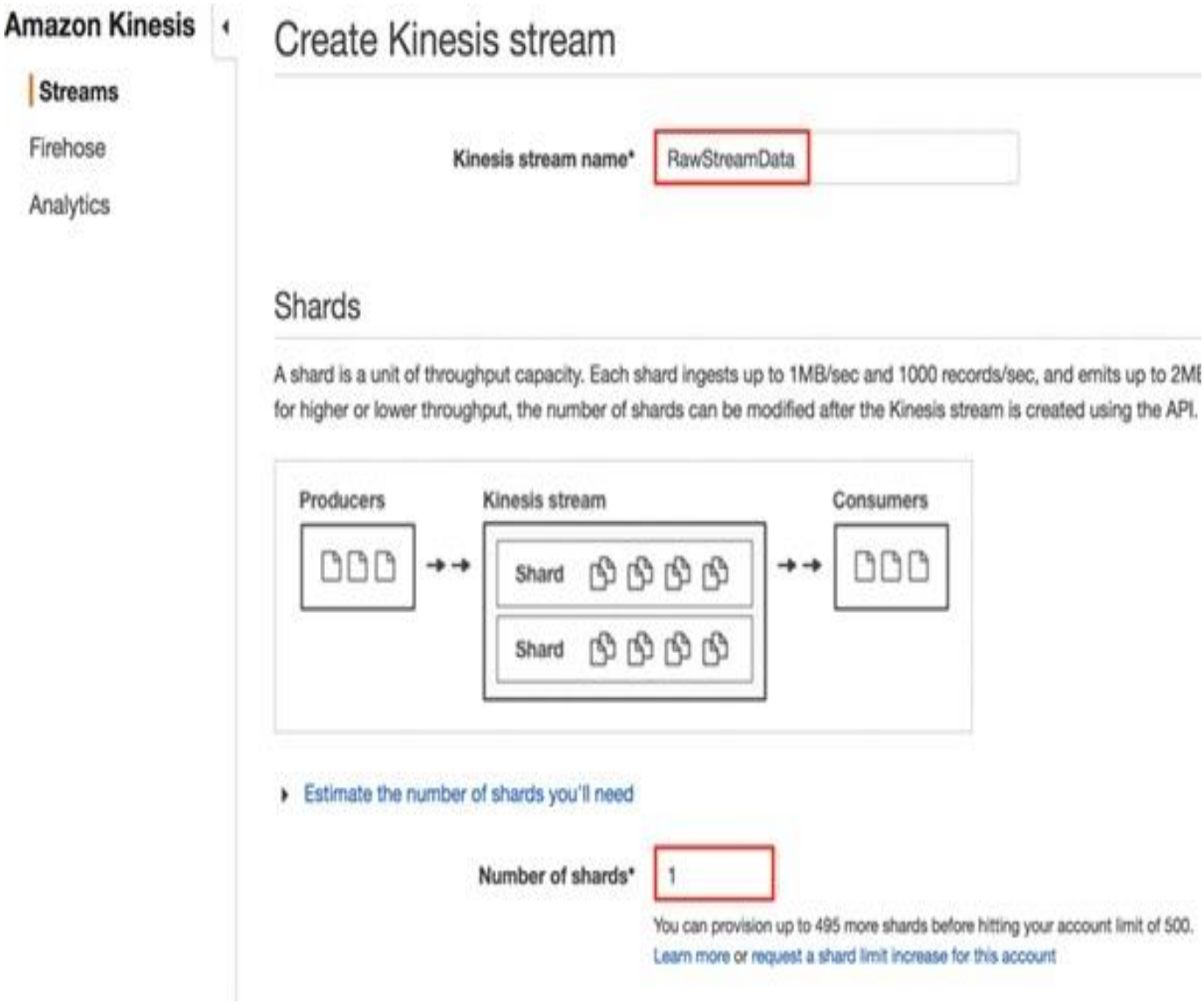
BY JOHN B ATHAPPULLY

1. Open the Amazon Kinesis Data Streams console.
2. Create a new Kinesis stream. Give it a name that indicates it's for raw incoming stream data—for example, RawStreamData. For Number of shards, type 1.



