Lab 4 - MSK Streaming Pipeline and Application Deployment

Objectives:

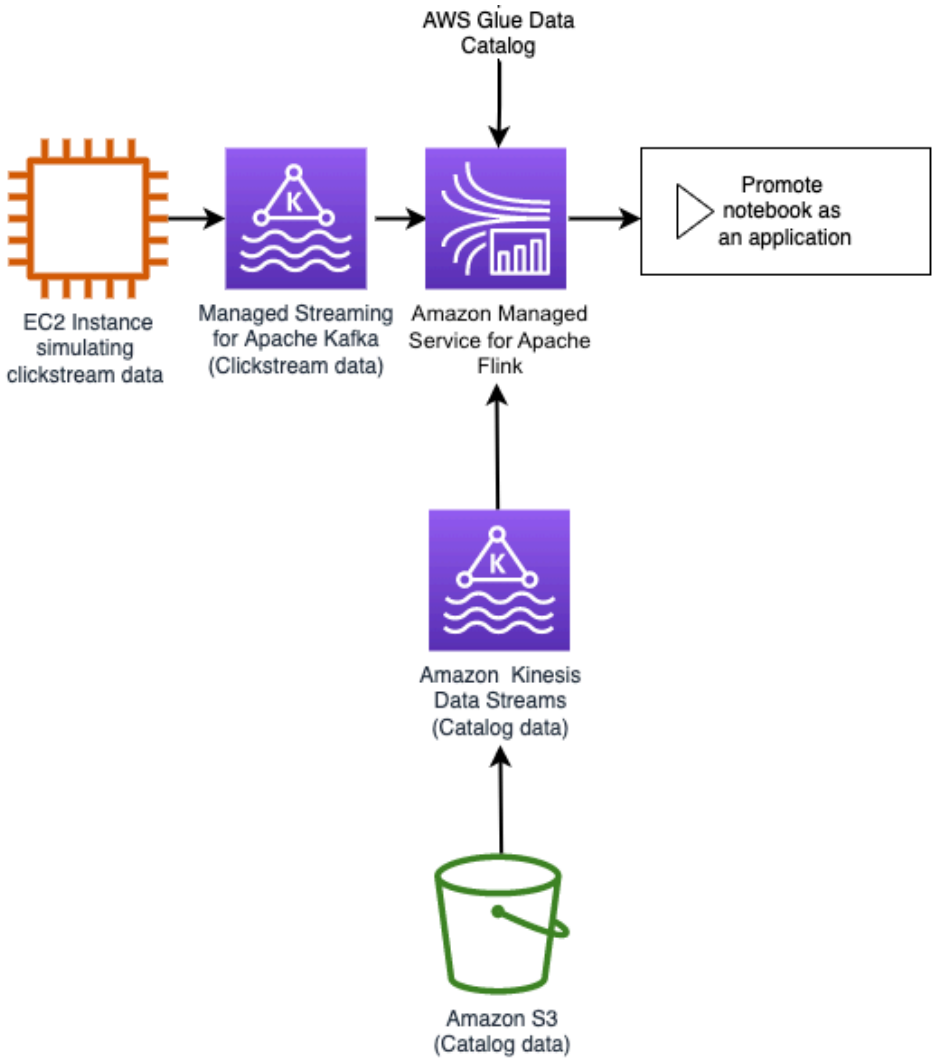
1. Build a real-time Amazon MSK streaming analytics pipeline in Managed Apache Flink Studio using Apache Flink and Apache Zeppelin.
2. Visualize the output.
3. Build and deploy the Zeppelin notebook as a long-standing application with the ability to durably store data in Amazon S3.

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.



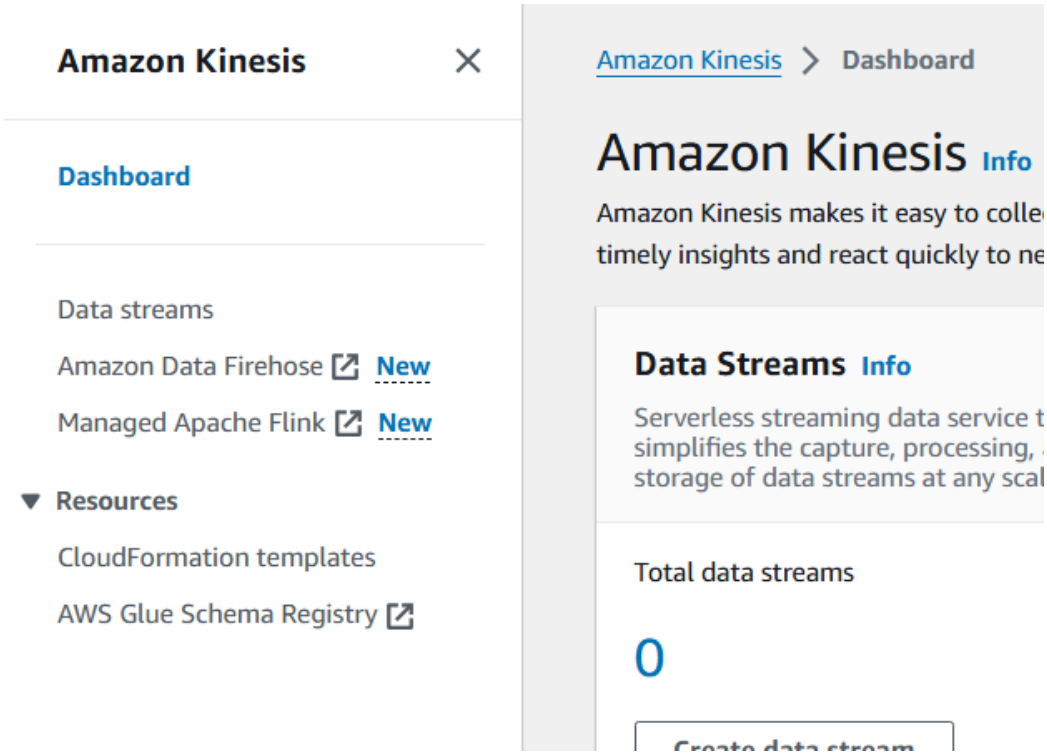
Doubt - how is kafka/msk getting integrated into this? in 2nd lab, we used KDS to get data from clickstream. Why are we using MSK here?

