Lab: Real-Time Clickstream Anomaly Detection Kinesis Analytics

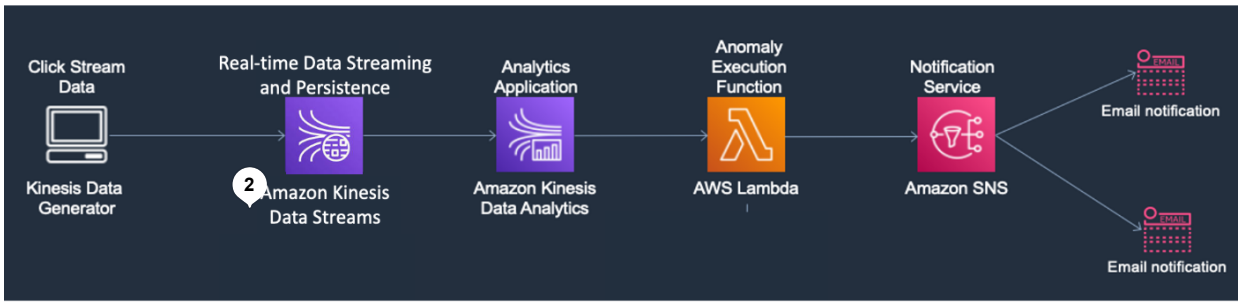
Introduction

This guide helps you complete Real-Time Clickstream Anomaly Detection using Amazon Kinesis Data Analytics.
Analyzing web log traffic to gain insights that drive business decisions has historically been performed using batch processing. Although effective, this approach results in delayed responses to emerging trends and user activities. There are solutions that process data in real-time using streaming and micro-batching technologies, but they can be complex to set up and maintain. Amazon Kinesis Data Analytics is a managed service that makes it easy to identify and respond to changes in data behavior in real-time.

Steps:

- Set up an Amazon Analytics Studio Application through CloudFormation stack deployment
- Generate real time website traffic using Amazon Kinesis Data Generator (KDG)
- Perform real-time Data Analytics
- Environment Cleanup
- Appendix: Anomaly Detection Scripts

In the Kinesis prelab setup, you fulfilled the prerequisites for this lab. In this lab, you will create the following Kinesis Data Analytics pipeline.

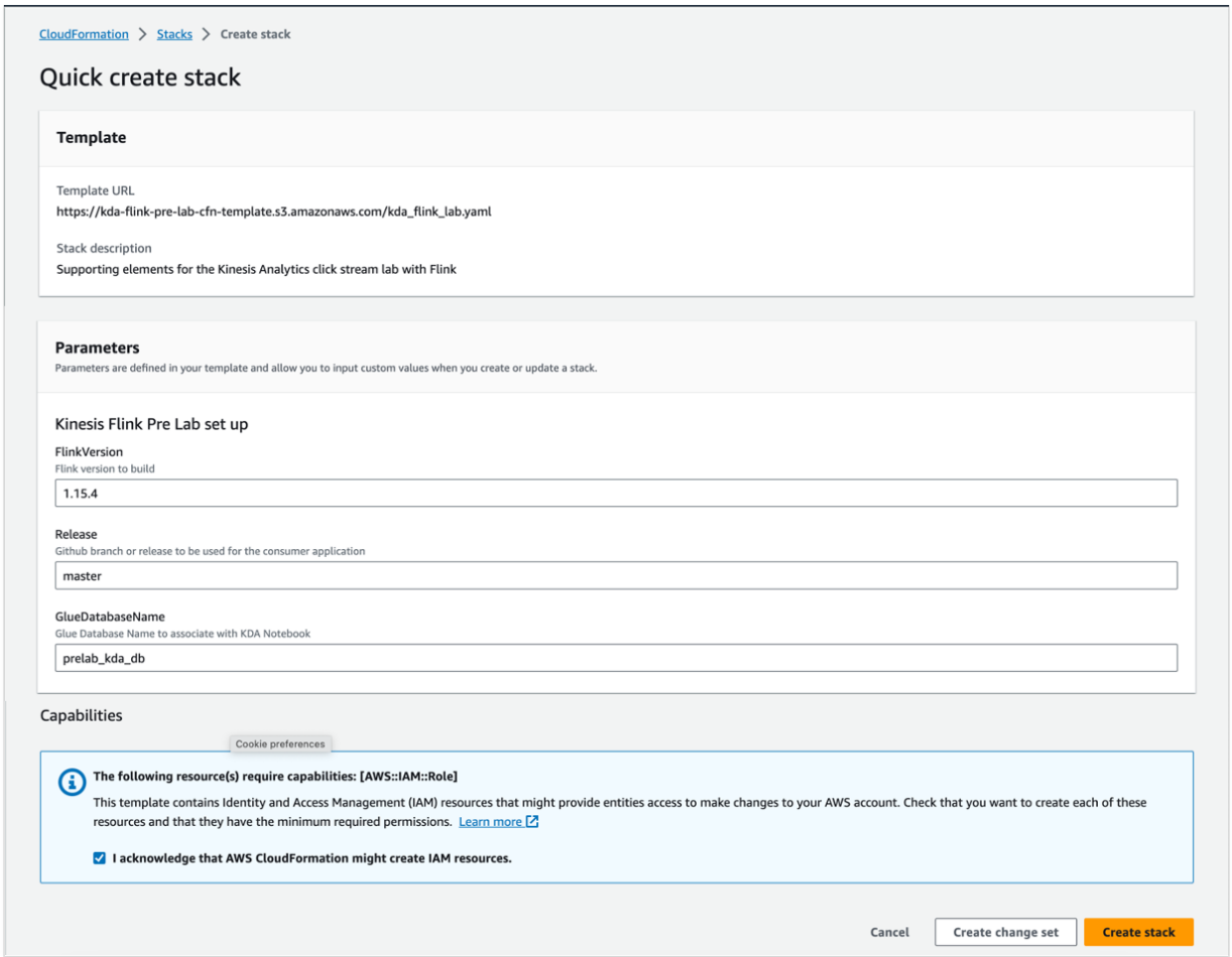


Set up an Amazon Analytics Studio Application through CloudFormation stack deployment

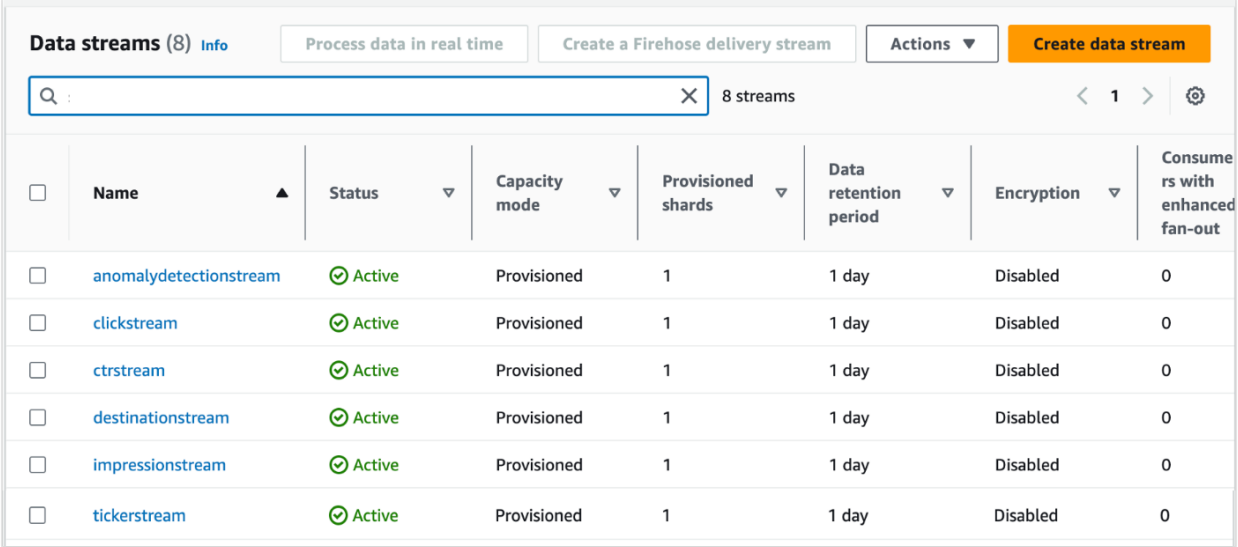
1. Click the **Deploy to AWS** button below to stand up the pre-lab workshop infrastructure