The Python code provided below simulates a streaming application, such as an IoT device, and generates random data and anomalies into a Kinesis stream. The code generates two temperature ranges, where the first range is the hypothetical sensor's normal operating temperature range (10-20), and the second is the anomaly temperature range (100-120).Make sure to change the stream name on line 16 and 20

and the Region on line 6 to match your configuration. Alternatively, you can download the Amazon Kinesis Data Generator from this repository and use it to generate the data.

```
      Import json
import datetime
import random
import testdata
from boto import kinesis
kinesis = kinesis.connect_to_region("us-east-1")
def getData(iotName, lowVal, highVal):
  data = {}
  data["iotName"] = iotName
  data["iotValue"] = random.randint(lowVal, highVal)
  return data
while 1:
  rnd = random.random()
  if (rnd < 0.01):
    data = json.dumps(getData("DemoSensor", 100, 120))
    kinesis.put_record("RawStreamData", data, "DemoSensor")
    print '**************************** anomaly ************************* ' + data
  else:
    data = json.dumps(getData("DemoSensor", 10, 20))
    kinesis.put_record("RawStreamData", data, "DemoSensor")
    print data
```

. Open the Amazon Elastic search Service console and create a new domain.

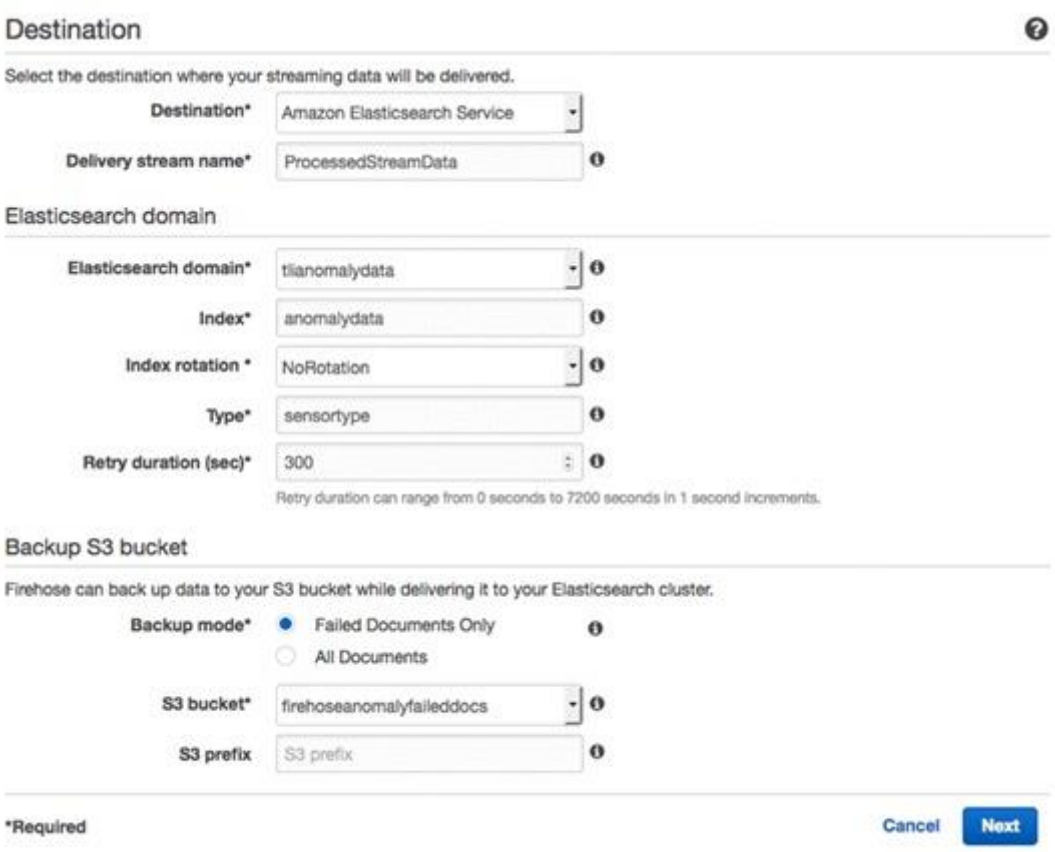
1. Give the domain a unique name. In the Configure cluster screen, use the default settings.
2. In the Set up access policy screen, in the Set the domain access policy list, choose Allow access to the domain from specific IP(s).
3. Enter the public IP address of your computer

After the Amazon Elasticsearch Service domain is up and running, you can set up and configure Kinesis Data Firehose to export results to Amazon Elasticsearch Service:

1. Open the Amazon Kinesis Data Firehose console and choose Create Delivery Stream.

2. In the Destination dropdown list, choose Amazon Elasticsearch Service.

1. Type a stream name, and choose the Amazon Elasticsearch Service domain that you created in Step 4.
2. Provide an index name and ES type. In the S3 bucket dropdown list, choose Create New S3 bucket. Choose

