prices) using the yfinance module, and pushing the data into the Kinesis stream. Subsequently, AWS Glue is configured to facilitate interactive querying of S3 files generated by the DataTransformer. The project concludes with the generation of visualizations using Jupyter Notebooks. The Data Transformation phase involves collecting ten days' worth of stock HIGH and LOW prices for various e-commerce stocks, transforming the data into JSON objects, and putting them into the Kinesis stream. Data Analysis focuses on preparing the data for analysis, setting up a Glue crawler, and running Athena queries to extract insights into daily volatility for each company. The final step, Data Visualization, requires the creation of visualizations based on query results, answering critical questions about maximum volatility trends and daily average volatility per company. The project's artifacts include application code, S3 content screenshots, query outputs, Jupyter Notebooks, and a comprehensive README providing insights and answers to visualization questions.

Data Transformation

The Data Transformer phase in this AWS Kinesis streaming exercise involves several key steps to set up the necessary infrastructure for data streaming into an S3 bucket. Firstly, an S3 bucket is created with a meaningful name to push streaming data into it. Following that, we log into AWS and search for Kinesis on the homepage. A Kinesis stream is then created by giving it a name and choosing the On-demand capacity mode. The process proceeds to set up a delivery stream under Amazon Kinesis Data Firehose, a specialized version of Kinesis used to automatically read and drop messages into S3.



stream-project was successfully created.

Amazon Kinesis Data Firehose > Data Firehose > stream-project

stream-project Info

Delivery stream details

Status	Destination	Data transformation	Creation time
Active	Amazon S3	Not enabled	December 21, 2023 at 10:26 EST
Source	**ARN**	**Dynamic partitioning**	**Error logs status**
Amazon Kinesis Data Streams	arn:aws:firehose:us-east-2:208116073361:deliverystream/stream-project	Not enabled	0 Destination error logs

Test with demo data Info
Ingest simulated data to test the configuration of your delivery stream. Standard Amazon Kinesis Data Firehose charges apply.

Delivery stream metrics Info

Records read from Kinesis Data Str… — No unit — No data available. Try adjusting the dashboard time range. — DataReadFromKinesisStream.Records

Bytes read from Kinesis Data Strea… — No unit — No data available. Try adjusting the dashboard time range. — DataReadFromKinesisStream.Bytes

DescribeStream operations throttle… — No unit — No data available. Try adjusting the dashboard time range. — ThrottledDescribeStream