Ans [GPT] -

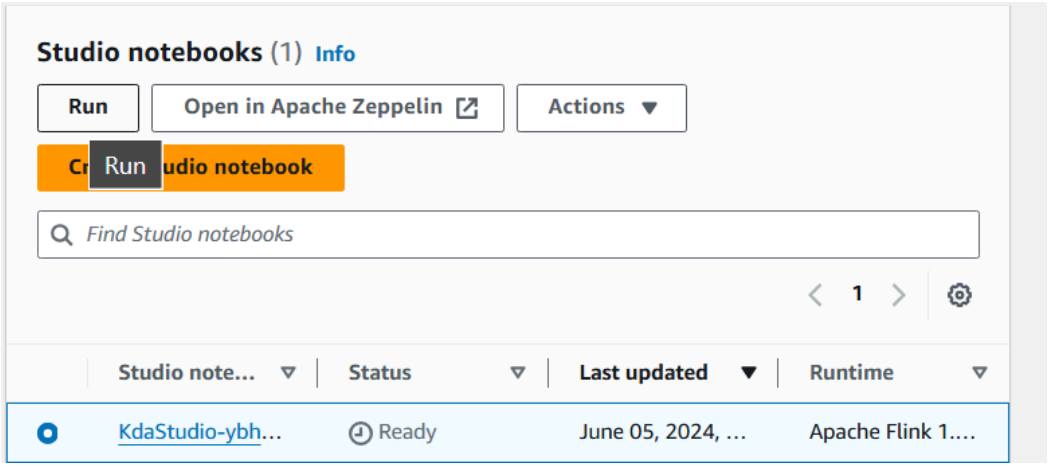
1. EC2 acts as a producer for Kafka here.
2. Flink acts as a consumer.
3. Realtime processing, as MSK does
4. In summary, the decision to use Kafka/MSK in this lab scenario instead of KDS likely stems from the advantages Kafka offers in terms of ecosystem compatibility, performance, flexibility in data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