Deploy to AWS

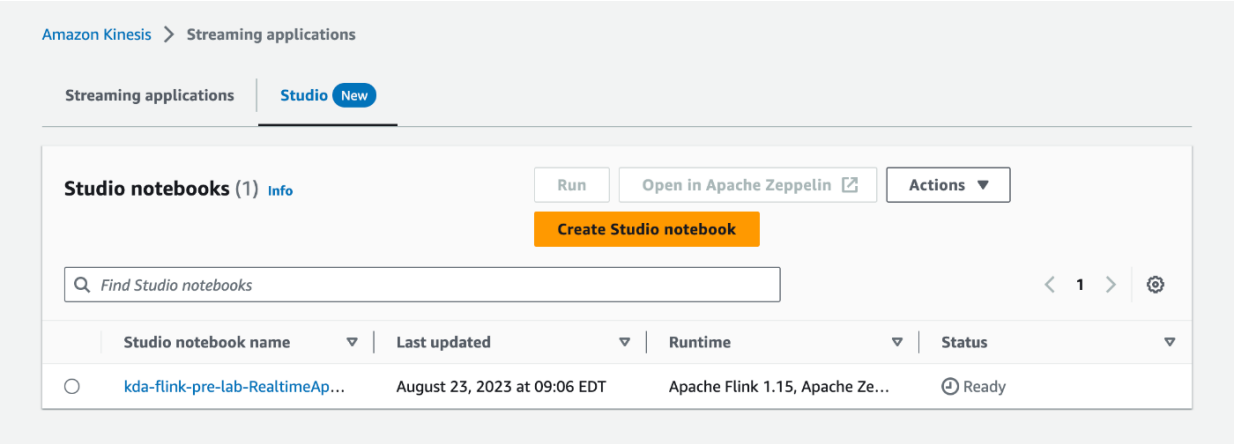
2. The button above will open a “Quick create stack” form, please accept the default parameters, select the checkbox to acknowledge new IAM role creation and select Create Stack to run the Amazon CloudFormation Template



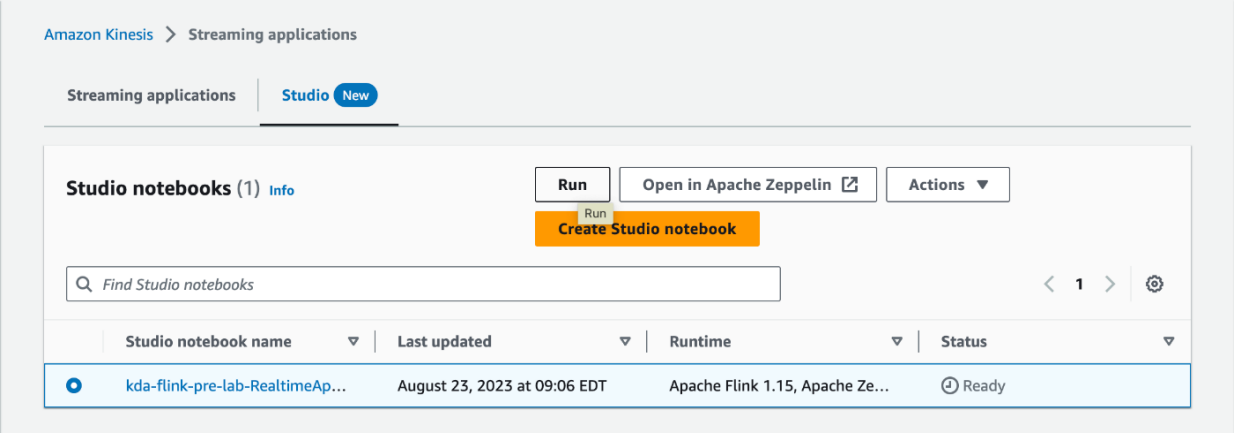
3. The stack will create six Amazon Kinesis Data Streams in the Amazon Kinesis Console -
    a. tickerstream – the raw stream to send the initial traffic
    b. clickstream – captures the number of clicks
    c. impressionstream – captures the number of impressions
    d. ctrstream – captures the calculated click through rate
    e. destinationstream – captures the anomaly scores
    f. anomalydetectionstream – captures the records with anomaly score greater than 1

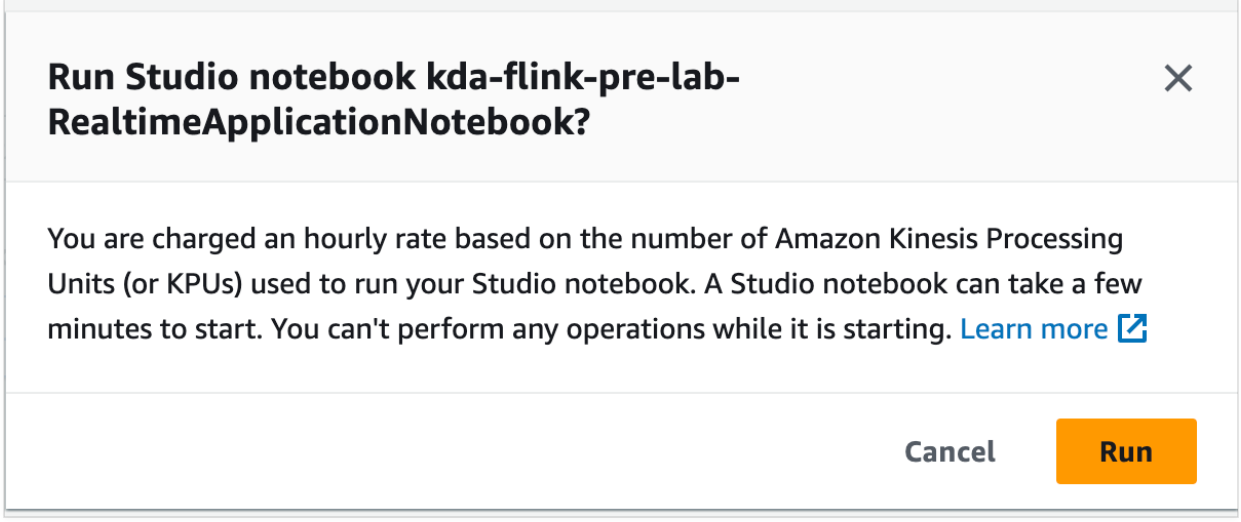


4. The template would also create a Amazon Kinesis Data Analytics Studio application called kda-flink-prelab-RealtimeApplicationNotebook in the Amazon Kinesis Console → Studio tab. We will write interactive Studio Notebook in Apache Zeppelin for real time data analysis.