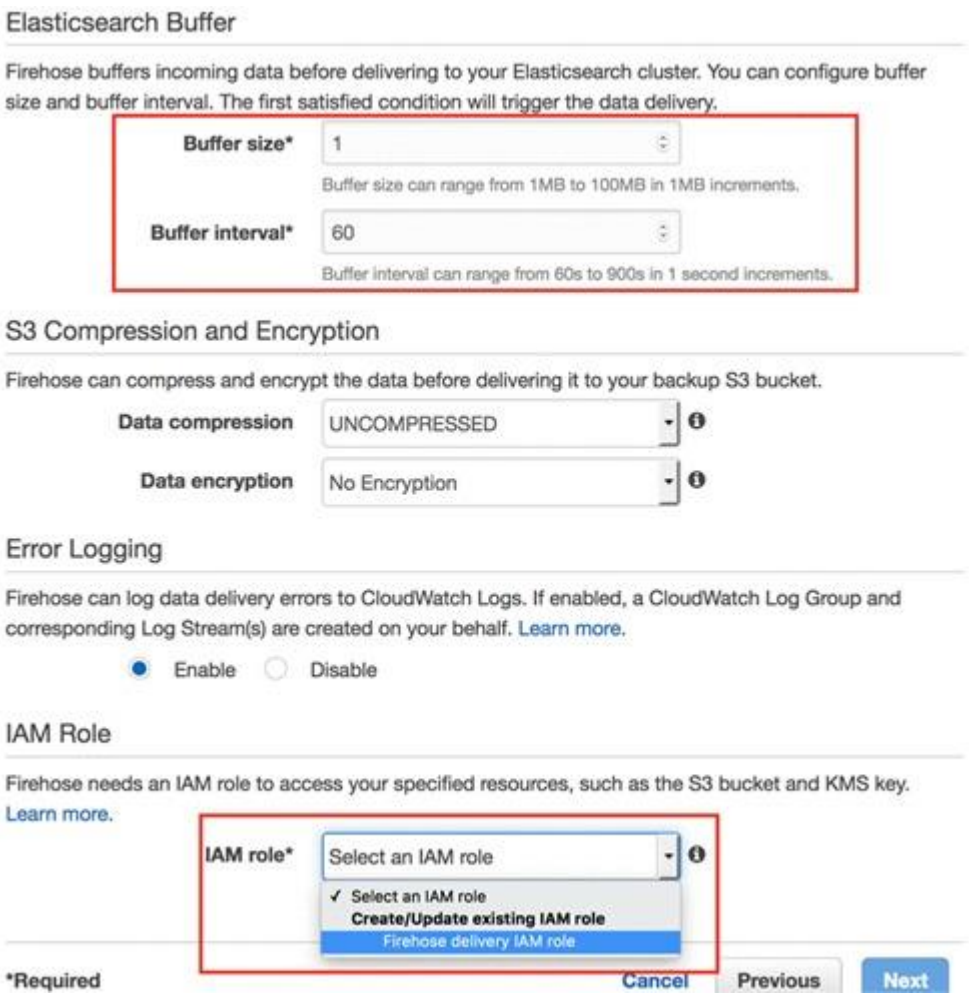


In the configuration, change the Elasticsearch Buffer size to 1 MB and the Buffer interval to 60s. Use the default settings for all other fields. This shortens the time for the data to reach the ES cluster.

1. Under IAM Role, choose Create/Update existing IAM role.
   The best practice is to create a new role every time. Otherwise, the console keeps adding policy documents to the same role. Eventually the size of the attached policies causes IAM to reject the role, but it does it in a non-obvious way, where the console basically quits functioning.
2. Choose Next to move to the Review page.

In the configuration, change the Elasticsearch Buffer size to 1 MB and the Buffer interval to 60s. Use the default settings for all other fields. This shortens the time for the data to reach the ES cluster.

1. Under IAM Role, choose Create/Update existing IAM role.
   The best practice is to create a new role every time. Otherwise, the console keeps adding policy documents to the same role. Eventually the size of the attached policies causes IAM to reject the role, but it does it in a non-obvious way, where the console basically quits functioning.
2. Choose Next to move to the Review page.



