# Lab 4 - MSK Streaming Pipeline and Application Deployment

Objectives:

1. Build a real-time Amazon MSK streaming analytics pipeline in Managed Apache Flink Studio using Apache Flink and Apache Zeppelin.
2. Visualize the output.
3. Build and deploy the Zeppelin notebook as a long-standing application with the ability to durably store data in Amazon S3.

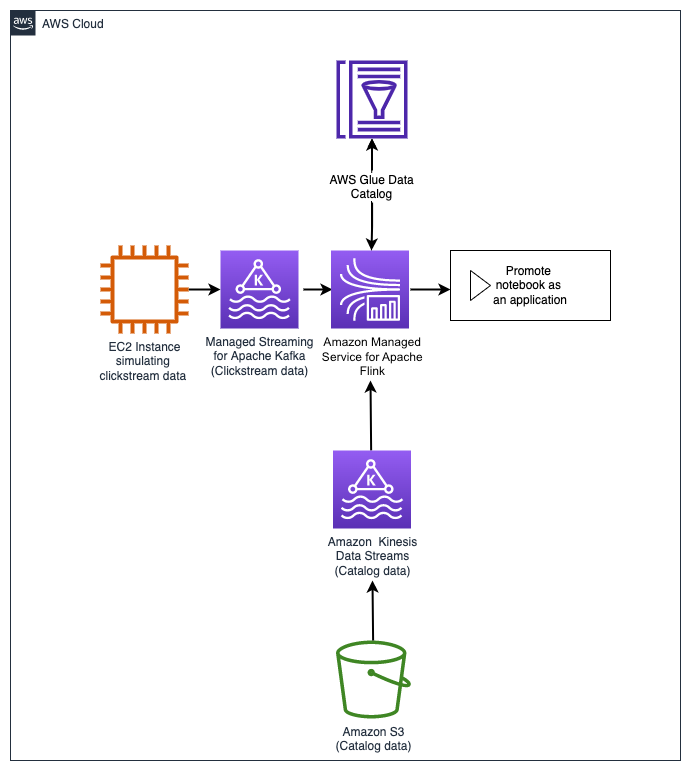
Simplified Steps [GPT]

1. Prepare the Environment: Set up the tools you'll use.
2. Create Data Channels: Set up channels to collect data.
3. Generate Fake Data: Simulate customer interactions.
4. Write Analysis Code: Use Zeppelin to analyze data in real-time.
5. **Deploy Analysis Code: Make the code run continuously and save results to Amazon S3.**

Point 5 -

Deploying the Zeppelin Notebook as an Application:

1. Once you are satisfied with the processing logic and the results, the notebook can be promoted to run as a long-standing application.
2. Continuous Processing: The deployed application continuously processes incoming data in real-time, without further manual intervention.
3. Fault Tolerance: The results are stored in Amazon S3, ensuring that even if there are any failures, the processed data is safely stored and can be analyzed later.



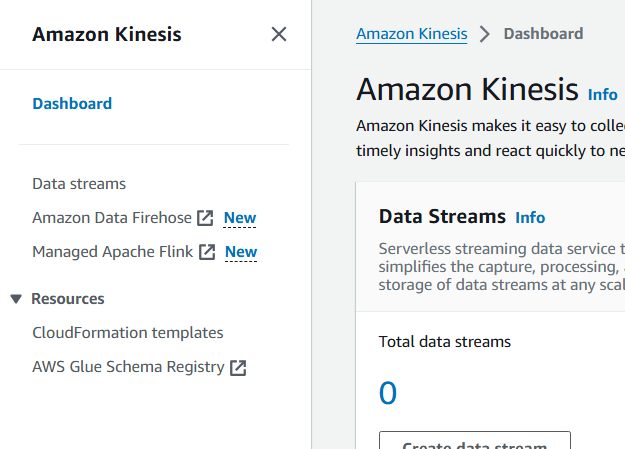
**Doubt -** how is kafka/msk getting integrated into this? in 2nd lab, we used KDS to get data from clickstream. Why are we using MSK here?