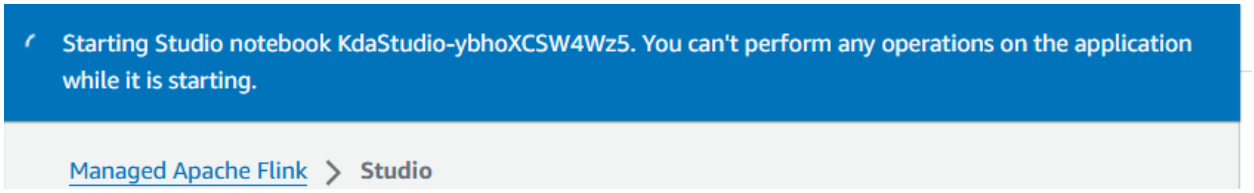


1.3 Meanwhile, download these two zeppelin files:

8. Save the Lab4_MSK_Analytics.zpln file to your local machine.

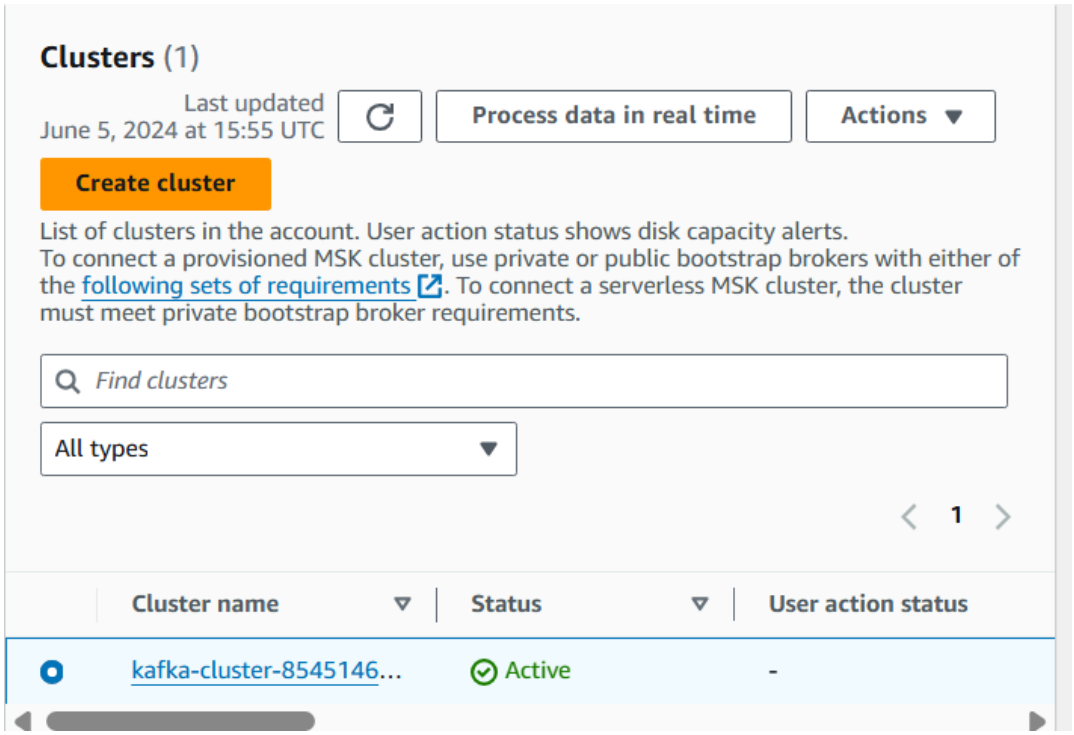
9. Save the Lab4_MSK_Application.zpln file to your local machine.

1.4 Wait for this to complete

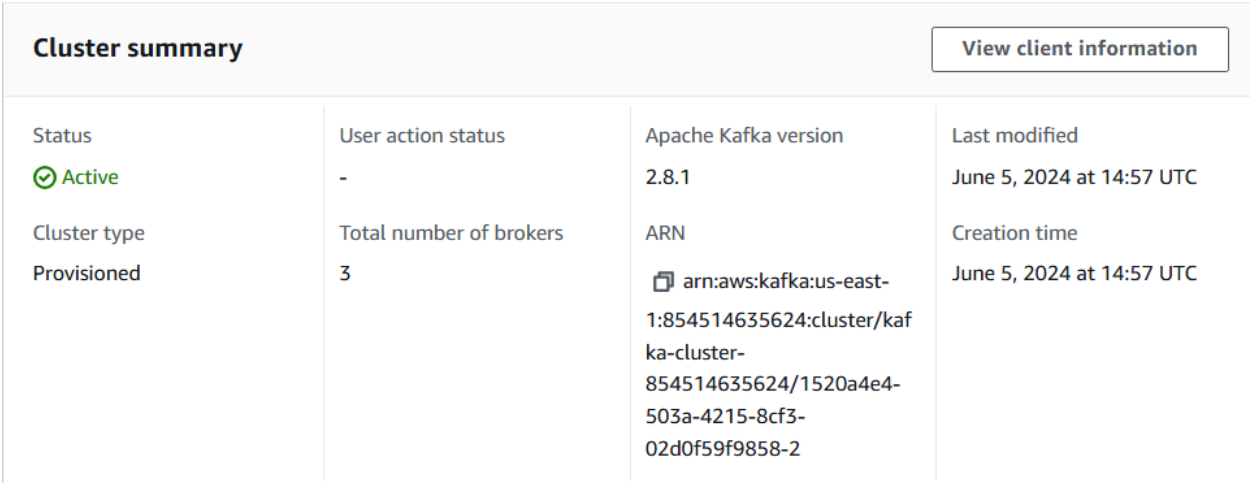


Task 2: Create topics in the MSK cluster and simulate clickstream data generation

