

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.

Amazon Kinesis

- Streams
- Firehose
- Analytics

Create Kinesis stream

Kinesis stream name*

Shards

A shard is a unit of throughput capacity. Each shard ingests up to 1MB/sec and 1000 records/sec, and emits up to 2MB/sec. For higher or lower throughput, the number of shards can be modified after the Kinesis stream is created using the API.

The diagram illustrates the Kinesis stream architecture. On the left, a box labeled 'Producers' contains three document icons. An arrow points from the producers to a central box labeled 'Kinesis stream'. This box contains two 'Shard' units, each represented by a document icon and four smaller document icons. An arrow points from the 'Kinesis stream' box to a box on the right labeled 'Consumers', which contains three document icons.

► [Estimate the number of shards you'll need](#)

Number of shards*

You can provision up to 495 more shards before hitting your account limit of 500.
[Learn more or request a shard limit increase for this account](#)

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

Set up access policy

To allow or block access to the domain, select a policy template from the template selector or add one or more Identity and Access Management (IAM) policy statements in the **Edit the access policy** box.

Set the domain access policy to **Allow access to the domain from specific IP(s)**

Add or edit the access policy

IP address

Add a comma-separated list of valid IPv4 addresses or CIDR blocks. The IP addresses listed here can access the domain endpoint.

your public ip here

Cancel OK

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.

Create Delivery Stream

Step 1: Destination

Step 2: Configuration

Step 3: Review

Destination

Select the destination where your streaming data will be delivered.

Destination* Select destination

- Select destination
- Amazon S3
- Amazon Redshift
- Amazon Elasticsearch Service

Cancel

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

Destination

Select the destination where your streaming data will be delivered.

Destination* Amazon Elasticsearch Service

Delivery stream name* ProcessedStreamData

Elasticsearch domain

Elasticsearch domain* titanomalydata

Index* anomalydata

Index rotation* NoRotation

Type* sensortype

Retry duration (sec)* 300

Retry duration can range from 0 seconds to 7200 seconds in 1 second increments.

Backup S3 bucket

Firehose can back up data to your S3 bucket while delivering it to your Elasticsearch cluster.

Backup mode* ☒ Failed Documents Only ☐ All Documents

S3 bucket* firehoseanomalyfaileddocs

S3 prefix S3 prefix

*Required

Cancel Next

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.

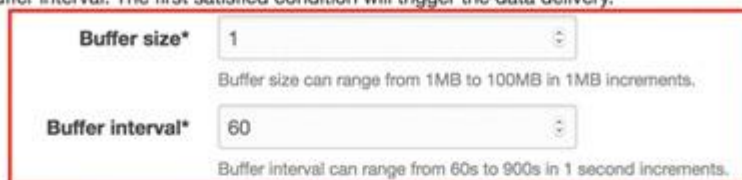
1. Under IAM Role, choose Create/Update existing IAM role.
The best practice is to create a new role every time. Otherwise, the console keeps adding policy documents to the same role. Eventually the size of the attached policies causes IAM to reject the role, but it does it in a non-obvious way, where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

Elasticsearch Buffer

Firehose buffers incoming data before delivering to your Elasticsearch cluster. You can configure buffer size and buffer interval. The first satisfied condition will trigger the data delivery.



S3 Compression and Encryption

Firehose can compress and encrypt the data before delivering it to your backup S3 bucket.



Error Logging

Firehose can log data delivery errors to CloudWatch Logs. If enabled, a CloudWatch Log Group and corresponding Log Stream(s) are created on your behalf. [Learn more.](#)

☒ Enable ☐ Disable

IAM Role

Firehose needs an IAM role to access your specified resources, such as the S3 bucket and KMS key. [Learn more.](#)