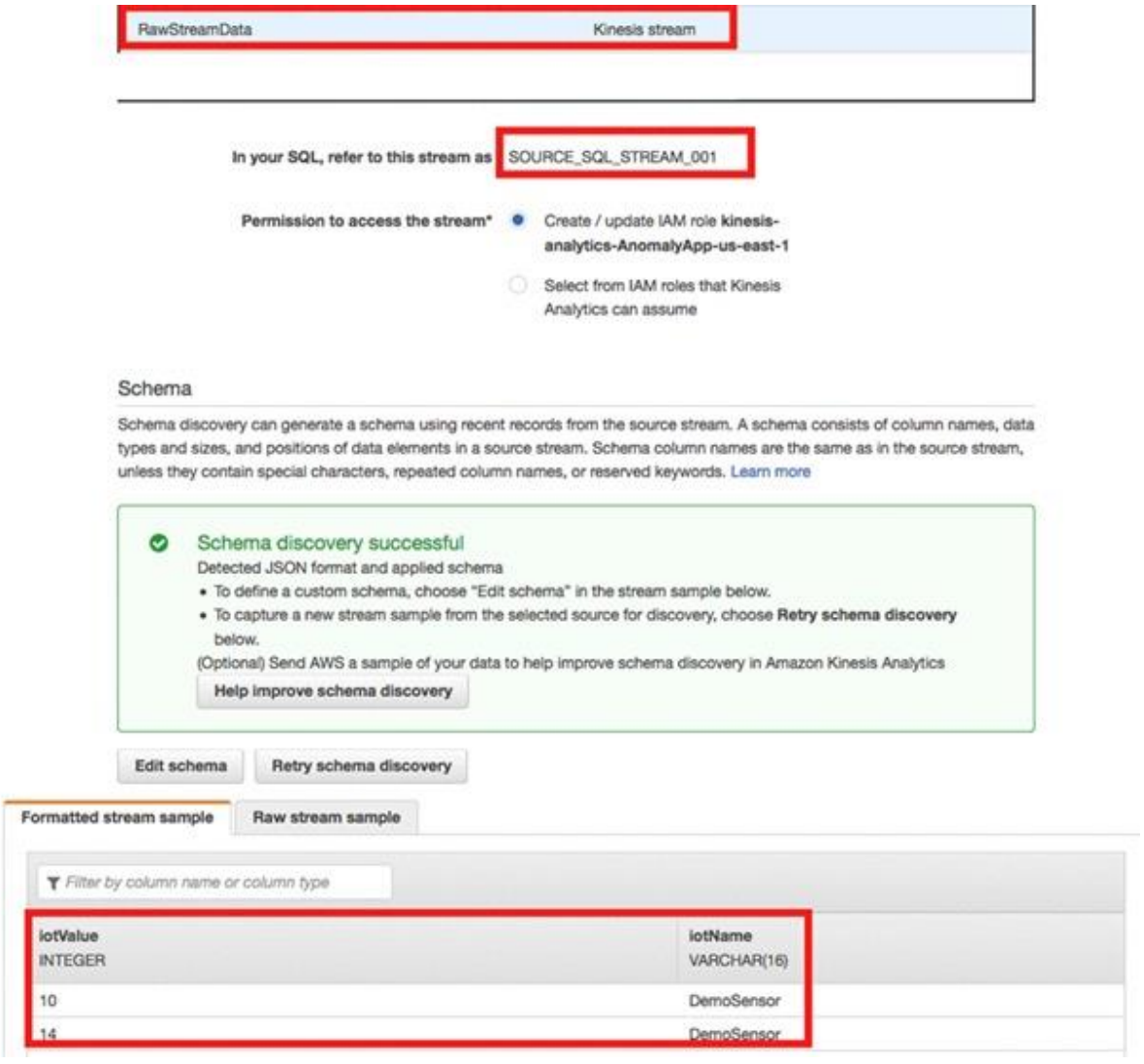
Review the configuration, and then choose Create Delivery Stream.

Run the Python file for 1-2 minutes, and then press Ctrl+C to stop the execution. This loads some data into the stream for you to visualize in the next step.

Analyzing the data

Now it's time to analyze the IoT streaming data using Amazon Kinesis Data Analytics.

1. Open the Amazon Kinesis Data Analytics console and create a new application. Give the application a name, and then choose Create Application.
2. On the next screen, choose Connect to a source. Choose the raw incoming data stream that you created earlier. (Note the stream name Source_SQL_STREAM_001 because you will need it later.)
3. Use the default settings for everything else. When the schema discovery process is complete, it displays a success message with the formatted stream sample in a table as shown in the following screenshot. Review the data, and then choose Save and continue.



