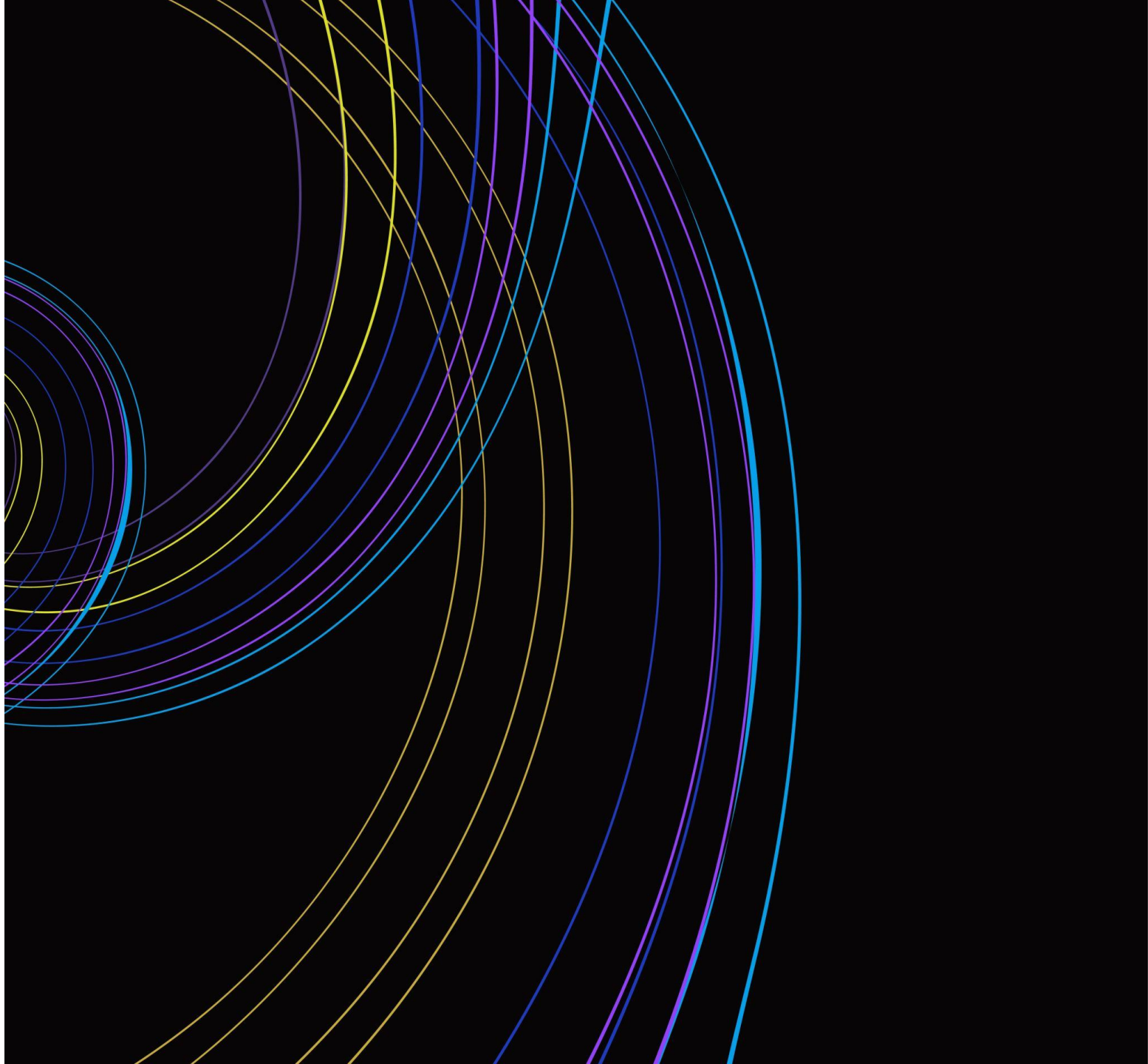
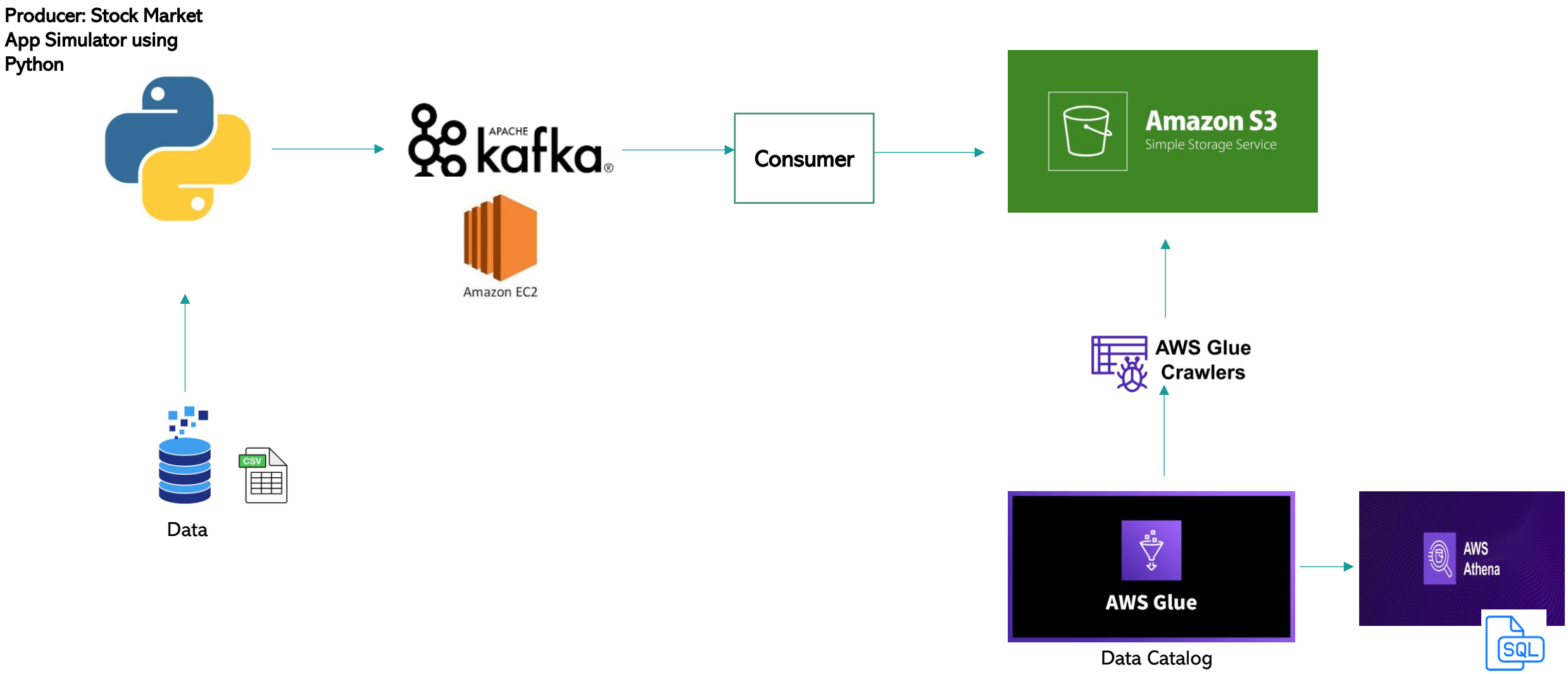
STREAMING & PROCESSING WITH KAFKA