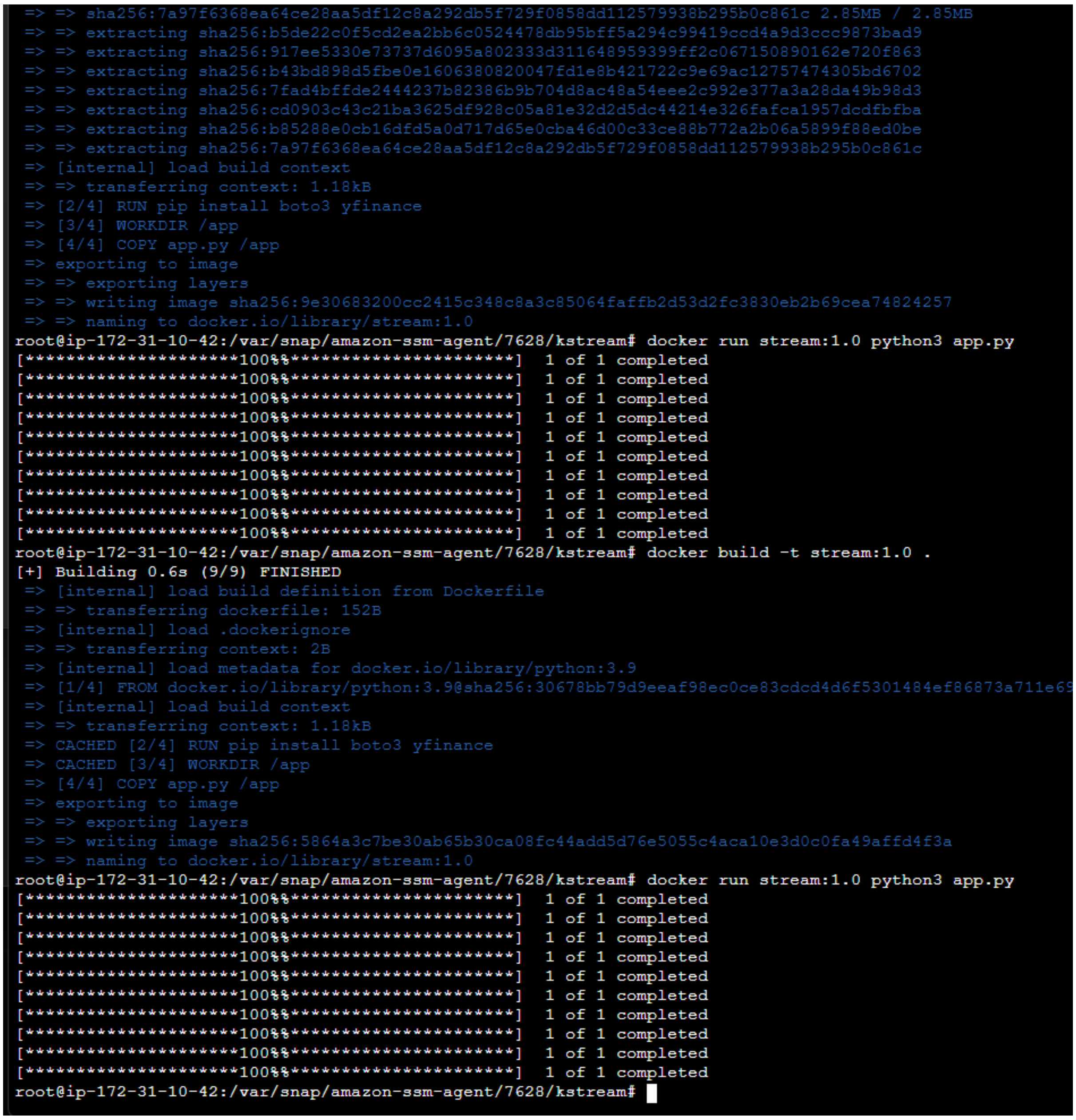
GetShardIterator operations throttl… — No unit — No data available. Try adjusting the dashboard time range.

GetRecords operations throttled (A… — No unit — No data available. Try adjusting the dashboard time range.

Delivery to Amazon S3 success — Percentage — No data available. Try adjusting the dashboard time range.