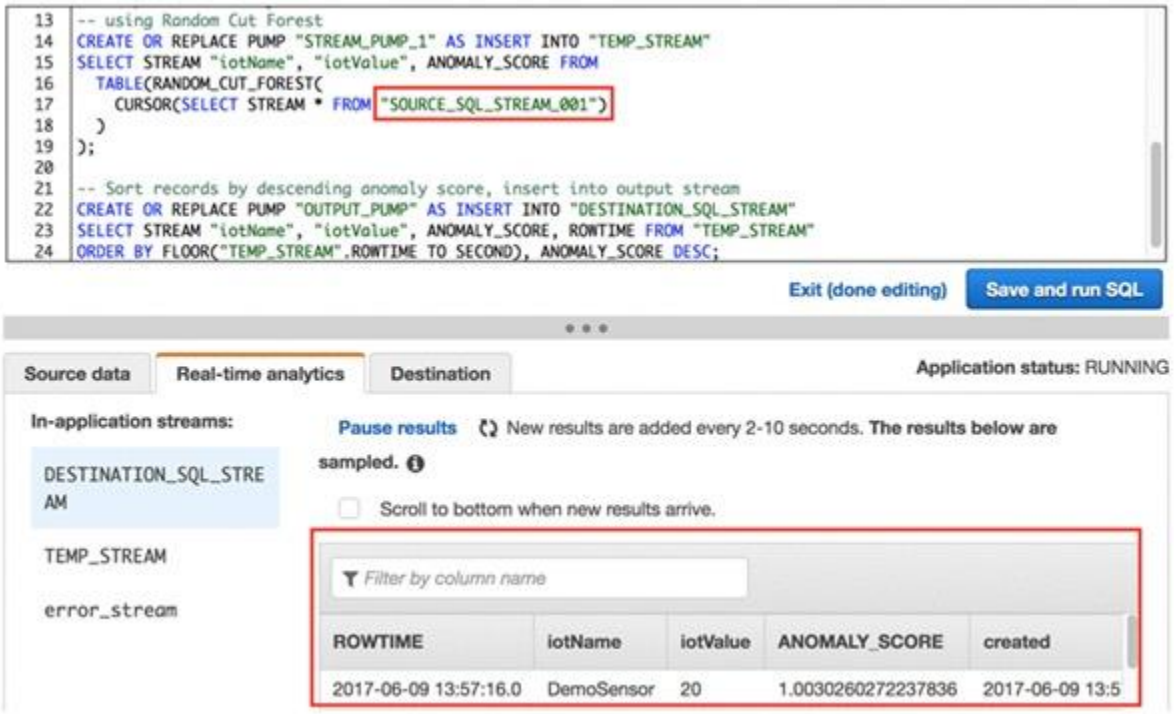
Next, choose Go to SQL editor. When prompted, choose Yes, start application.
Copy the following SQL code and paste it into the SQL editor window

```
CREATE OR REPLACE STREAM "TEMP_STREAM" (
  "iotName" varchar (40), "iotValue"   integer,
```

```
  "ANOMALY_SCORE" DOUBLE);
-- Creates an output stream and defines a schema
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
  "iotName"       varchar(40),
  "iotValue"      integer,
  "ANOMALY_SCORE" DOUBLE,
  "created" TimeStamp);
-- Compute an anomaly score for each record in the source stream
-- using Random Cut Forest
CREATE OR REPLACE PUMP "STREAM_PUMP_1" AS INSERT INTO
"TEMP_STREAM"
SELECT STREAM "iotName", "iotValue", ANOMALY_SCORE FROM
TABLE(RANDOM_CUT_FOREST(
   CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001")
)
);
-- Sort records by descending anomaly score, insert into output stream
CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS INSERT INTO
"DESTINATION_SQL_STREAM"
SELECT STREAM "iotName", "iotValue", ANOMALY_SCORE, ROWTIME FROM
"TEMP_STREAM"
ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE
DESC;
```

Choose **Save and run SQL**.

As the application is running, it displays the results as stream data arrives. If you don't see any data coming in, run the Python script again to generate some fresh data. When there is data, it appears in a grid as shown in the following screenshot.