Ans [GPT] -

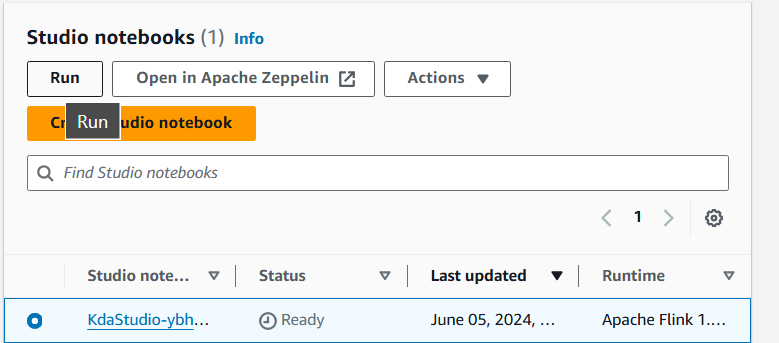
1. EC2 acts as a producer for Kafka here.
2. Flink acts as a consumer.
3. Realtime processing, as MSK does
4. In summary, the decision to use Kafka/MSK in this lab scenario instead of KDS likely stems from the advantages Kafka offers in terms of ecosystem compatibility, performance, flexibility in data retention, and operational control.

**Task 1: Setting up the Zeppelin notebook environment**

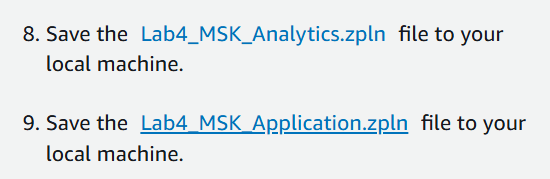
1.1 Open Kinesis in console, and select Managed Apache Flink, inside it, open Studio Notebook option.



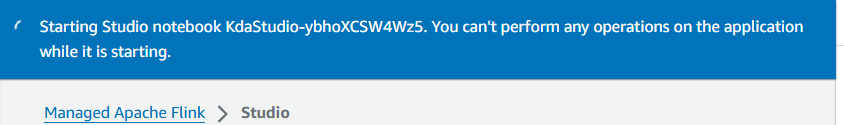
1.2 Run the pre-created notebook



1.3 Meanwhile, download these two zeppelin files:



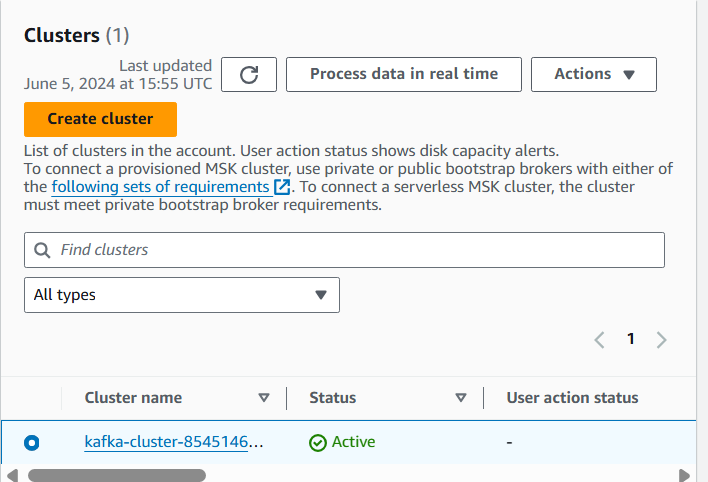
1.4 Wait for this to complete



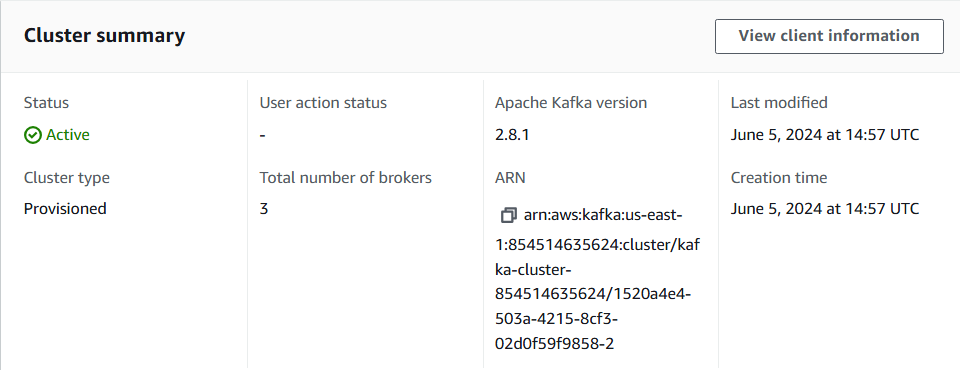
**Task 2: Create topics in the MSK cluster and simulate clickstream data generation**