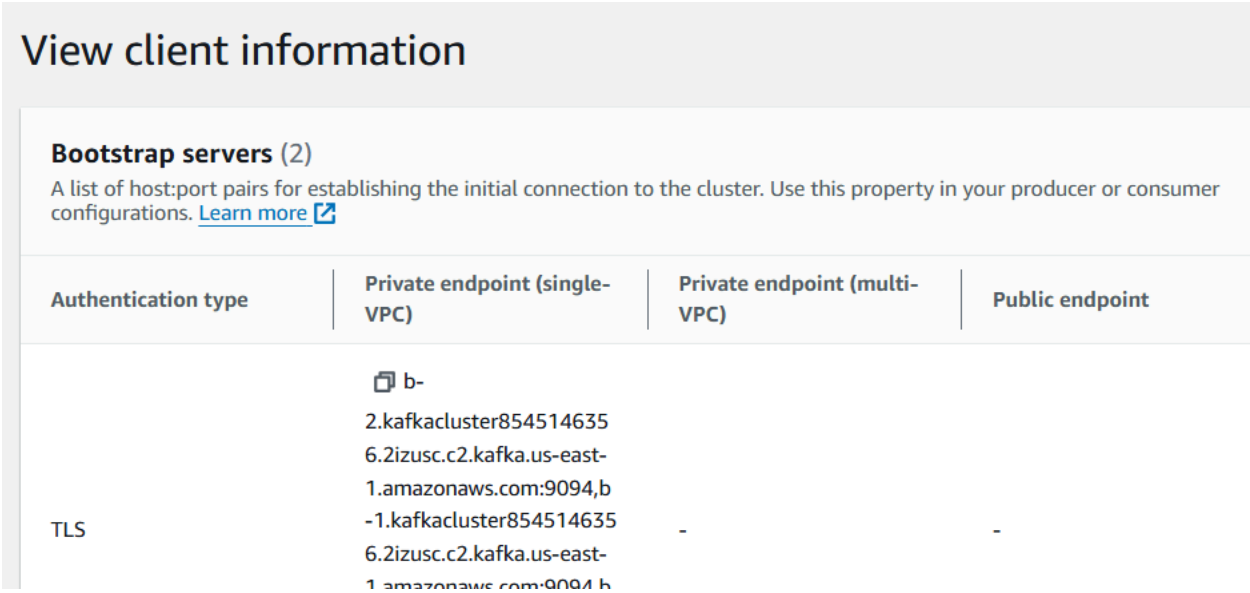
2.1 Open MSK in console (new tab)
2.2 Click on the link for pre created cluster



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

```
cd /home/ssm-user; source ~/.bashrc; source /home/kafka/bin/msk.env; source /hom
e/kafka/bin/msktopic.env; cd /home/app;
sh-4.2$ cd /home/ssm-user; source ~/.bashrc; source /home/kafka/bin/msk.env; sou
rce /home/kafka/bin/msktopic.env; cd /home/app;
Created topic clickstreamtopic.
Created topic catalog.
__amazon_msk_canary
__consumer_offsets
catalog
clickstreamtopic
[ssm-user@ip-10-0-1-183 app]$
```

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

2.7 SIMULATE CLICKSTREAM DATA GENERATION

```
[ssm-user@ip-10-0-1-183 app]$ python3 clickmskgenerator_items.py $MSK_BOOTSTRAP clickstreamtopic 1
```

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

EC2 is now functioning as a producer for MSK.
Example-

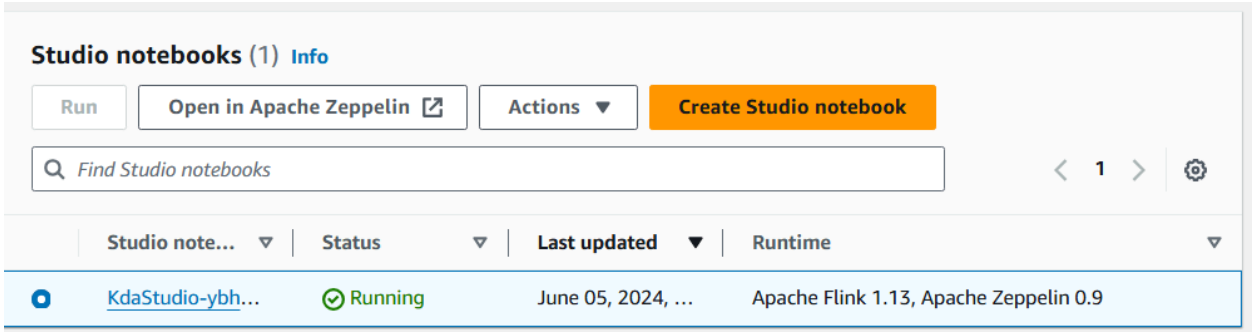
```
{'event_id': 'd5aa8bdb49f7cf8c6ab52678e98ba409', 'event': 'entered_payment_method', 'user_id': 8, 'item_id': 41, 'item_quantity': 0, 'event_time': '2024-06-05 16:11:16.499884', 'os
: 'ios', 'page': 'home', 'url': 'www.example.com'}
sent event to Kafka! topic clickstreamtopic partition 1 offset 53
```

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

Lab4_MSK_Analytics

Task 4: Analytics development with Zeppelin notebook

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

The streaming data will be written to the **clickstreamdatatopic** topic and items catalog will be written to **catalog_items_stream** topic created.

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.

```
CREATE TABLE clickstream_events (
    event_id STRING,
    event STRING,
    user_id STRING,
    item_id STRING,
    item_quantity BIGINT,
    event_time TIMESTAMP(3),
    os STRING,
    page STRING,
    url STRING
  )
WITH (
    'connector' = 'kafka',
    'topic' = 'clickstreamtopic',
    'properties.bootstrap.servers' = 'b-2.kafkacluster8545146356.2izusc.c2
        .kafka.us-east-1.amazonaws.com:9098,b-1.kafkacluster8545146356.2izusc
        .c2.kafka.us-east-1.amazonaws.com:9098,b-3.kafkacluster8545146356
        .2izusc.c2.kafka.us-east-1.amazonaws.com:9098',
    'properties.group.id' = 'KdaStudioGroup',
    'scan.startup.mode' = 'latest-offset',
    'format' = 'json',
    'properties.security.protocol' = 'SASL_SSL',
    'properties.sasl.mechanism' = 'AWS_MSK_IAM',
    'properties.sasl.jaas.config' = 'software.amazon.msk.auth.iam
        .IAMLoginModule required;',
    'properties.sasl.client.callback.handler.class' = 'software.amazon.msk.auth
        .iam.IAMClientCallbackHandler'
);
```