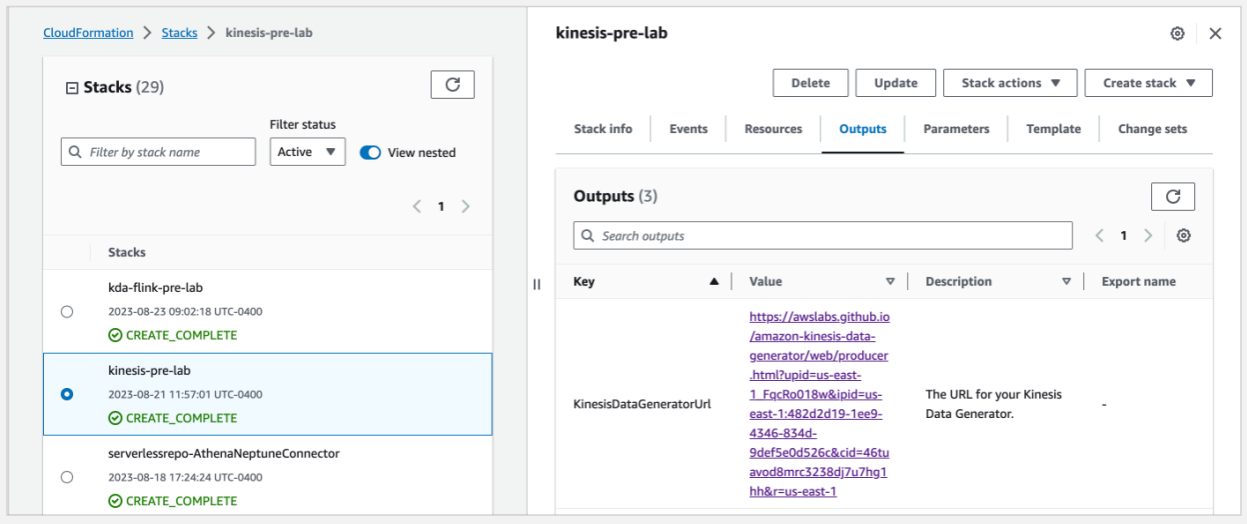
5. Run the Studio Application by selecting the kda-flink-prelab-RealtimeApplicationNotebook under Studio tab. Select “Run” again on the next screen.





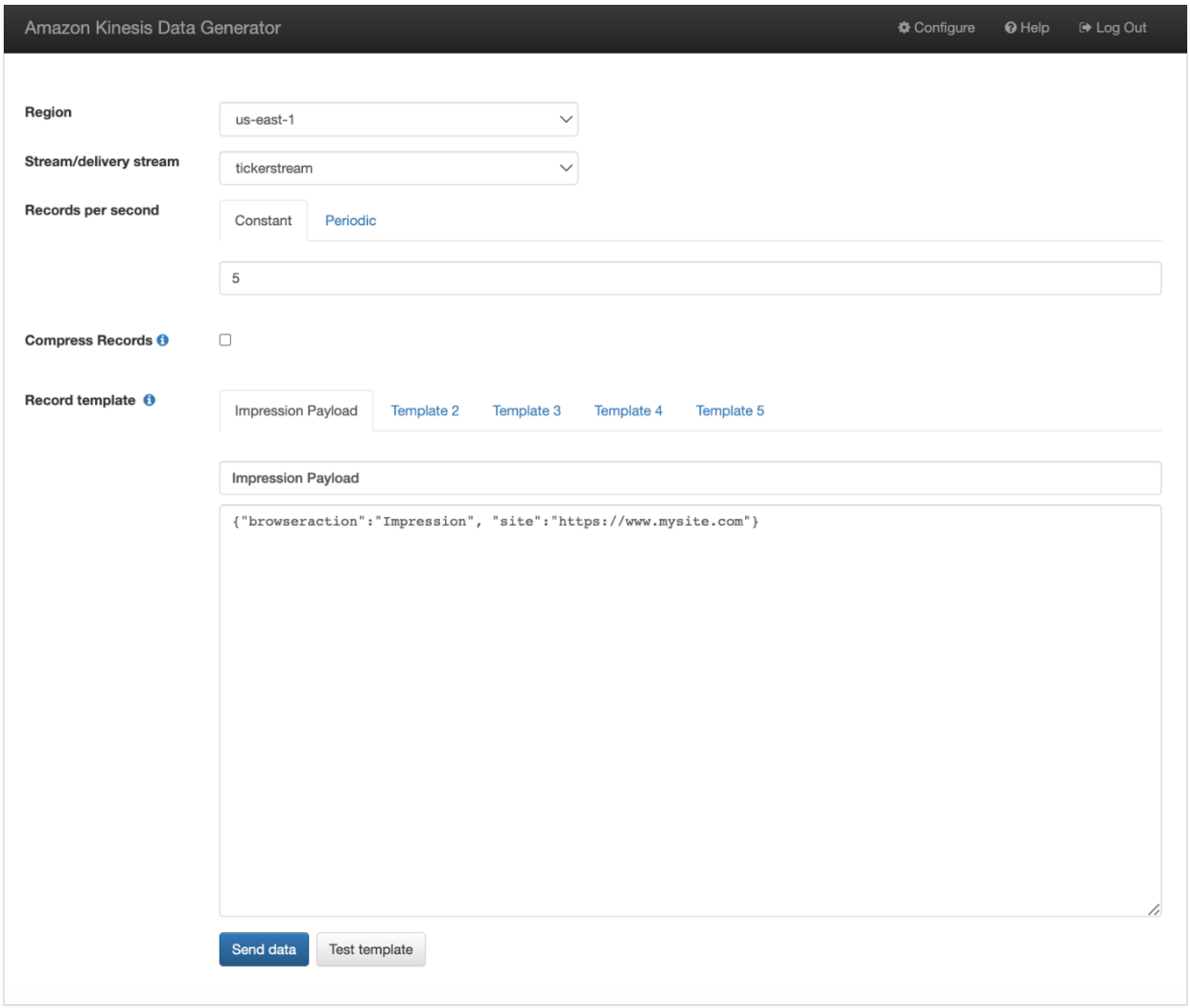
Generate real time website traffic using Amazon Kinesis Data Generator (KDG)

1. Navigate to the Amazon CloudFormation console in your AWS account, click on the **Kinesis-pre-lab** stack created during the **Streaming Data Analytics Prelab setup**.
2. Go to the Outputs tab of the stack to get the Kinesis Data Generator link as shown below:



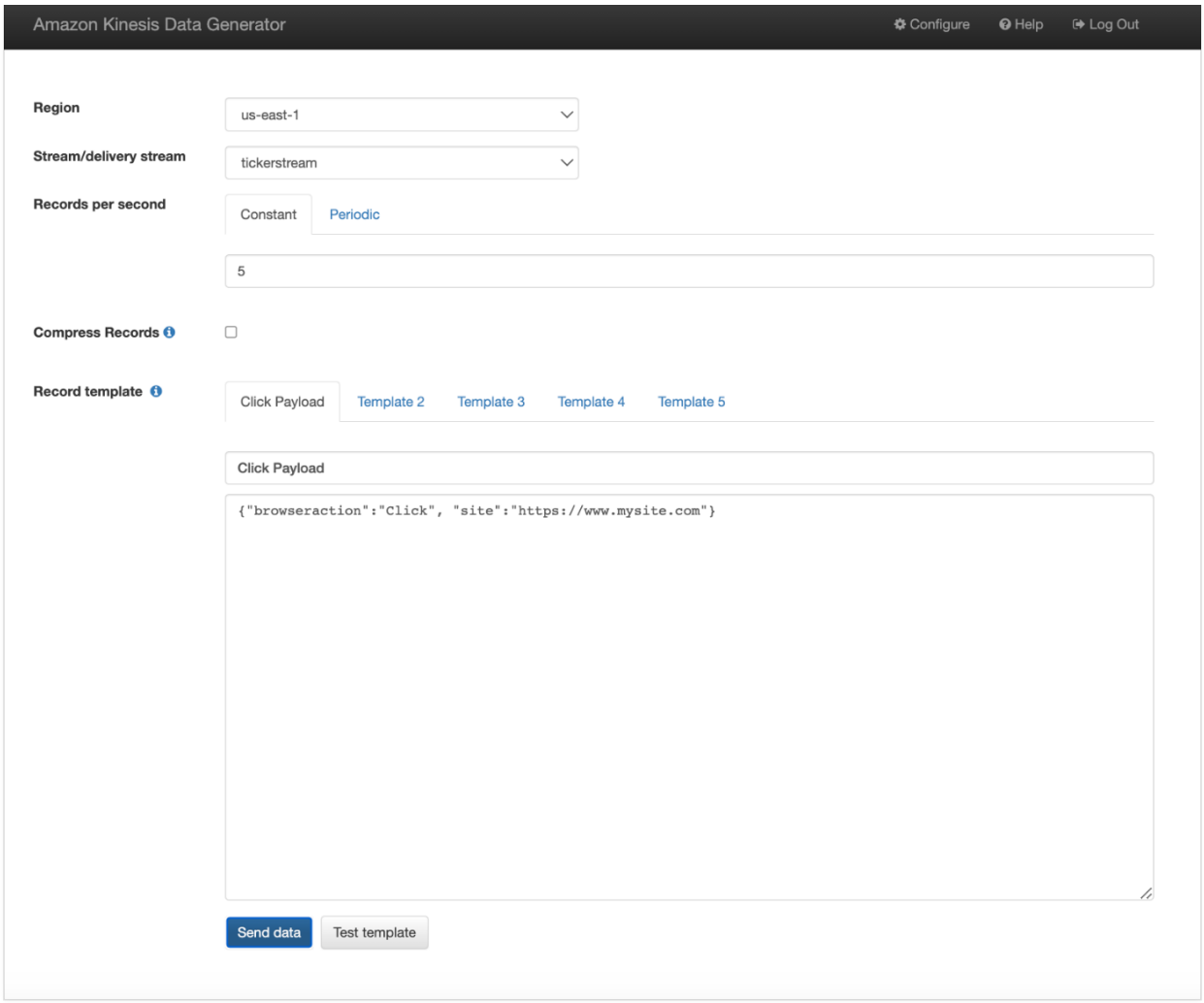
3. Open two concurrent sessions of the KDG UI in your browser. Sign in using the username and password you entered in the CloudFormation template while creating the Streaming Data Analytics Prelab setup.
   a. We want to generate more Click messages than Impressions. So, in the first session, send impression messages at rate of one message per second for 30 seconds to the tickerstream, the message body is

```
{"browseraction":"Impression", "site": "https://www.mysite.com"}
```

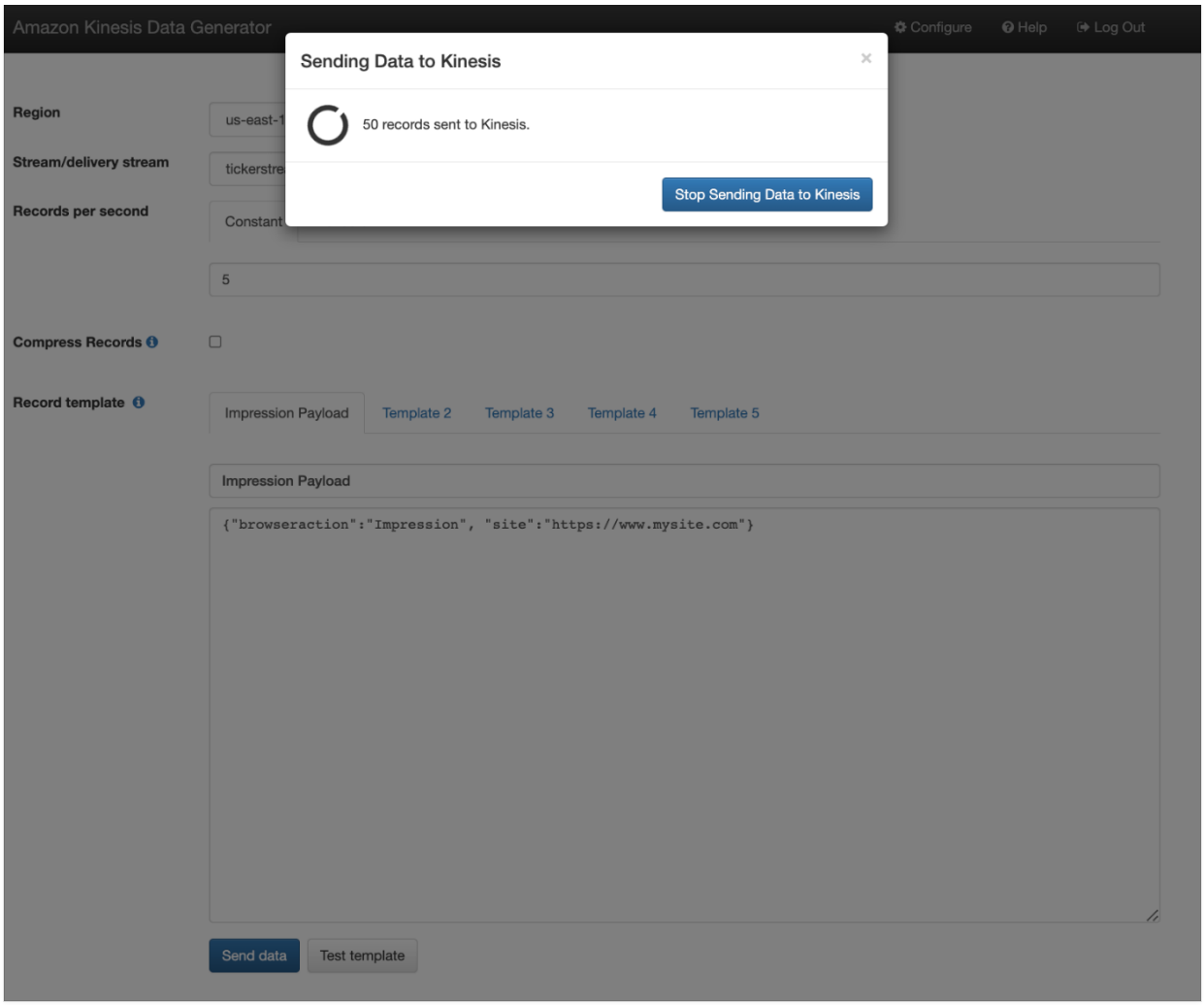


b. Then in the second session, send click messages at a rate of five messages per second for 30 seconds to the tickerstream, the message body is

```
{"browseraction":"Click", "site": "https://www.mysite.com"}
```