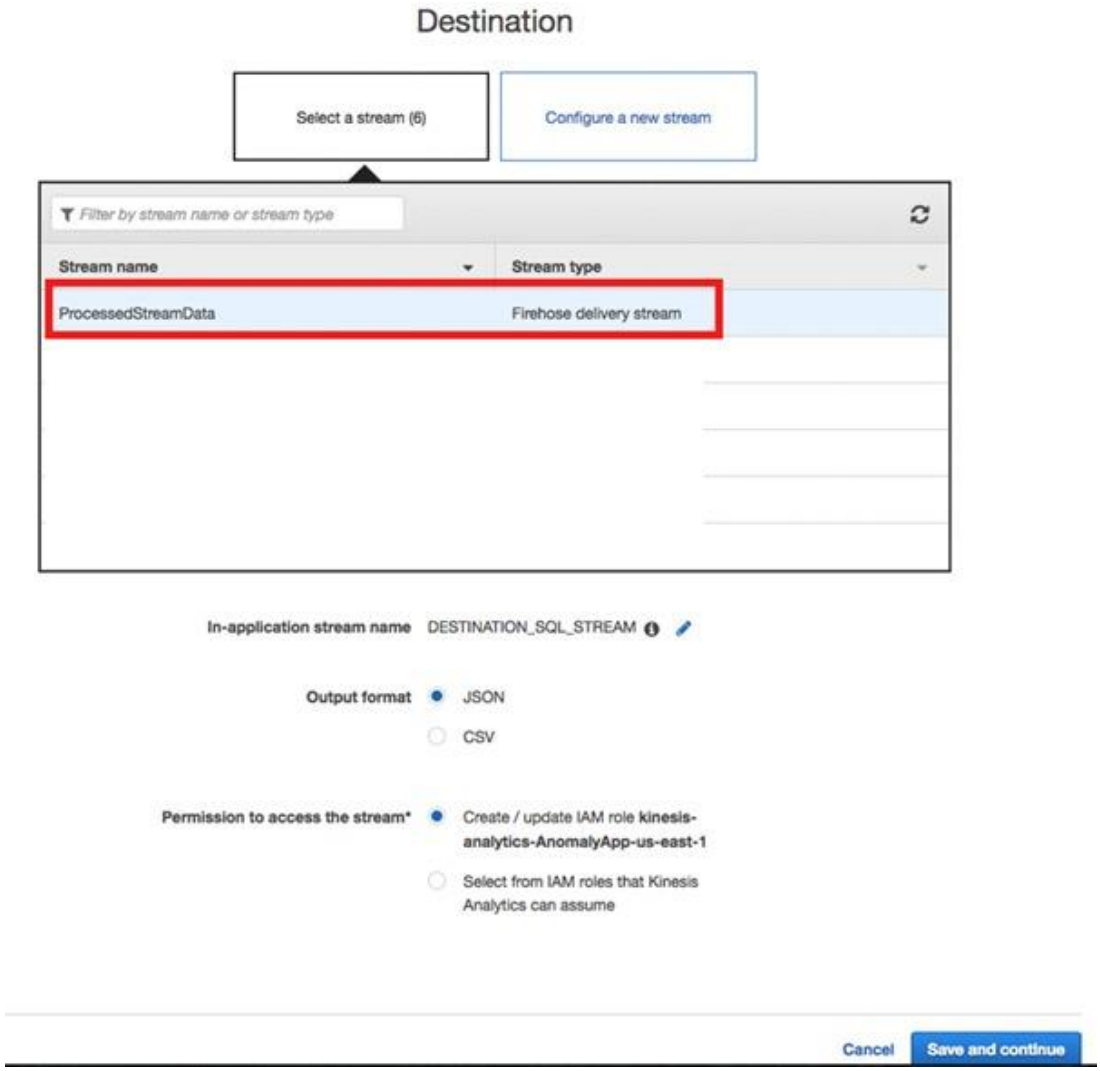


Note that you are selecting data from the source stream name Source_SQL_STREAM_001 that you created previously. Also note the ANOMALY_SCORE column.
This is the value that the Random_Cut_Forest function calculates based on the temperature ranges provided by the Python script. Higher (anomaly) temperature ranges have a higher score.
Looking at the SQL code, note that the first two blocks of code create two new streams to store temporary data and the final result. The third block of code analyzes the raw source data (Stream_Pump_1) using the Random_Cut_Forest function.
It calculates an anomaly score (ANOMALY_SCORE) and inserts it into the TEMP_STREAM stream. The final code block loads the result stored in the TEMP_STREAM into DESTINATION_SQL_STREAM.
Choose Exit (done editing) next to the Save and run SQL button to return to the application configuration page.

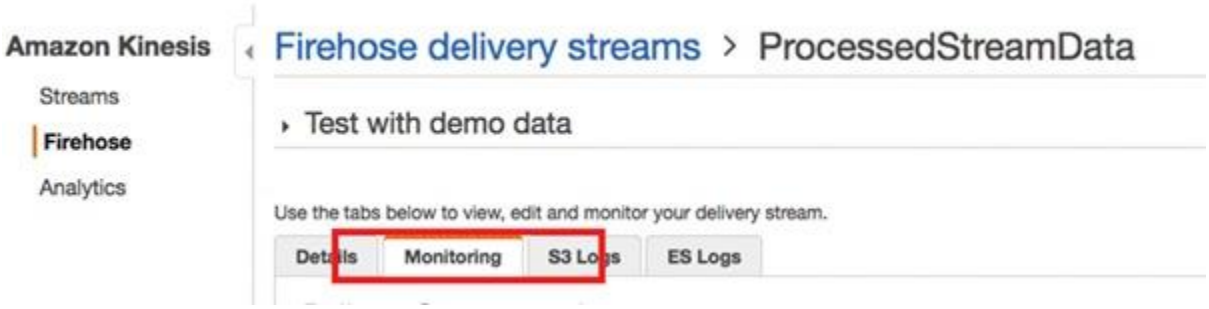
Load processed data into the Kinesis Data Firehose delivery stream

Now, you can export the result from DESTINATION_SQL_STREAM into the Amazon Kinesis Data Firehose stream that you created previously.