2.1 Open MSK in console (new tab)

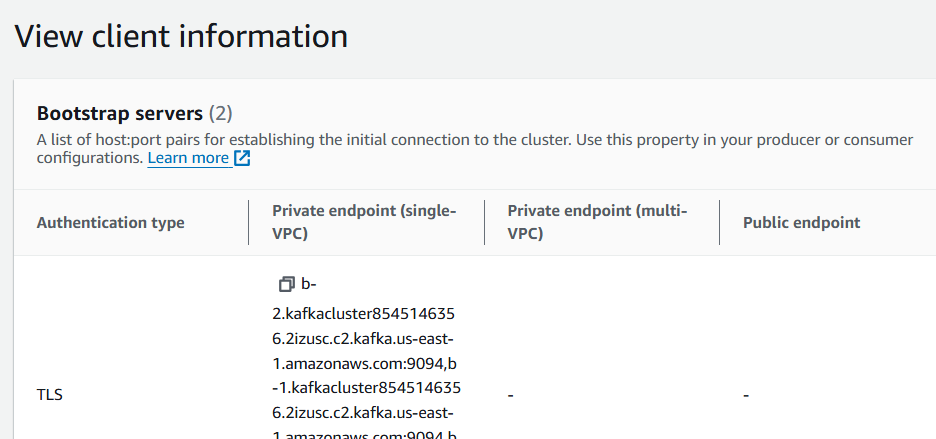
2.2 Click on the link for pre created cluster



2.3 Select to view client info



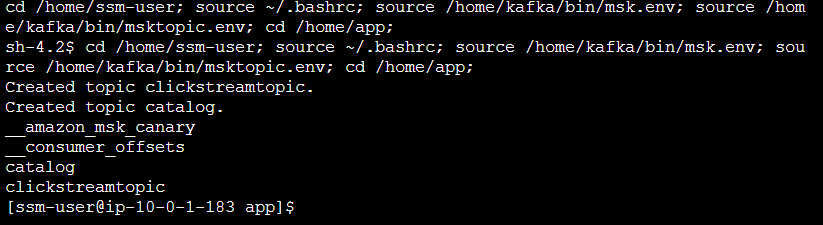
2.4 These values will be used further in the lab (not clear to me)



2.5 Open the CLI URL given in lab in new tab

2.6 This automatically creates the following topics:

1. Clickstreamtopic
2. catalog



Rest are the ones created by default during provisioning of the cluster.

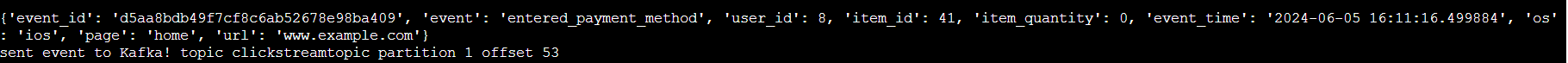
2.7 SIMULATE CLICKSTREAM DATA GENERATION



This script starts the clickstream generator and writes to the clickstreamtopic MSK topic.

**EC2 is now functioning as a producer for MSK.**

**Example-**

****

Note that we only discussed clickstreamtopic here, because EC2 will write clicking data into it.

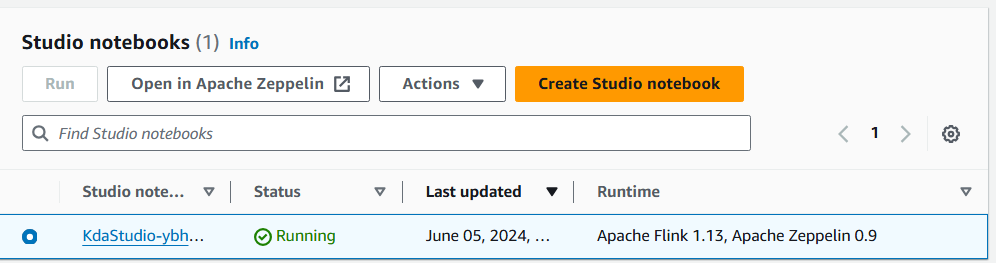
On the other hand,

Catalog details topic will get it’s data directly from the S3 bucket (which had initially gotten data from EC2 via KDS, in lab 1: so here that EC2->KDS->S3 step has been skipped in the lab)