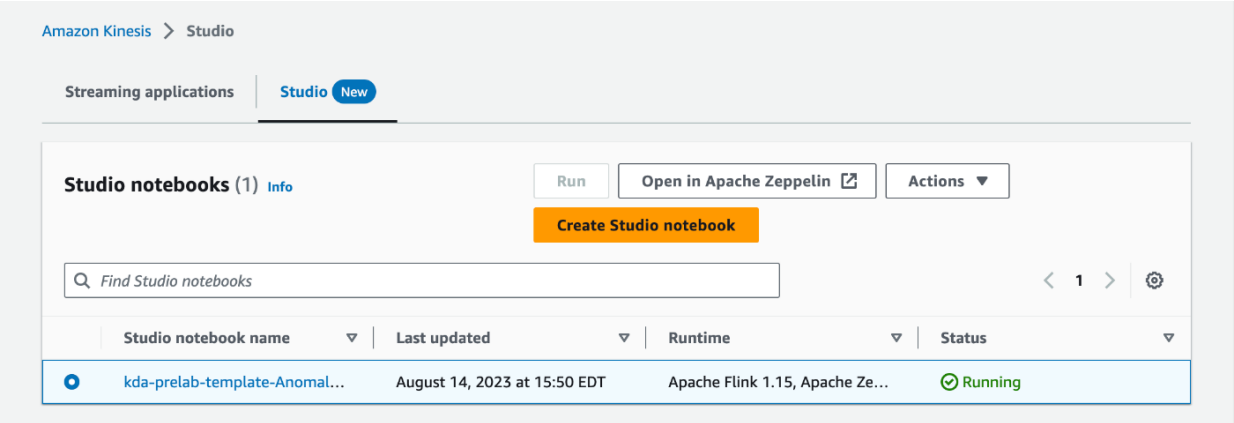
c. You can view the number of messages being sent to the data stream when you start posting the messages



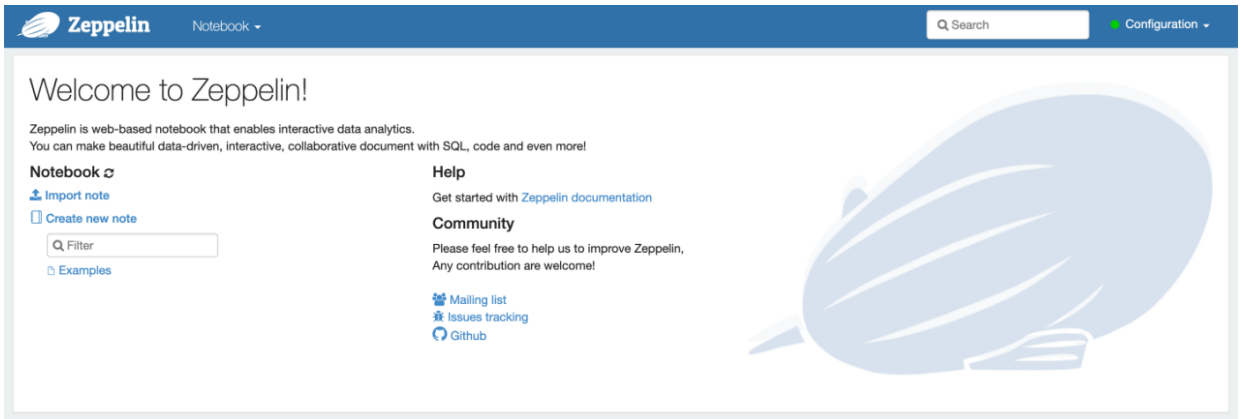
d. After 30 seconds, please stop sending both Click and Impression messages

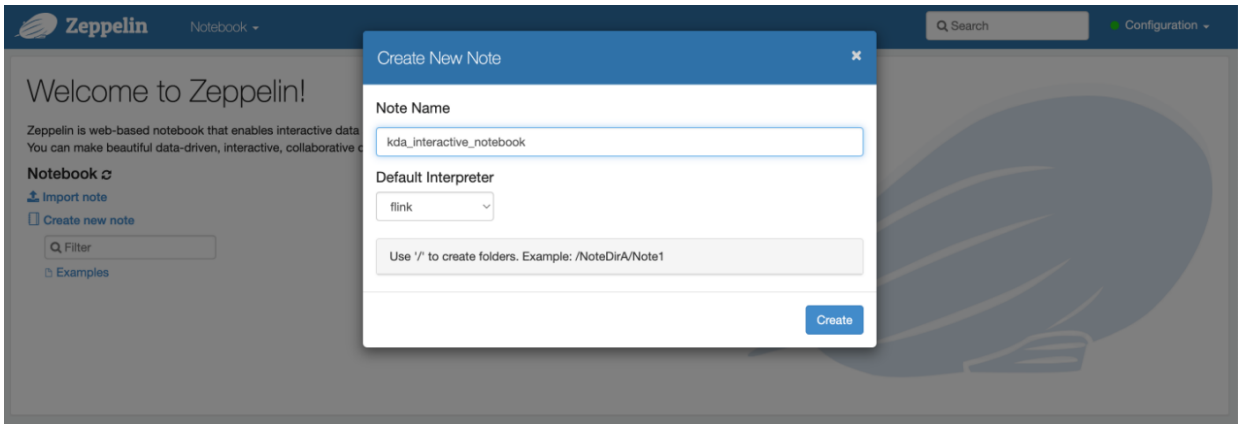
Perform real-time Data Analytics

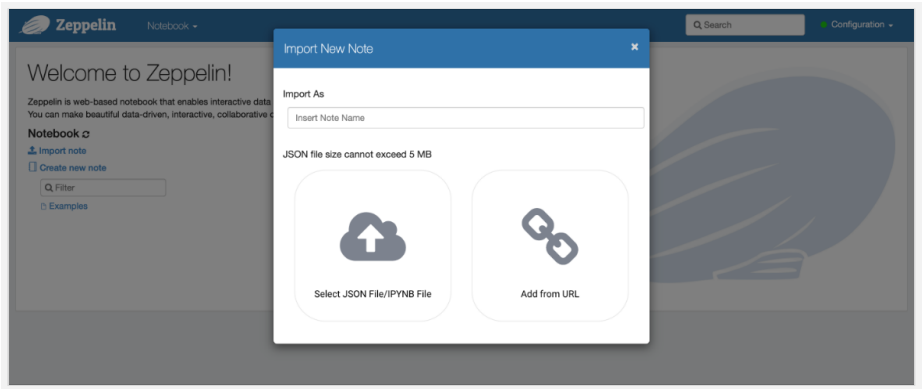
1. Navigate to the Amazon Kinesis Console → Analytics Application. Under the Studio tabs, select kda-prelab-template-RealtimeApplicationNotebook. Select “Open in Apache Zeppelin”.



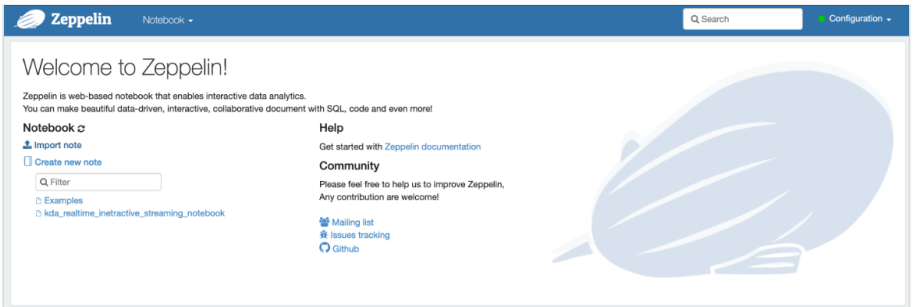
2. On the Apache Zeppelin Console, select Create new note. Provide the notebook name as **kda_Interactive_notebook**