4.2 now we create table for Catalog details from S3
Connected via S3 bucket path

```
CREATE TABLE catalog_items_s3 (
    item_id STRING,
    item_name STRING,
    item_price STRING,
    page STRING
  )
WITH (
    'connector' = 'filesystem',
    'path' = 's3://databucket-us-east-1-673106895/input/',
    'format' = 'json'
);
```

We also create catalog items “stream”

We’ll insert data into this stream from the catalog_s3 table, in further steps.

```
CREATE TABLE catalog_items_stream (
    item_id STRING,
    item_name STRING,
    item_price STRING,
    page STRING
  )
WITH (
    'connector' = 'kafka',
    'topic' = 'catalog',
    'properties.bootstrap.servers' = 'b-2.kafkacluster8545146356.2izusc.c2
        .kafka.us-east-1.amazonaws.com:9098,b-1.kafkacluster8545146356.2izusc
        .c2.kafka.us-east-1.amazonaws.com:9098,b-3.kafkacluster8545146356
        .2izusc.c2.kafka.us-east-1.amazonaws.com:9098',
```

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.

```
CREATE TABLE sink_table (
    event_id STRING,
    event STRING,
    user_id STRING,
    item_id STRING,
    item_quantity BIGINT,
    event_time TIMESTAMP(3),
    os STRING,
    page STRING,
    url STRING,
    item_name STRING,
    item_price STRING
)
PARTITIONED BY ( page , event )
WITH (
 'connector'= 'filesystem',
 'path' = 's3://databucket-us-east-1-673106895/data/',
 'format' = 'json',
 'sink.rolling-policy.rollover-interval' = '60s',
 'sink.rolling-policy.check-interval' = '30s'
);
```

4.4 INSERT INTO the catalog_items_stream table the items catalog stored in Amazon S3.

```
1 %flink.ssql(type=update)
2 INSERT INTO catalog_items_stream
3 SELECT item_id,item_name,item_price,page
4 FROM  catalog_items_s3;
```

```
Insertion successfully.
```

4.5 Join the streaming data and catalog data
4.6 Group the data to get sales per 10 secs

```
%flink.ssql(type=update)

--identify Sales per Segment in the last 10 seconds

    SELECT
        TUMBLE_START(PROCTIME(), INTERVAL '10' seconds) as start_window,
        TUMBLE_END(PROCTIME(), INTERVAL '10' seconds) as end_window,
        clickstream_events.page,
        SUM(CAST(item_price as FLOAT) * item_quantity) AS SALES
from clickstream_events
inner join  catalog_items_stream
on   clickstream_events.item_id = catalog_items_stream.item_id
WHERE (event= 'purchased_item')
GROUP BY TUMBLE(PROCTIME(), INTERVAL '10' seconds ), clickstream_events.page,
    item_price;
```

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.

```
3 st_env.get_config().get_configuration().set_string(
4     "execution.checkpointing.interval", "1min"
5 )
6
7 st_env.get_config().get_configuration().set_string(
8     "execution.checkpointing.mode", "EXACTLY_ONCE"
9 )
```

```
/flink.common.configuration.Configuration at 0x7fd0643693d0>
```

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

FLINK JOB RUNNING 0%

```
%flink.ssql(type=update)
INSERT INTO sink_table
SELECT
 event_id ,
    event ,
    user_id,
    catalog_items_stream.item_id,
    item_quantity,
    event_time,
    os,
    catalog_items_stream.page,
    url,
    item_name,
    item_price
from clickstream_events
inner join catalog_items_stream
on   clickstream_events.item_id = catalog_items_stream.item_id;
```

READY

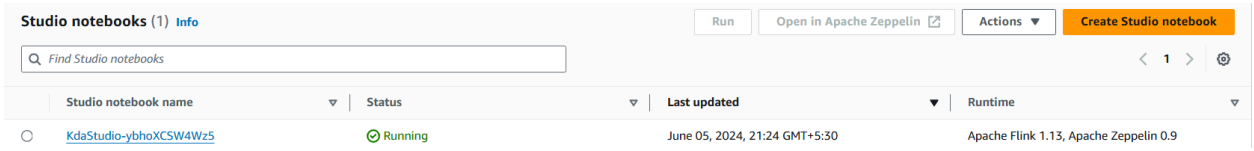
Congratulations!