Data Collector

This process includes selecting S3 as the destination, specifying the S3 bucket path, and configuring buffer size and interval settings The next step involves providing access to the EC2 instance, which acts as the producer, to connect and push data into the Kinesis stream. This is achieved by configuring the IAM role associated with the EC2 instance, attaching the necessary policies, and ensuring proper permissions are granted. Once the Data Collector is configured, the EC2 instance, acting as the producer, can begin streaming data into the Kinesis stream, which subsequently triggers the Kinesis Data Firehose to read and automatically deliver the data to the specified S3 bucket. The entire process ensures a streamlined flow of streaming data from the source to the storage destination for further analysis and visualization.



```
 => => sha256:7a97f6368ea64ce28aa5df12c8a292db5f729f0858dd112579938b295b0c861c 2.85MB / 2.85MB
 => => extracting sha256:b5de22c0f5cd2ea2bb6c0524478db95bff5a294c99419ccd4a9d3ccc9873bad9
 => => extracting sha256:917ee5330e73737d6095a802333d311648959399ff2c067150890162e720f863
 => => extracting sha256:b43bd898d5fbe0e1606380820047fd1e8b421722c9e69ac12757474305bd6702
 => => extracting sha256:7fad4bffde2444237b82386b9b704d8ac48a54eee2c992e377a3a28da49b98d3
 => => extracting sha256:cd0903c43c21ba3625df928c05a81e32d2d5dc44214e326fafca1957dcdfbfba
 => => extracting sha256:b85288e0cb16dfd5a0d717d65e0cba46d00c33ce88b772a2b06a5899f88ed0be
 => => extracting sha256:7a97f6368ea64ce28aa5df12c8a292db5f729f0858dd112579938b295b0c861c
 => [internal] load build context
 => => transferring context: 1.18kB
 => [2/4] RUN pip install boto3 yfinance
 => [3/4] WORKDIR /app
 => [4/4] COPY app.py /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:9e30683200cc2415c348c8a3c85064faffb2d53d2fc3830eb2b69cea74824257
 => => naming to docker.io/library/stream:1.0
root@ip-172-31-10-42:/var/snap/amazon-ssm-agent/7628/kstream# docker run stream:1.0 python3 app.py
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
root@ip-172-31-10-42:/var/snap/amazon-ssm-agent/7628/kstream# docker build -t stream:1.0 .
[+] Building 0.6s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 152B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.9
 => [1/4] FROM docker.io/library/python:3.9@sha256:30678bb79d9eeaf98ec0ce83cdcd4d6f5301484ef86873a711e69
 => [internal] load build context
 => => transferring context: 1.18kB
 => CACHED [2/4] RUN pip install boto3 yfinance
 => CACHED [3/4] WORKDIR /app
 => [4/4] COPY app.py /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:5864a3c7be30ab65b30ca08fc44add5d76e5055c4aca10e3d0c0fa49affd4f3a
 => => naming to docker.io/library/stream:1.0
root@ip-172-31-10-42:/var/snap/amazon-ssm-agent/7628/kstream# docker run stream:1.0 python3 app.py
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
root@ip-172-31-10-42:/var/snap/amazon-ssm-agent/7628/kstream#
```

Data Analyzer

The Data Analyzer phase in the AWS Kinesis streaming exercise revolves around preparing and querying the data that has been collected in the S3 bucket. To facilitate this, an AWS Glue crawler is set up to run against the S3 bucket, allowing the stored data to be cataloged for efficient querying. Following the cataloging process, AWS Athena is employed to execute queries against the data, enabling the extraction of valuable insights. Specifically, the queries aim to compute essential metrics such as average volatility, highest volatility, and lowest volatility per company per day, providing a comprehensive analysis of the financial data collected over the specified period. The results of these queries are saved in a CSV file named 'results.csv,' which includes columns for company, date, average volatility, highest volatility, and lowest volatility. A corresponding 'query.txt' file contains the SQL query executed in AWS Athena to generate the results. The Data Analyzer phase ensures that the collected data is prepared for in-depth analysis and serves as a foundation for the subsequent visualization stage.

Data Visualization

The Data Visualization phase of the AWS Kinesis streaming project focuses on creating meaningful visual representations of the analyzed financial data to aid in understanding and interpretation. Leveraging Jupyter Notebook, the 'Analysis.ipynb' file is crafted to generate two distinctive visualizations based on the results obtained from AWS Athena. The first visualization takes the form of a single Line Chart, where each line corresponds to a specific company, depicting the maximum volatility trend over the specified period. This chart shows how Costco’s shares have the highest volatility. One of the reasons for this volatility can be associated with changes in consumer behavior, competition, or industry regulations that may contribute to fluctuations in stock prices.

The second visualization adopts the form of a Grouped Bar Chart, where each group represents a company, and the bars within each group represent the daily highest volatility. This chart provides insights into the daily volatility trends across companies, allowing for a deeper understanding of how volatility varies on a day-to-day basis. By juxtaposing the results from the two visualizations, one can assess that Costco’s shares depicted as the most volatile in the first chart consistently exhibit high volatility across different days in the second chart.

References:

Professor’s - In-class Assignment - Provisioning an EC2 instance
CIS 9760 - Extra Credit Assignment - Getting an Object into S3 via AWS Management
CIS9760 - In-class Assignment - Querying Big Data on S3 via AWS Athena and AWS Glue
CIS9760 - In-class Assignment - Streaming data via AWS Kinesis Nov 14 2022
ChatGpt for app.py