**Task 3: Import the Zeppelin notebook**

3.1 Go back to the step where we waited (in task 1 - after 1.4)

3.2 the notebook is ready, click on view in zeppelin

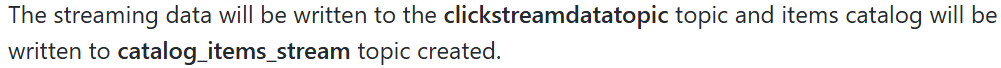


3.3 Import and open this



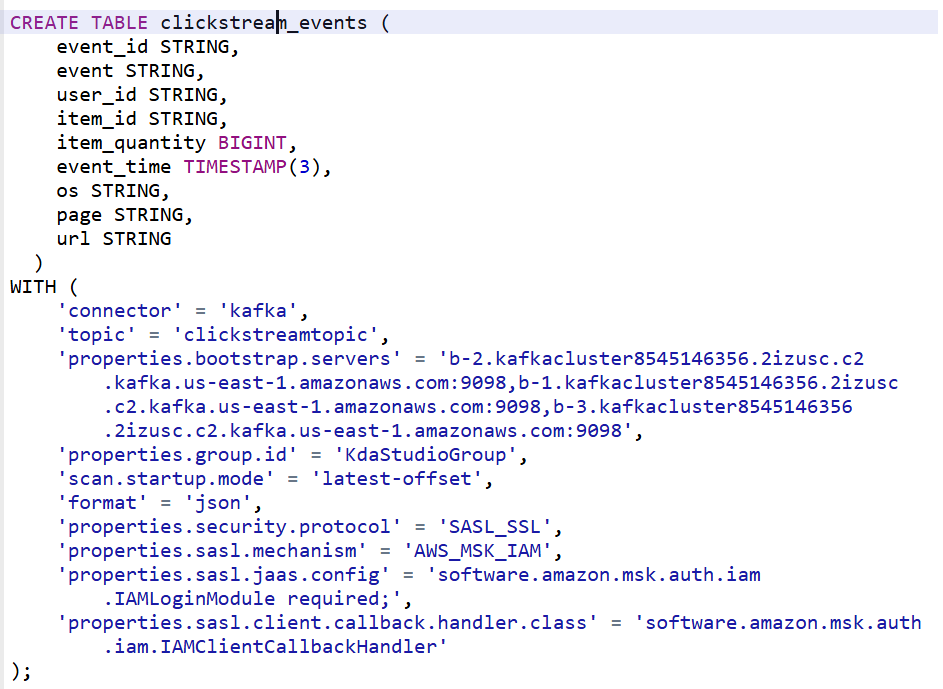
**Task 4: Analytics development with Zeppelin notebook**

Configure Managed Apache Flink as a consumer to query and analyze the streaming data.



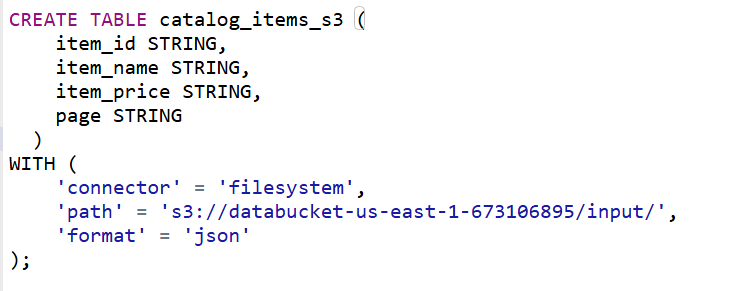
4.1 Creating the table in which data from clickstreamtopic will be stored (linked using bootstrap server value given in lab)

Here, data in clickstream\_events table comes from clickstreamtopic topic in Kafka, which got it’s data from EC2 (as a producer). So here Flink (via zeppelin table) is acting as a consumer for the data stored in clickstreamtopic topic.



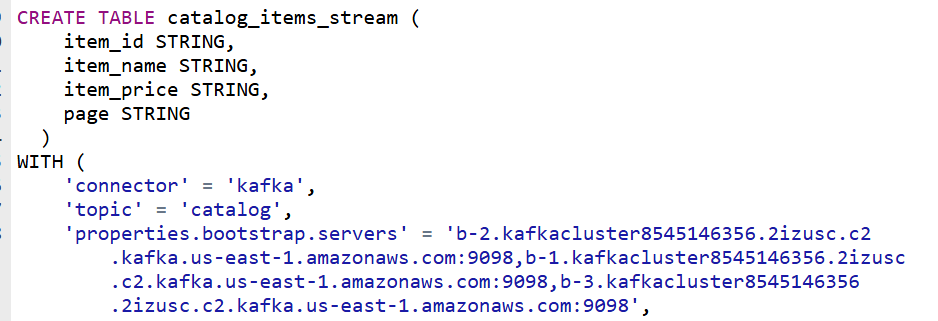
4.2 now we create table for Catalog details from S3