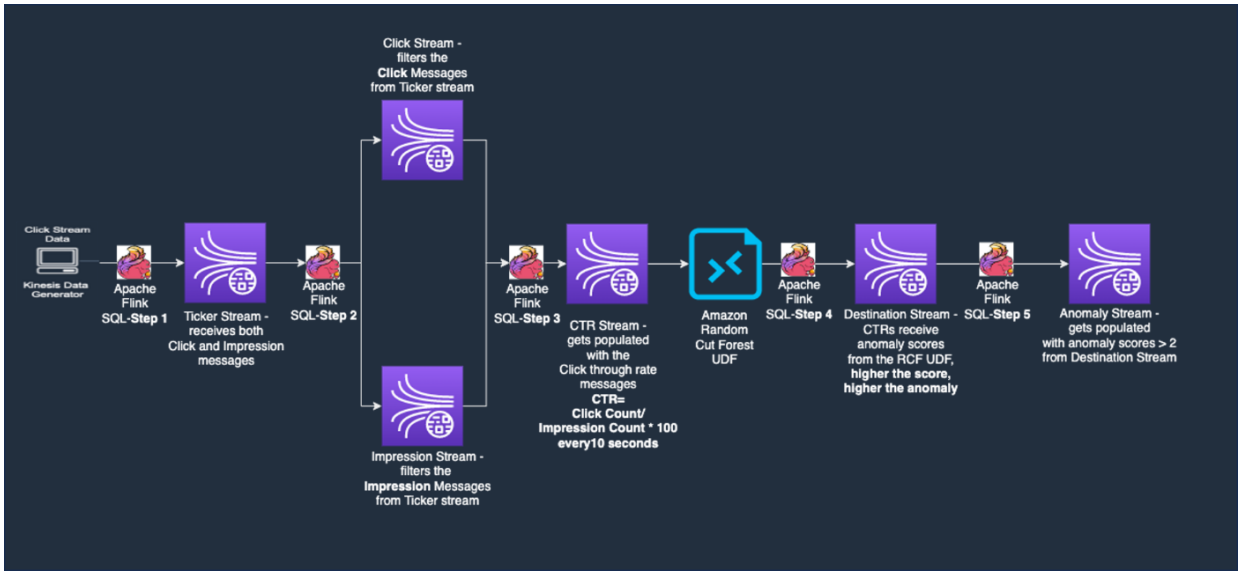


3. Now lets perform real time interactive analytics with Kinesis data streams. We will
    i. Create Flink tables using Flink SQL Queries
    ii. Use Flink SQL queries to transform and create new data streams in real-time
    iii. Perform anomaly detection using Flink User Defined Function and trigger anomaly notification emails in real-time.
    iv. The scripts are available here.
    v. A notebook is also available here which can be downloaded and imported through Apache Zeppelin console.
    vi. 

vii. You can then open the notebook and run the paragraphs one after the other.

viii. 

Please refer to the anomaly detection logic in the image below before you start running the remaining steps



a. Run the table creation scripts.

#1 Create all the Streaming tables

FINISHED

```
%flink.pyflink
st_env.execute_sql("""DROP TEMPORARY TABLE IF EXISTS tmptickerstream""")
st_env.execute_sql("""DROP TEMPORARY TABLE IF EXISTS tmpclickstream""")
st_env.execute_sql("""DROP TEMPORARY TABLE IF EXISTS tmpimpressionstream""")
st_env.execute_sql("""DROP TABLE IF EXISTS clickstream""")
st_env.execute_sql("""DROP TABLE IF EXISTS impressionstream""")
st_env.execute_sql("""DROP TEMPORARY TABLE IF EXISTS tmpctrstream""")
st_env.execute_sql("""DROP TABLE IF EXISTS ctrstream""")
st_env.execute_sql("""DROP TEMPORARY TABLE IF EXISTS tmpdestinationstream""")
st_env.execute_sql("""DROP TABLE IF EXISTS destinationstream""")
st_env.execute_sql("""DROP TEMPORARY TABLE IF EXISTS tmpanomalydetectionstream""")
st_env.execute_sql("""DROP TABLE IF EXISTS anomalydetectionstream""")
st_env.execute_sql("""CREATE TEMPORARY TABLE tmptickerstream (
  `browseraction` STRING,
  `site`          STRING,
  `rowtime`       TIMESTAMP(3) METADATA FROM 'timestamp',
  WATERMARK FOR rowtime AS rowtime - INTERVAL '0' SECOND
)
 WITH (
  'connector' = 'kinesis',
  'stream' = 'ticker',
  'aws.region' = 'us-east-1',
  'scan.stream.initpos' = 'TRIM_HORIZON',
  'format' = 'json'
)""")
```

b. User Defined Function (UDF) performs Anomaly Detection in real-time using Random Cut Forest algorithm. Run step #2.

#2: Create the UDF (User Defined Function)

FINISHED

```
%flink(parallelism=1)
import software.amazon.flink.example.RandomCutForestUDF
stenv.registerFunction("RANDOM_CUT_FOREST", new RandomCutForestUDF())
```

```
import software.amazon.flink.example.RandomCutForestUDF
```

Took 2 sec. Last updated by anonymous at August 23 2023, 3:27:35 AM. (outdated)

c. You can view real time data from website visits by running the query in Step #3.

#3: View the Ticker Stream Data in real-time

FLINK JOB ABORT

```
%flink.ssql(type=update)
select * from tmptickerstream;
```

settings

browseraction	site	rowtime
Click	https://www.mysite.com	2023-08-23 07:30:11.812
Click	https://www.mysite.com	2023-08-23 07:30:13.007
Click	https://www.mysite.com	2023-08-23 07:30:13.009
Click	https://www.mysite.com	2023-08-23 07:30:13.009
Click	https://www.mysite.com	2023-08-23 07:30:13.009
Click	https://www.mysite.com	2023-08-23 07:30:13.009
Click	https://www.mysite.com	2023-08-23 07:30:13.982
Click	https://www.mysite.com	2023-08-23 07:30:13.984
Click	https://www.mysite.com	2023-08-23 07:30:13.984

Note: Use %flink.ssql while querying the table

d. Create impressionstream by filtering messages from tickerstream.

#4: Generate the Impression Stream Data in real-time based on the browser action value in the Ticker stream

FLINK JOB ERROR

```
%flink.ssql(type=update, parallelsim=1)
INSERT INTO tmpimpressionstream
SELECT
  TUMBLE_START(rowtime, INTERVAL '10' SECOND) AS rowtime,
  COUNT(*) AS impressioncount
FROM tmptickerstream
WHERE browseraction='Impression'
GROUP BY
  TUMBLE(rowtime, INTERVAL '10' SECOND);
```

e. Create clickstream by filtering messages from tickerstream.

#5: Generate the Click Stream Data in real-time based on the browser action value in the Ticker stream

FLINK JOB ERROR

```
%flink.ssql(type=update, parallelsim=1)
INSERT INTO tmpclickstream
SELECT
  TUMBLE_START(rowtime, INTERVAL '10' SECOND) AS rowtime,
  COUNT(*) AS clickcount
FROM tmptickerstream
WHERE browseraction='Click'
GROUP BY
  TUMBLE(rowtime, INTERVAL '10' SECOND);
```

f. Calculate Click Through Rate (CTR) and populate ctrstream.

#10: View and Analyze the Anomalies in real-time

FLINK JOB ABORT

```
%flink.ssql
select * from anomalydetectionstream;
```

settings

rowtime	ctrpercent	anomaly_score
2023-08-23 13:46:30.000000	875.0	6.530422766870707
2023-08-23 13:46:40.000000	500.0	6.723916774777987
2023-08-23 13:46:50.000000	500.0	6.530422766870706
2023-08-23 13:47:00.000000	500.0	6.675543272801167
2023-08-23 13:47:10.000000	500.0	6.869037280708447
2023-08-23 13:47:20.000000	500.0	6.869037280708447
2023-08-23 13:47:30.000000	166.7	6.3853022609402466
2023-08-23 13:47:50.000000	500.0	6.482049264893886

g. You can view the Click Through Rate in real time by executing Step #7.