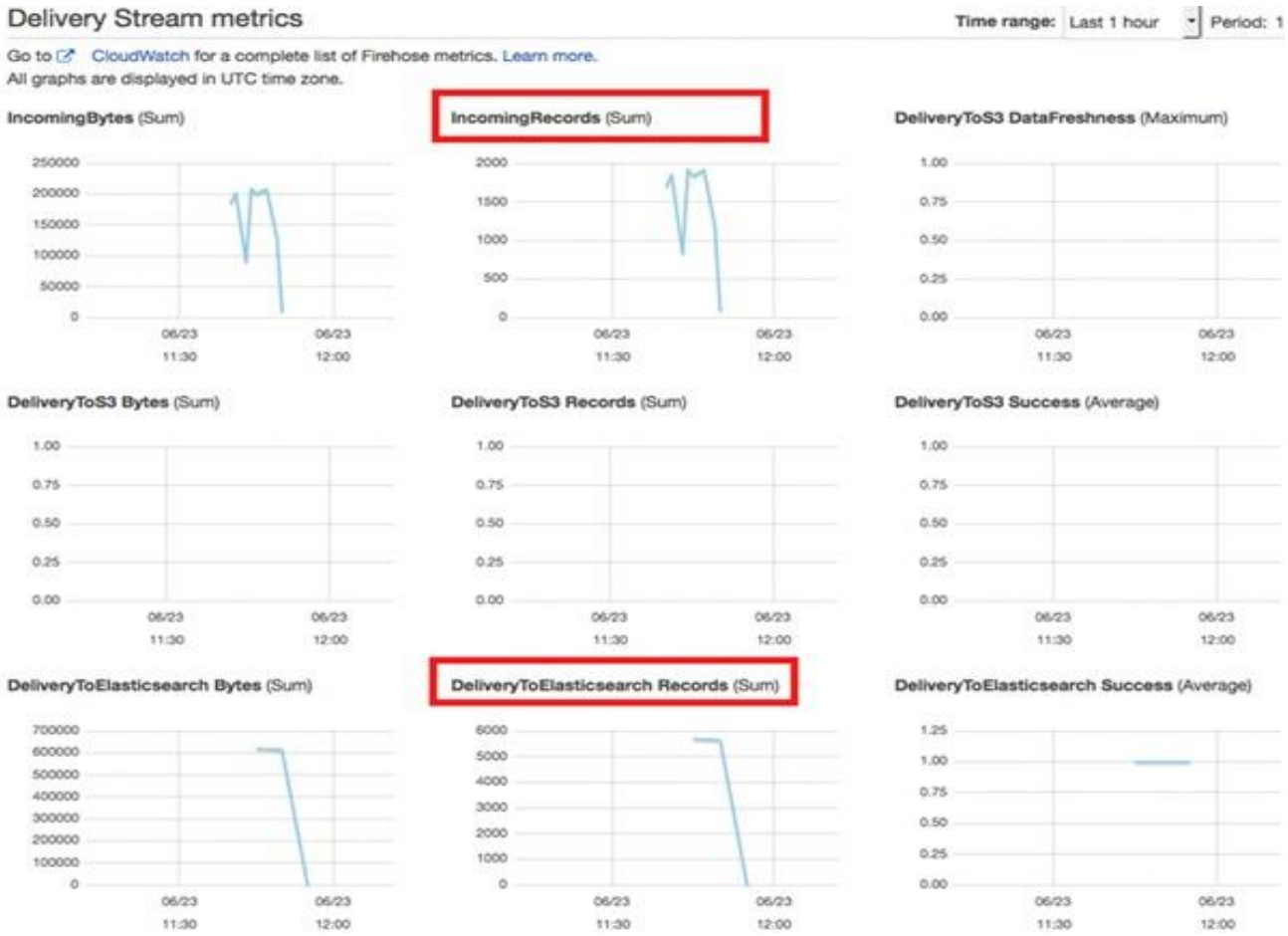
1. On the application configuration page, choose Connect to a destination.
2. Choose the stream name that you created earlier, and use the default settings for everything else. Then choose Save and Continue.