Connected via S3 bucket path

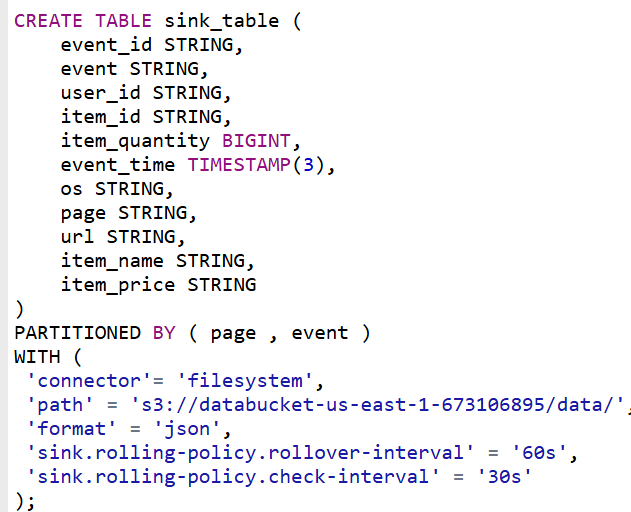


We also create catalog items “stream”

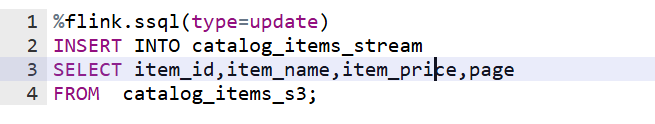
We’ll insert data into this stream from the catalog\_s3 table, in further steps.



4.3 Creating the sink table, where the processed data will be written. The table schema combines fields from both clickstream\_events and catalog\_items\_stream. The data is partitioned by page and event.



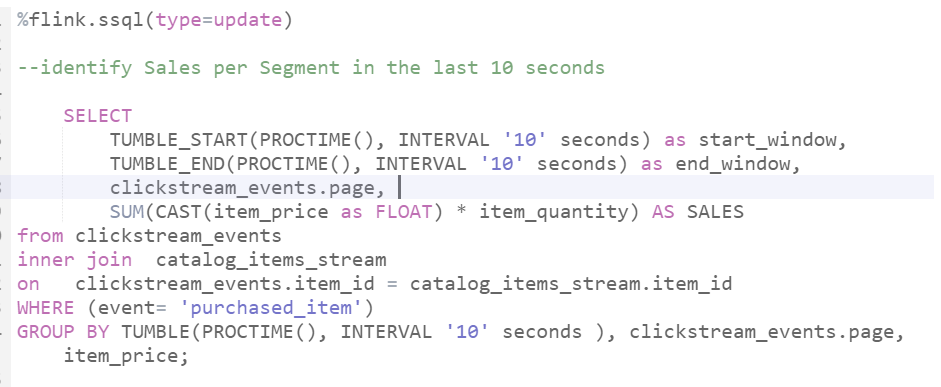
4.4 INSERT INTO the catalog\_items\_stream table the items catalog stored in Amazon S3.