~Siddhant Prakash More



ARCHITECTURE DIAGRAM



PRODUCER:

```
[2]: import pandas as pd
     from kafka import KafkaProducer
     from time import sleep
     from json import dumps
     import json
```

```
[3]: producer = KafkaProducer(bootstrap_servers=[''], #change ip here
                              value_serializer=lambda x:
                              dumps(x).encode('utf-8'))
```

```
[4]: producer.send('demo_testing2', value={'surname':'more'})
```

```
[4]: <kafka.producer.future.FutureRecordMetadata at 0x29628c84c10>
```

```
[5]: df = pd.read_csv("indexProcessed.csv")
```

```
[6]: df.head()
```

[6]:

	Index	Date	Open	High	Low	Close	Adj Close	Volume
0	HSI	1986-12-31	2568.300049	2568.300049	2568.300049	2568.300049	2568.300049	0.0
1	HSI	1987-01-02	2540.100098	2540.100098	2540.100098	2540.100098	2540.100098	0.0
2	HSI	1987-01-05	2552.399902	2552.399902	2552.399902	2552.399902	2552.399902	0.0
3	HSI	1987-01-06	2583.899902	2583.899902	2583.899902	2583.899902	2583.899902	0.0
4	HSI	1987-01-07	2607.100098	2607.100098	2607.100098	2607.100098	2607.100098	0.0

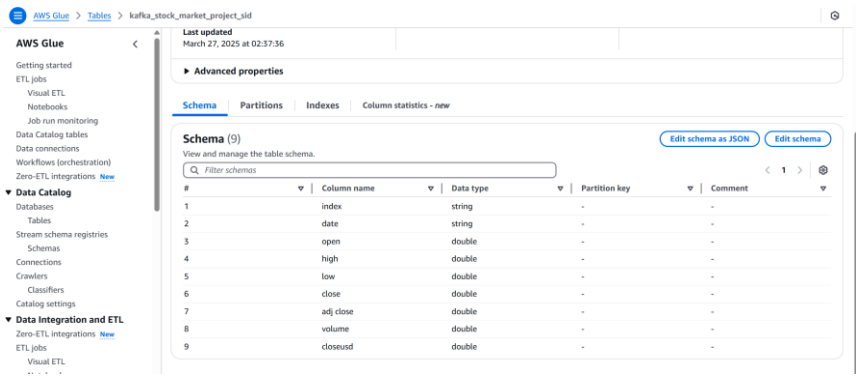
```
[7]: while True:
         dict_stock = df.sample(1).to_dict(orient="records")[0]
         producer.send('demo_testing2', value=dict_stock)
         sleep(1)
```