You have successfully completed the following:

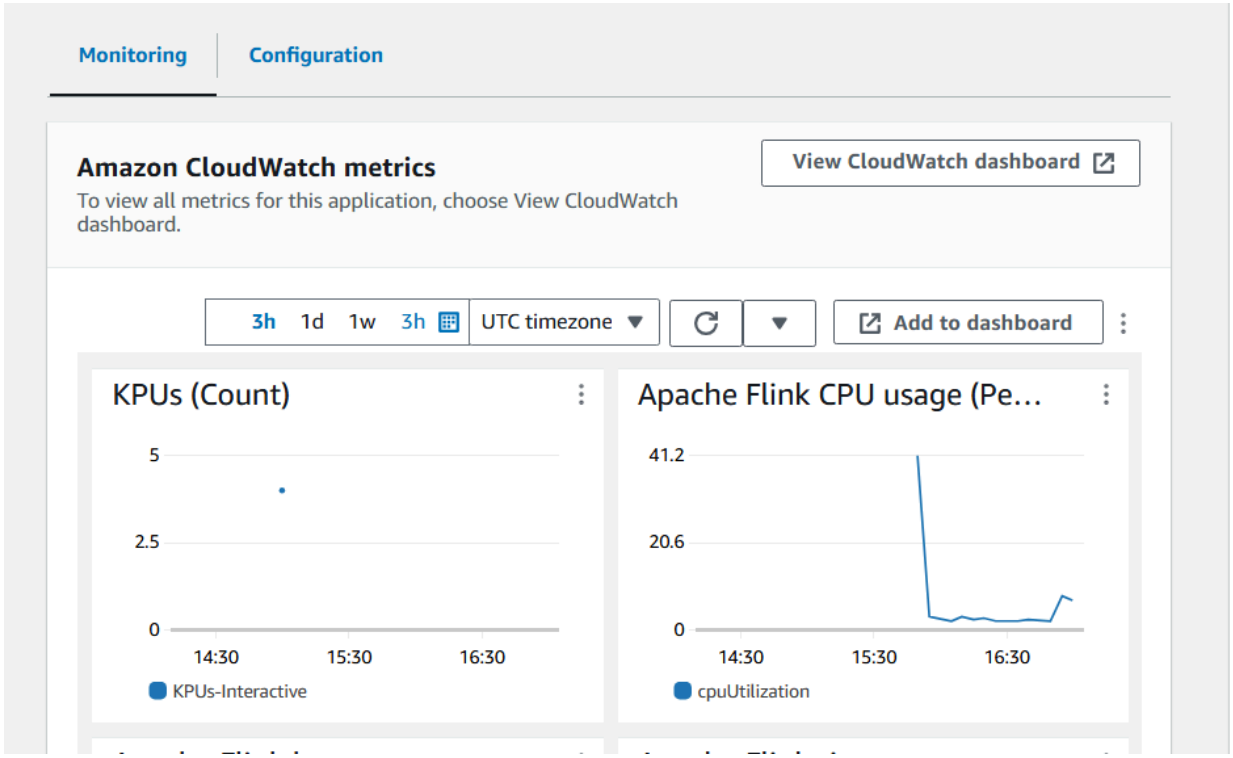
1. Built a stream processing pipeline with MSK in Kinesis Data Analytics Studio using Apache Flink and Apache Zeppelin by ingesting clickstream data and enriching the clickstream data with catalog data stored in Amazon S3. You performed analysis on the enriched data to identify the sales per category in real time.

Task 5: Build and deploy the streaming pipeline as an application
(build and deploy the notebook into an application from Managed Apache Flink Studio.)

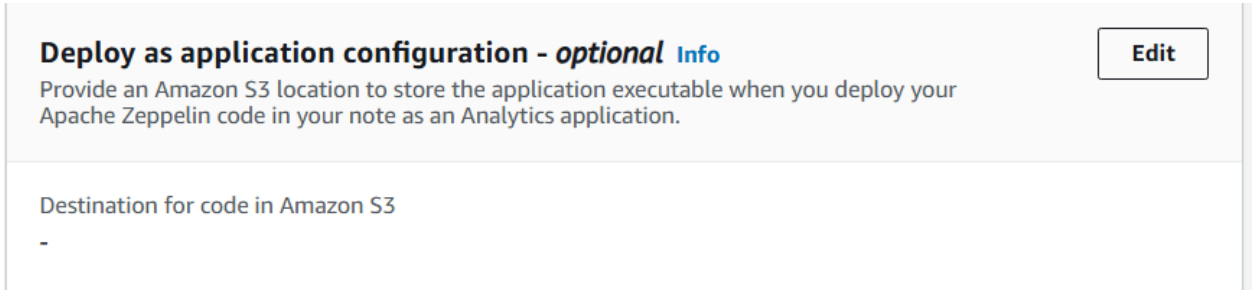
5.1 Go to kinesis console tab, and select the notebook