4.5 Join the streaming data and catalog data

4.6 Group the data to get sales per 10 secs



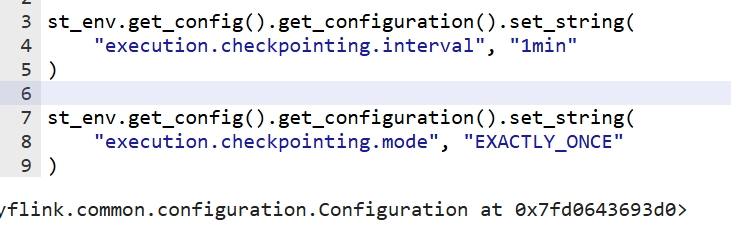
**VISUALIZATION PART BROKEN**

**Lab threw error at this output, so output cannot be shown**

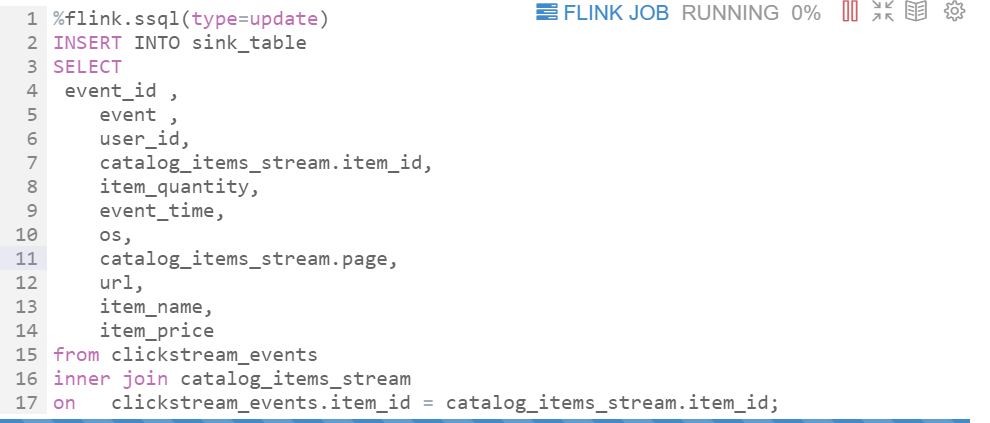
**Skipping…**

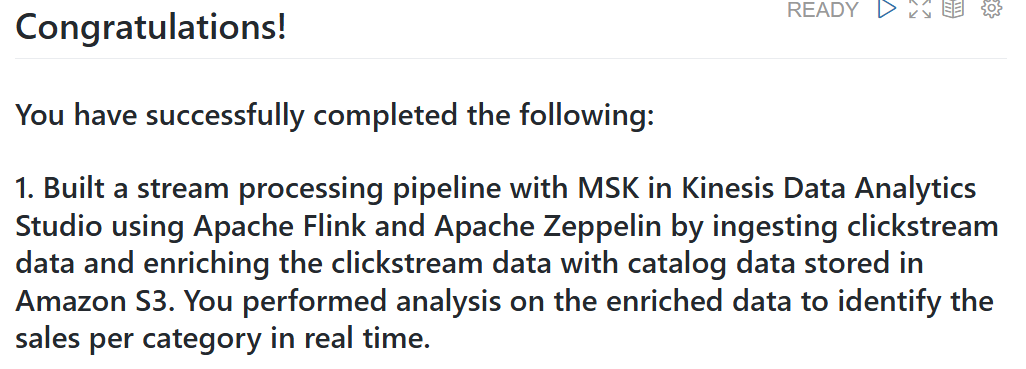
4.7 Enable checkpointing

Checkpointing needs to be enabled to write data to Amazon S3.

****

4.8 Store data into S3 (collection of streaming and catalog data, at once)

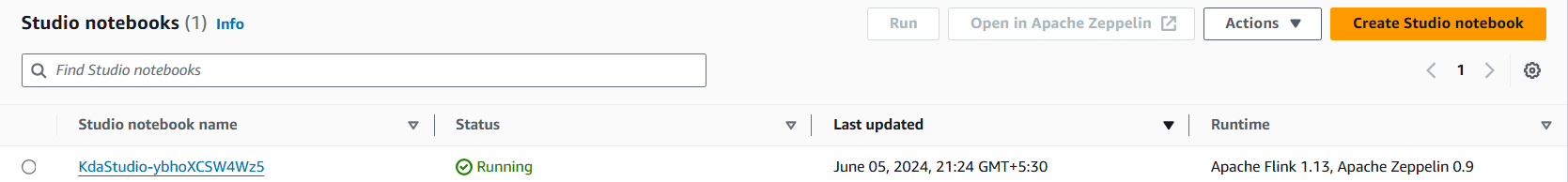




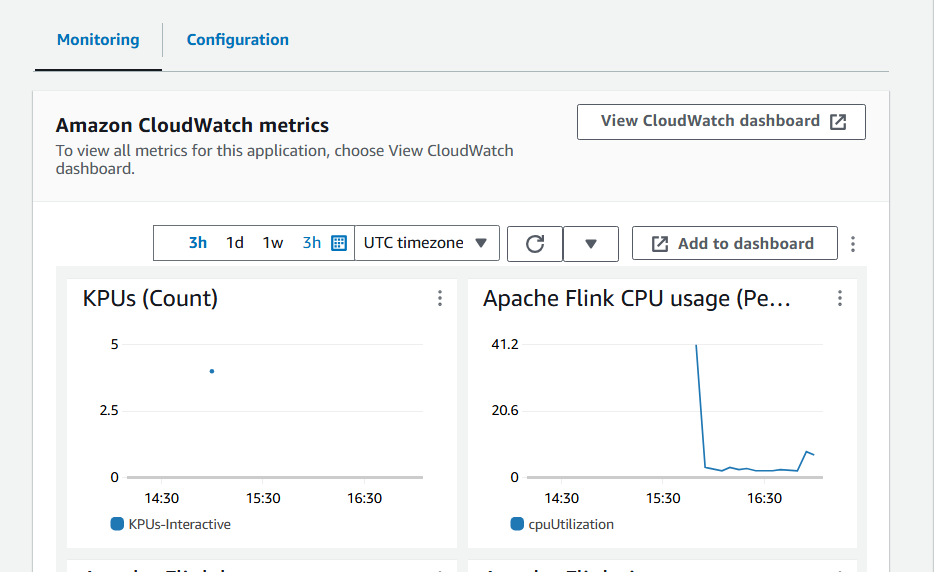
**Task 5: Build and deploy the streaming pipeline as an application**