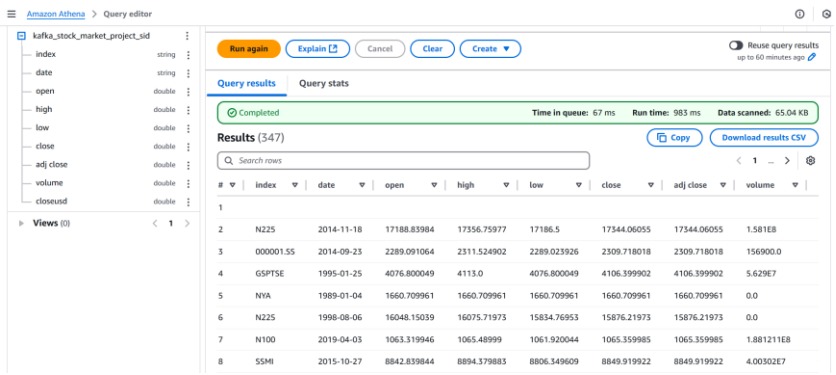
- Simulates real-time stock market data using a preprocessed CSV file
- Reads data using **Pandas** and converts it to JSON format.
- Sends messages to Kafka topic (demo_testing2) using KafkaProducer
- Uses a while True loop to continuously stream random stock records at 1-second intervals.
- Acts as a real-time data generator running on an EC2 instance.

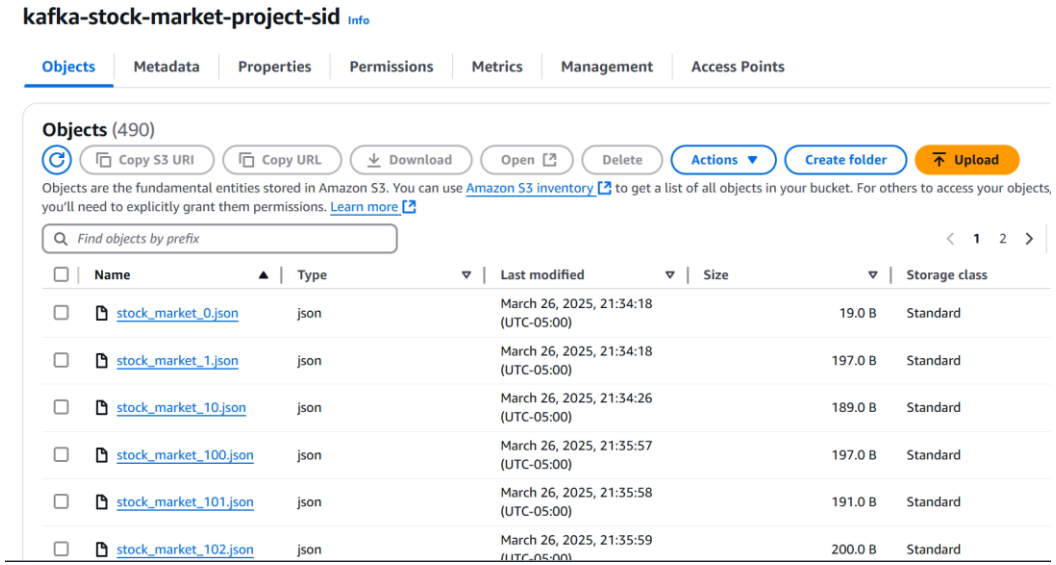
CONSUMER:

```
[1]: from kafka import KafkaConsumer
     from time import sleep
     from json import dumps,loads
     import json
     from s3fs import S3FileSystem

[2]: consumer = KafkaConsumer(
         'demo_testing2',
          bootstrap_servers=[''], #add your IP here
         value_deserializer=lambda x: loads(x.decode('utf-8')))

[3]: # for c in consumer:
     #     print(c.value)

[4]: s3 = S3FileSystem()

[5]: for count, i in enumerate(consumer):
         with s3.open("kafka-stock-market-project-sid/stock_market_{}.json".format(count), 'w') as file:
             json.dump(i.value, file)
```

- Listens to the Kafka topic demo_testing2 using KafkaConsumer.
- Deserializes incoming JSON messages using UTF-8 encoding.
- Uses s3fs with boto3 under the hood to interact with Amazon S3.
- For each consumed message, generates a new .json file.
- Uploads the file to the S3 bucket kafka-stock-market-project-sid.
- Runs on an EC2 instance to stay close to Kafka and minimize latency.







DATA STORAGE, CATALOGING & QUERYING

CONCLUSION:

Built a real-time data streaming pipeline using Apache Kafka on AWS EC2.

Simulated stock market data using a Python producer.

Streamed data into Kafka topics, then consumed and stored it in Amazon S3.

Automated metadata cataloging with AWS Glue Crawlers.

Enabled serverless querying and analysis using Amazon Athena.

Demonstrated how Kafka integrates with AWS for scalable and real-time data processing.