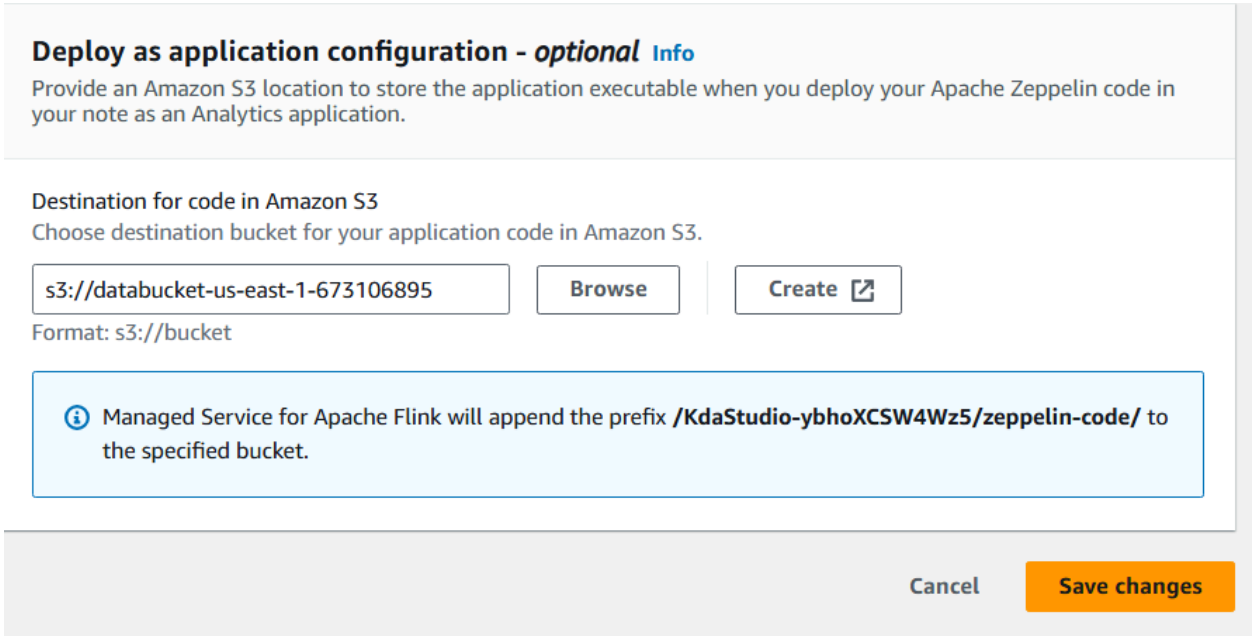
5.2 Click on configuration



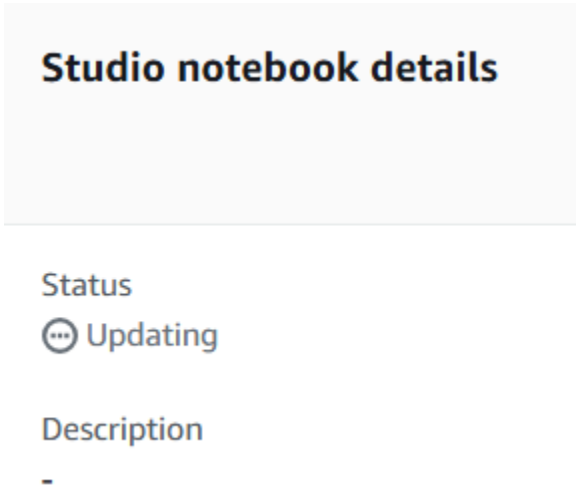
5.3 Click edit



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

In this notebook, you promote the code in your analysis note (Lab4_MSK_Analytics) to a continuously running stream processing application. After you deploy a note to run in streaming mode, Kinesis Data Analytics creates an application for you that runs continuously, reads data from your sources, writes to your destinations, maintains long-running application state, and autoscales automatically based on the throughput of your source streams.

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

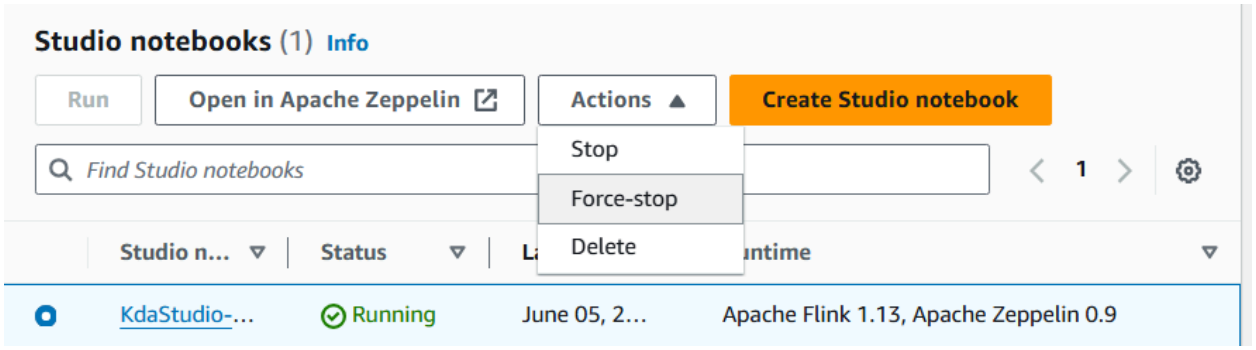
```
INSERT INTO sink_table
SELECT
 event_id ,
    event ,
    user_id,
    catalog_items_stream.item_id,
    item_quantity,
    event_time,
    os,
    catalog_items_stream.page,
    url,
    item_name,
    item_price
from clickstream_events
inner join catalog_items_stream
on   clickstream_events.item_id = catalog_items_stream.item_id;
```