(build and deploy the notebook into an application from Managed Apache Flink Studio.)

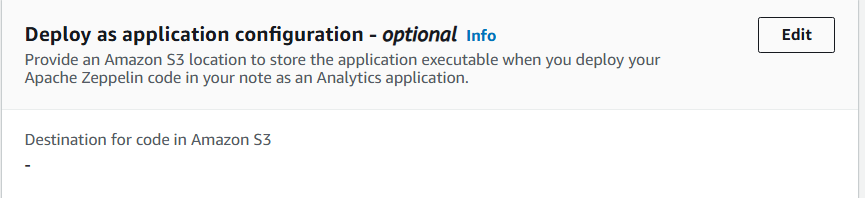
5.1 Go to kinesis console tab, and select the notebook



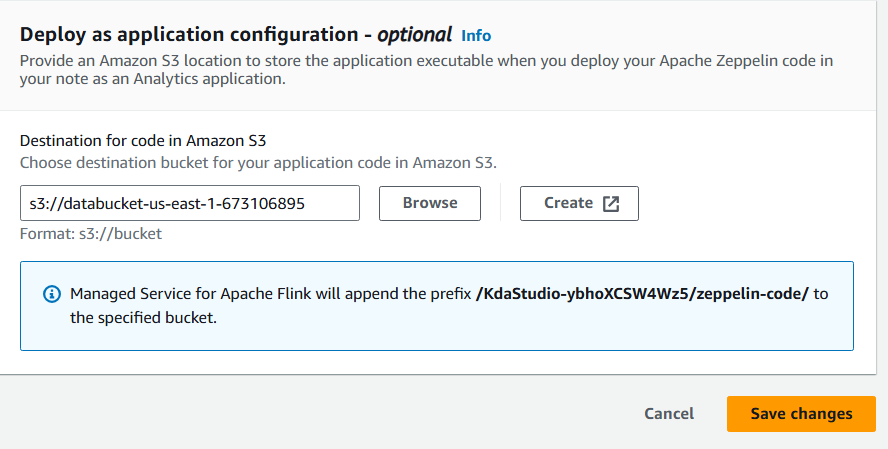
5.2 Click on configuration



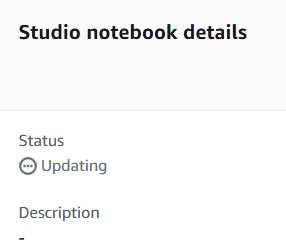
5.3 Click edit



5.4 Choose this S3 bucket as destination for code in S3



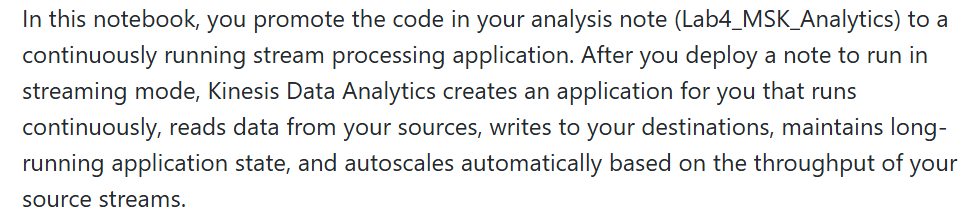
5.5 wait for status change to running



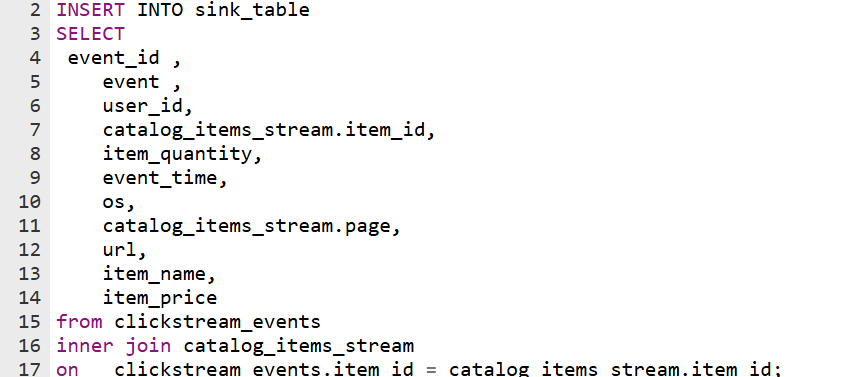
5.6 Import the other notebook in zipprlin