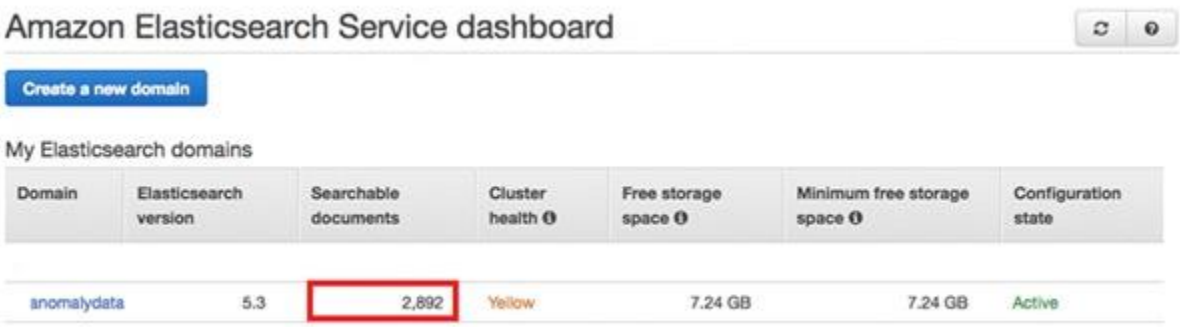
On the application configuration page, choose Exit to Kinesis Data Analytics applications to return to the Amazon Kinesis Data Analytics console.
Run the Python script again for 4-5 minutes to generate enough data to flow through Amazon Kinesis Data Streams, Kinesis Data Analytics, Kinesis Data Firehose, and finally into the Amazon Elasticsearch Service domain.
Open the Kinesis Data Firehose console, choose the stream, and then choose the Monitoring



As the processed data flows into Kinesis Data Firehose and Amazon Elasticsearch Service, the metrics appear on the Delivery Stream metrics page. Keep in mind that the metrics page takes a few minutes to refresh with the latest data.



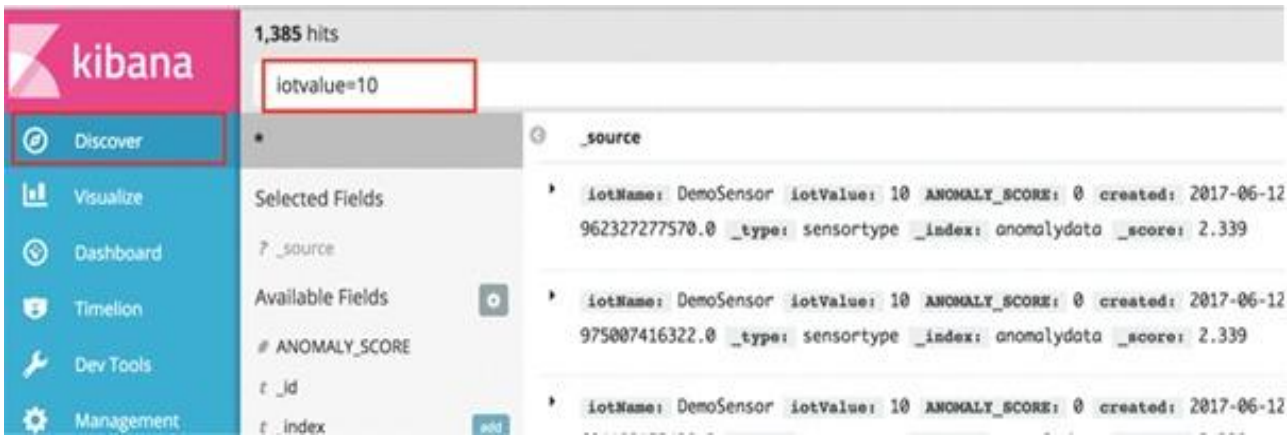
Open the Amazon Elasticsearch Service dashboard in the AWS Management Console. The count in the **Searchable documents** column increases as shown in the following screenshot. In addition, the domain shows a cluster health of **Yellow**. This is because, by default, it needs two instances to deploy redundant copies of the index. To fix this, you can deploy two instances instead of one.



Visualize the data using Kibana
Now it's time to launch Kibana and visualize the data.

1. Use the ES domain link to go to the cluster detail page, and then choose the Kibana link as shown in the following screenshot.

In the Kibana dashboard, choose the **Discover** tab to perform a query.



You can also visualize the data using the different types of charts offered by Kibana. For example, by going to the **Visualize** tab, you can quickly create a split bar chart that aggregates by ANOMALY_SCORE per minute.