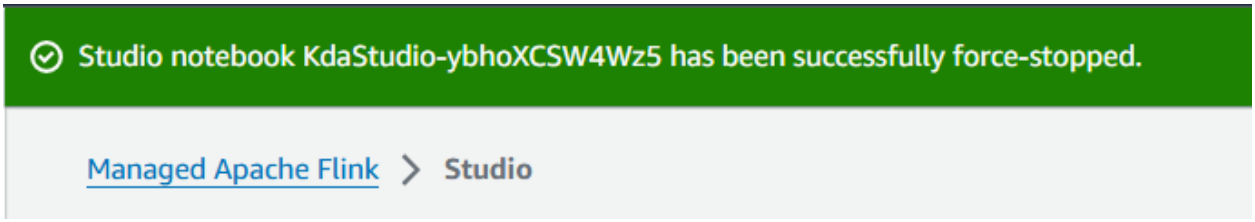
5.9 Lab guide tells to force-stop the notebook

40. In the left navigation pane, choose **Studio notebooks** .

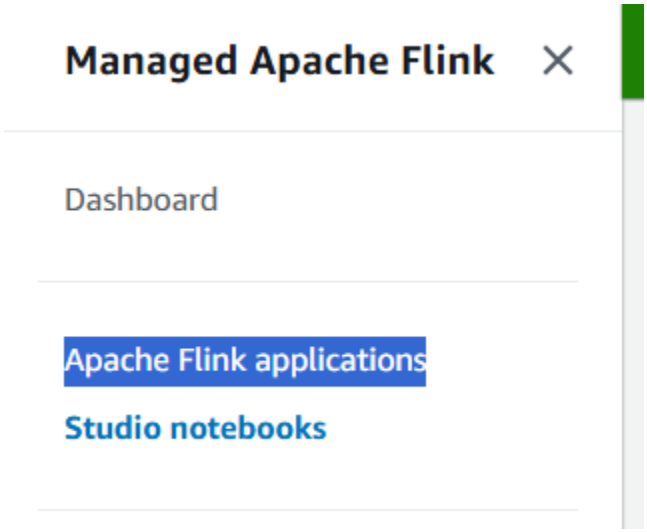
41. Choose the option button next to the Studio notebook name starting with **KdaStudio-** and from **Actions** , choose **Force-stop** .

42. To confirm force-stop, in the text input field, enter the Managed Apache Flink Studio notebook name and choose **Force-stop** .



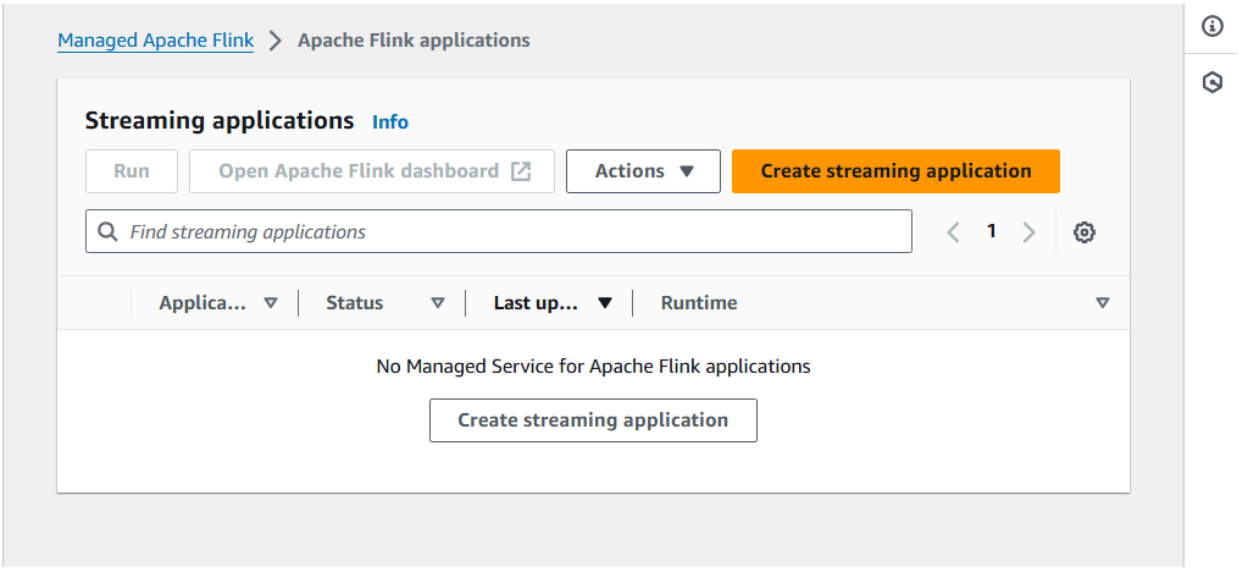


5.10 Choose from left pane



5.11 **ISSUE IN LAB HERE -**

No streaming application found



Apart from this, whole lab goes smoothly.