**BUILD AND DEPLOY APPLICATION**

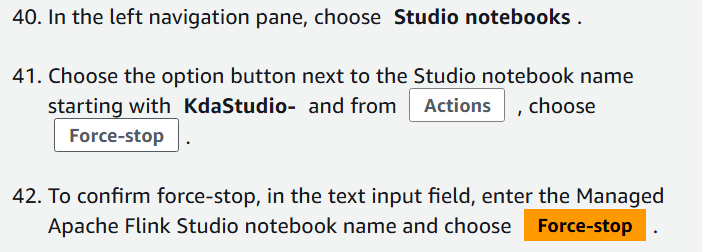
5.7 summary

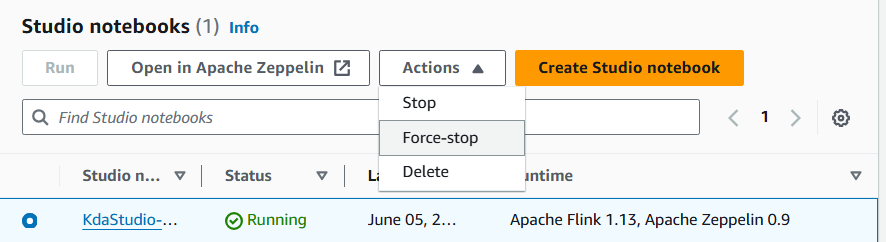


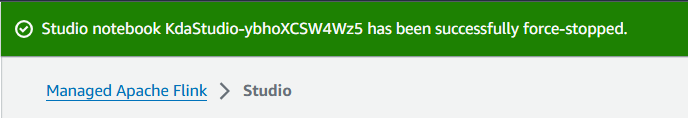
5.8 only the last step of prev notebook is done here (to merge clickstream and catalog data)



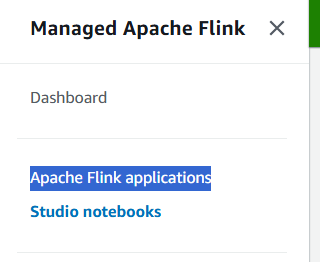
5.9 Lab guide tells to force-stop the notebook





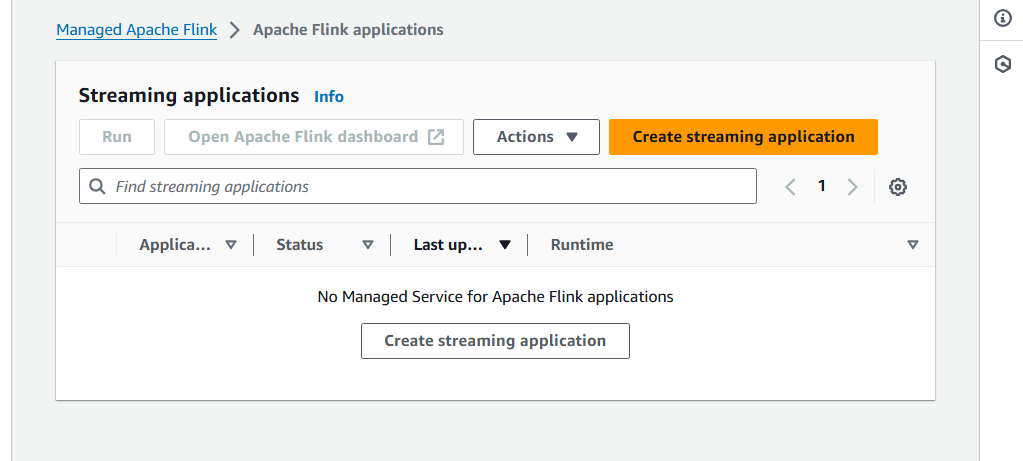


5.10 Choose from left pane



5.11 **ISSUE IN LAB HERE -**

**No streaming application found**

****

**Apart from this, whole lab goes smoothly.**