#7: Analyze the Click Through Rate in real-time

FLINK JOB ERROR

```
%flink.ssql(type=update)
select * from ctrstream;
```

settings

time	ctr
1692798390	875.0
1692798400	500.0
1692798410	500.0
1692798420	500.0
1692798430	500.0
1692798440	500.0
1692798450	166.7
1692798470	500.0

h. Use the UDF (Random Cut Forest) to generate anomaly scores.

2

#8: Generate the Anomaly Score in real-time with User Defined Function RANDOM_CUT_FOREST

FLINK JOB ABORT

```
%flink.ssql(type=update, parallelsim=1)
INSERT INTO tmpdestinationstream
SELECT `time`, ctr, RANDOM_CUT_FOREST(cast(ctr as float)) as score
FROM ctrstream;
```

i. Populate anomalydetectionstream by executing Step #9.

#9: Create a Stream with the Anomaly data in real-time

FLINK JOB ABORT

```
%flink.ssql(type=update, parallelsim=1)
INSERT INTO tmpanomalydetectionstream
SELECT to_timestamp(from_unixtime(`time`)) as rowtime, ctr as ctrpercent, score as anomaly_score
FROM destinationstream
WHERE score>2;
```

Now check the anomaly scores from Random Cut Forest algorithm in real time.

a. 2

#10: View and Analyze the Anomalies in real-time

FLINK JOB ABORT

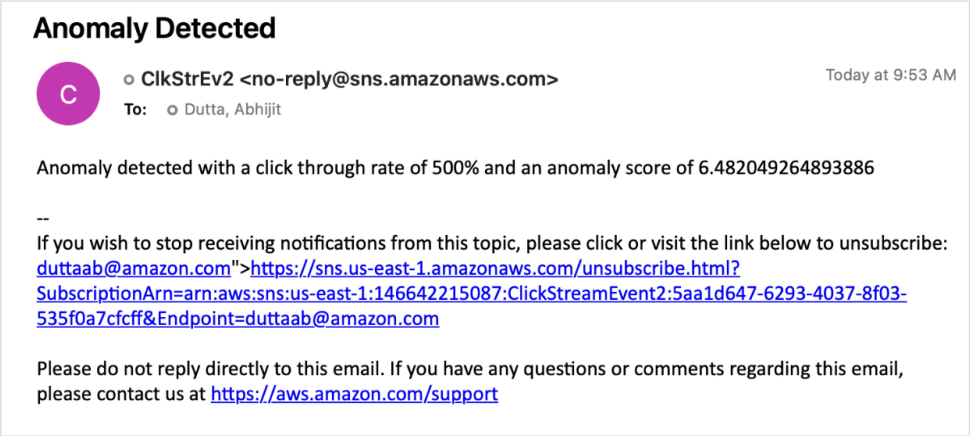
```
%flink.ssql
select * from anomalydetectionstream;
```

settings

rowtime	ctrpercent	anomaly_score
2023-08-23 13:46:30.000000	875.0	6.530422766870707
2023-08-23 13:46:40.000000	500.0	6.723916774777987
2023-08-23 13:46:50.000000	500.0	6.530422766870706
2023-08-23 13:47:00.000000	500.0	6.675543272801167
2023-08-23 13:47:10.000000	500.0	6.869037280708447
2023-08-23 13:47:20.000000	500.0	6.869037280708447
2023-08-23 13:47:30.000000	166.7	6.3853022609402466
2023-08-23 13:47:50.000000	500.0	6.482049264893886

b. You will start receiving notifications in your email when anomalies are detected:

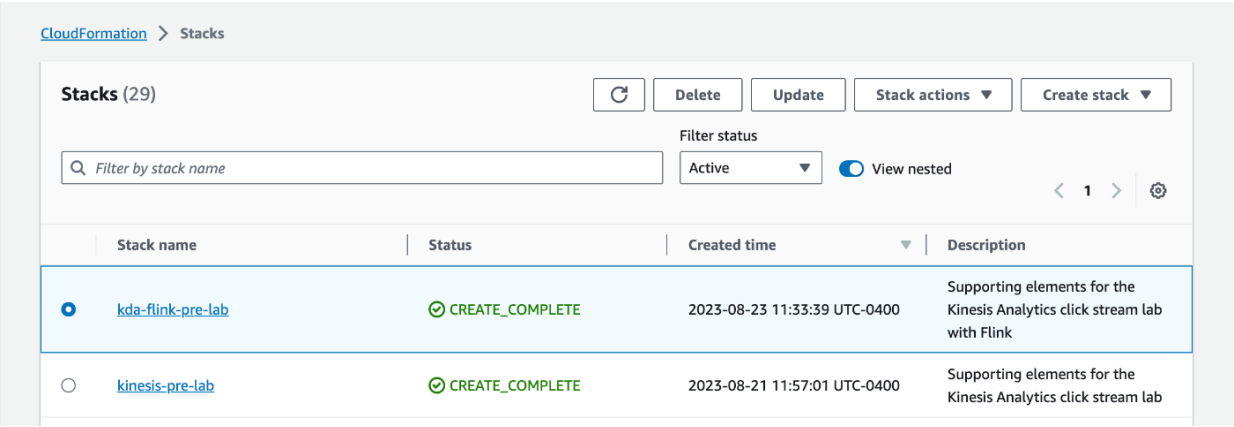
c.



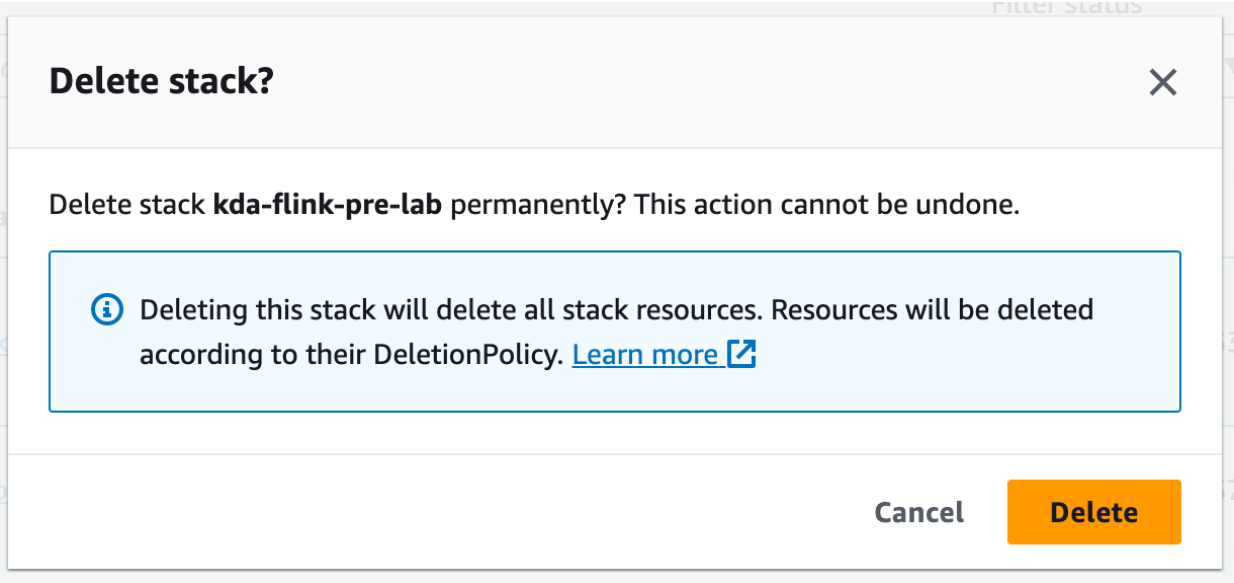
d. If you do not receive the anomaly notification emails on your first attempt
   i. Open the two concurrent sessions of KDG UI in your browser again.
   ii. In the first session, send impression messages at rate of one message per second to the tickerstream, the message body is {"browseraction":"Impression", "site":"https://www.mysite.com"}
   iii. In the second session, send click messages at a rate of five messages per second to the tickerstream, the message body is {"browseraction":"Click", "site":"https://www.mysite.com"}
   iv. Stop sending messages after 30-40 seconds.
   v. Now on the Apache Zeppelin Notebook, repeat steps 3 to 10 and you should start receiving email notifications from the second attempt.

Environment Clean Up

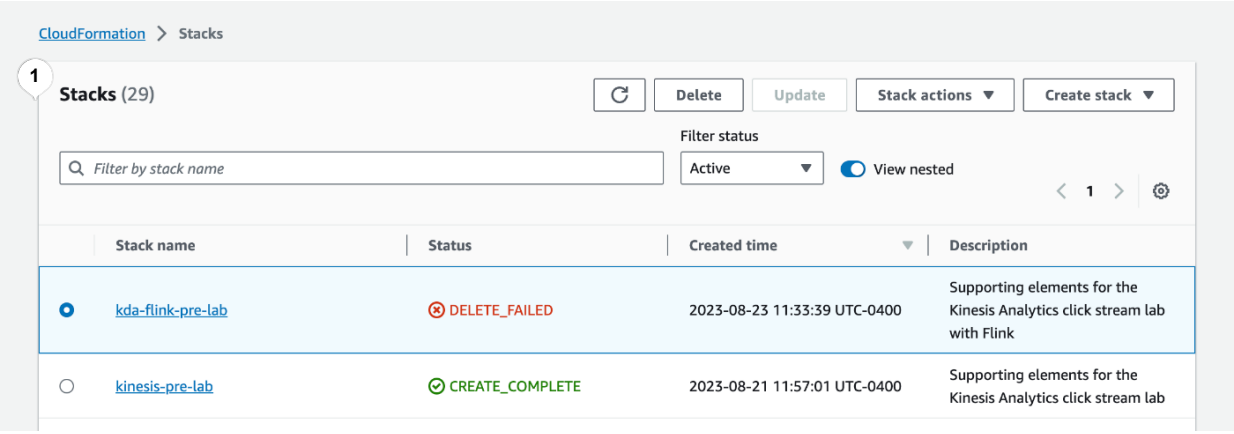
1. After completing the lab, click **Actions** → **Stop Application** to stop your application and avoid flood of SMS and e-mails messages.
2. If you would like to delete the entire resource stack, navigate to Amazon CloudFormation Console → Stacks, select **kda-flink-pre-lab** and click Delete to remove the stack. This step will clean up all the resources created by the stack earlier.



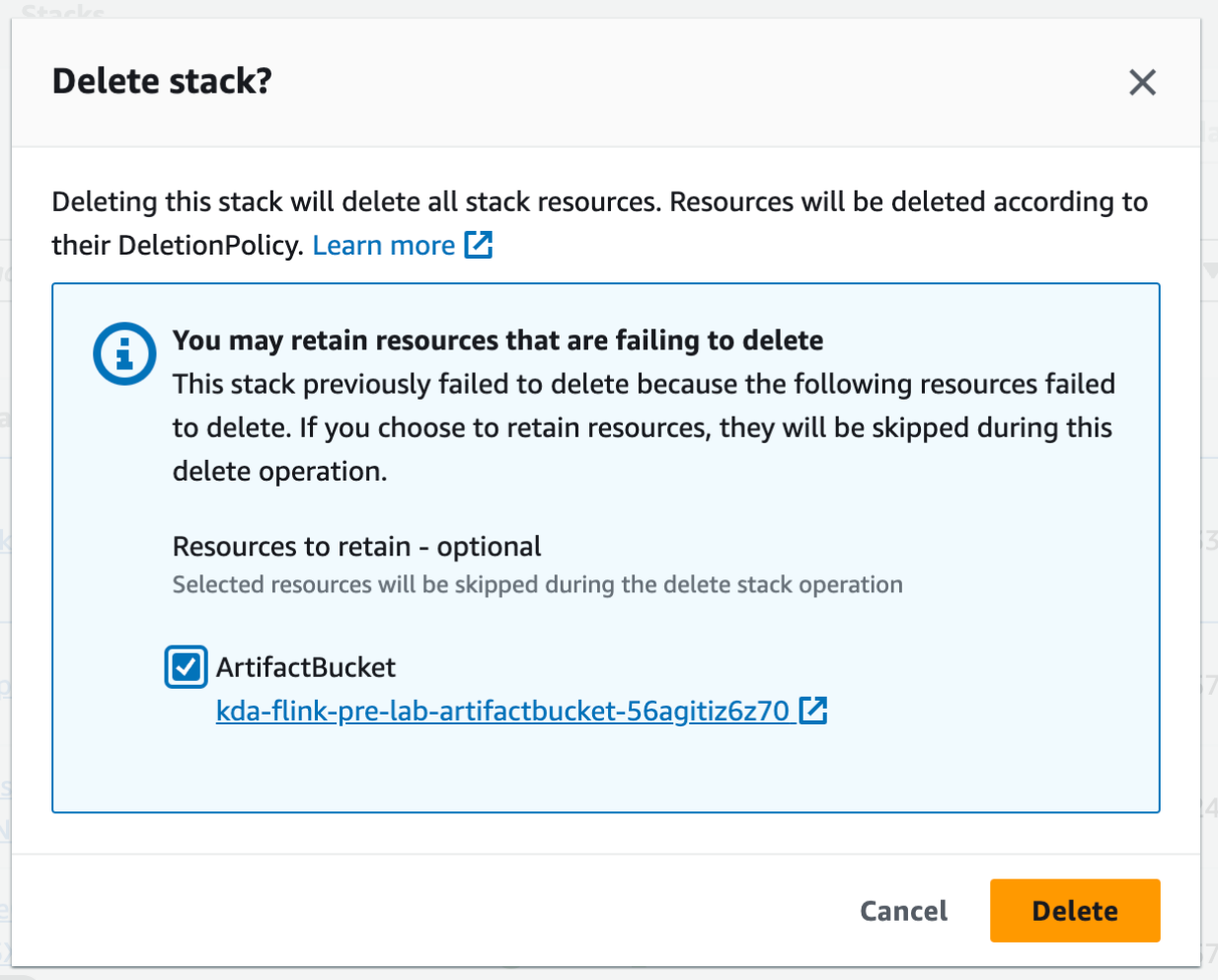
2. Select Delete on the next screen



3. If Delete Stack operation fails due to non-empty S3 buckets, select Delete again. On the pop up, either retain the S3 buckets failing to delete or manually empty the buckets.



4. This should pop up the list of the non-empty S3 buckets, either retain the S3 buckets failing to delete by hitting the checkbox or manually empty and remove those buckets. Hit “Delete” again once you have performed either of the previously mentioned steps, this time it should remove the stack completely.



Appendix: Anomaly Detection Scripts

1. Amazon CloudFormation Template: https://kda-flink-pre-lab-cfn-template.s3.amazonaws.com/kda_flink_lab.yaml
2. Apache Zeppelin Notebook: https://kda-flink-pre-lab-cfn-template.s3.amazonaws.com/kda_realtime_inetractive_streaming_notebook.zpln
3. Amazon Random Cut Forest User Defined Function: https://raw.githubusercontent.com/duttaabhijit06/amazon-Kinesis-data-analytics-examples/master/AnomalyDetection/RandomCutForest/src/main/java/software/amazon/flink/example/RandomCutForestUDF.java