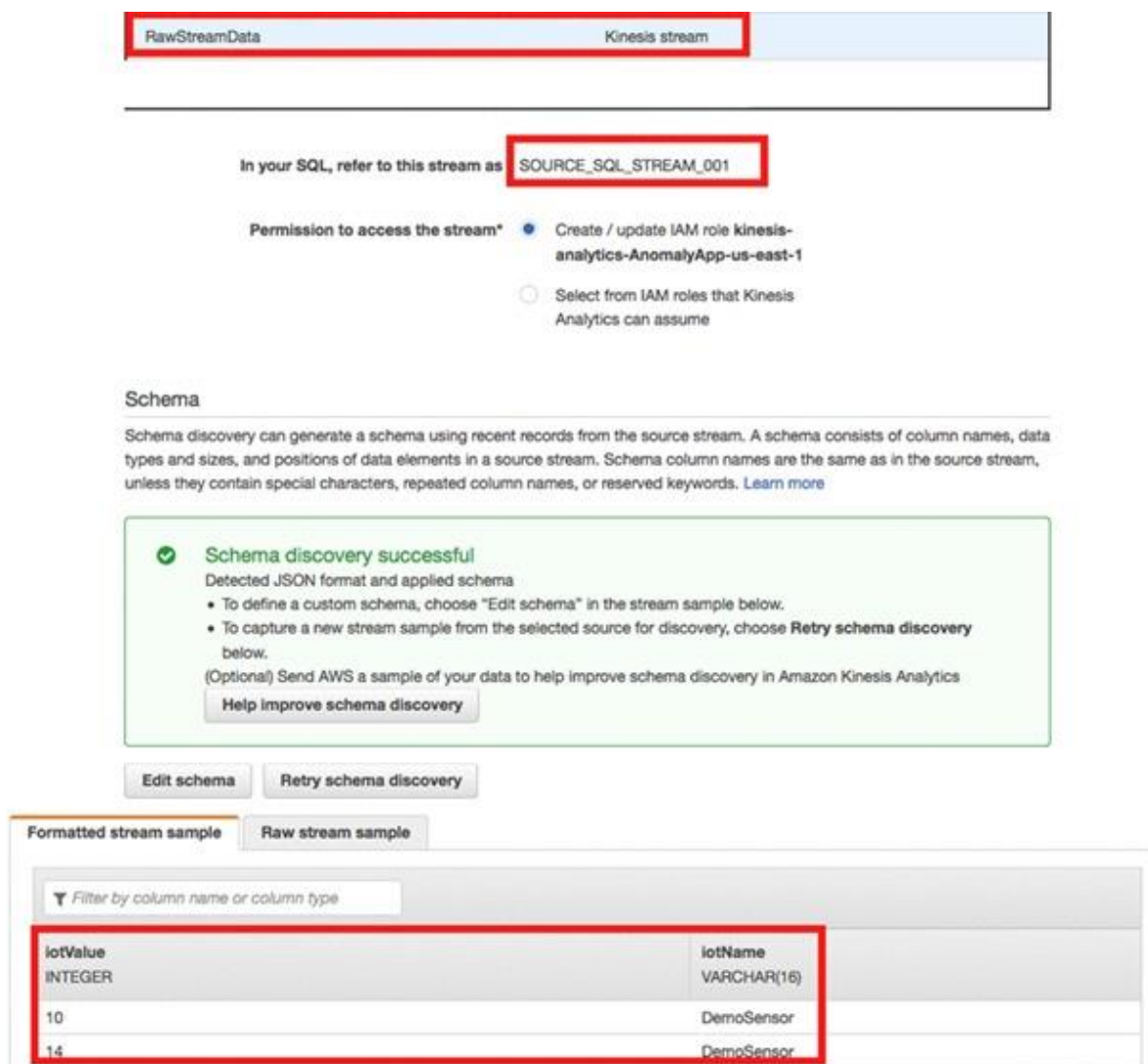
Review the configuration, and then choose Create Delivery Stream.

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.



RawStreamData Kinesis stream

In your SQL, refer to this stream as SOURCE_SQL_STREAM_001

Permission to access the stream* ☒ Create / update IAM role kinesis-analytics-AnomalyApp-us-east-1
☐ Select from IAM roles that Kinesis Analytics can assume

Schema

Schema discovery can generate a schema using recent records from the source stream. A schema consists of column names, data types and sizes, and positions of data elements in a source stream. Schema column names are the same as in the source stream, unless they contain special characters, repeated column names, or reserved keywords. [Learn more](#)

✓ **Schema discovery successful**
Detected JSON format and applied schema

- To define a custom schema, choose "Edit schema" in the stream sample below.
- To capture a new stream sample from the selected source for discovery, choose **Retry schema discovery** below.

(Optional) Send AWS a sample of your data to help improve schema discovery in Amazon Kinesis Analytics
[Help improve schema discovery](#)

[Edit schema](#) [Retry schema discovery](#)

Formatted stream sample **Raw stream sample**

Filter by column name or column type

iotValue	iotName
INTEGER	VARCHAR(16)
10	DemoSensor
14	DemoSensor

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.

The screenshot shows a web-based interface for a stream processing application. At the top, there is a SQL editor with the following code:

```

13 -- using Random Cut Forest
14 CREATE OR REPLACE PUMP "STREAM_PUMP_1" AS INSERT INTO "TEMP_STREAM"
15 SELECT STREAM "iotName", "iotValue", ANOMALY_SCORE FROM
16 TABLE(RANDOM_CUT_FOREST(
17   CURSOR(SELECT STREAM * FROM "SOURCE_SQL_STREAM_001")
18 )
19 );
20
21 -- Sort records by descending anomaly score, insert into output stream
22 CREATE OR REPLACE PUMP "OUTPUT_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
23 SELECT STREAM "iotName", "iotValue", ANOMALY_SCORE, ROWTIME FROM "TEMP_STREAM"
24 ORDER BY FLOOR("TEMP_STREAM".ROWTIME TO SECOND), ANOMALY_SCORE DESC;

```

Below the editor are two buttons: "Exit (done editing)" and "Save and run SQL".

The main interface has three tabs: "Source data", "Real-time analytics" (which is active), and "Destination". The "Application status" is shown as "RUNNING".

Under "In-application streams:", there is a list of streams: "DESTINATION_SQL_STREAM", "TEMP_STREAM", and "error_stream". The "DESTINATION_SQL_STREAM" stream is selected.

There is a "Pause results" button and a note: "New results are added every 2-10 seconds. The results below are sampled." There is also a checkbox labeled "Scroll to bottom when new results arrive." which is currently unchecked.

A data grid is displayed, showing a single row of data. The grid has a search bar labeled "Filter by column name". The columns are: ROWTIME, iotName, iotValue, ANOMALY_SCORE, and created. The data row shows: 2017-06-09 13:57:16.0, DemoSensor, 20, 1.0030260272237836, and 2017-06-09 13:57:16.0.

ROWTIME	iotName	iotValue	ANOMALY_SCORE	created
2017-06-09 13:57:16.0	DemoSensor	20	1.0030260272237836	2017-06-09 13:57:16.0

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.

Destination

Select a stream (6)

Configure a new stream

Filter by stream name or stream type↻

Stream name	Stream type
ProcessedStreamData	Firehose delivery stream

In-application stream name

DESTINATION_SQL_STREAM ⓘ ✎

Output format

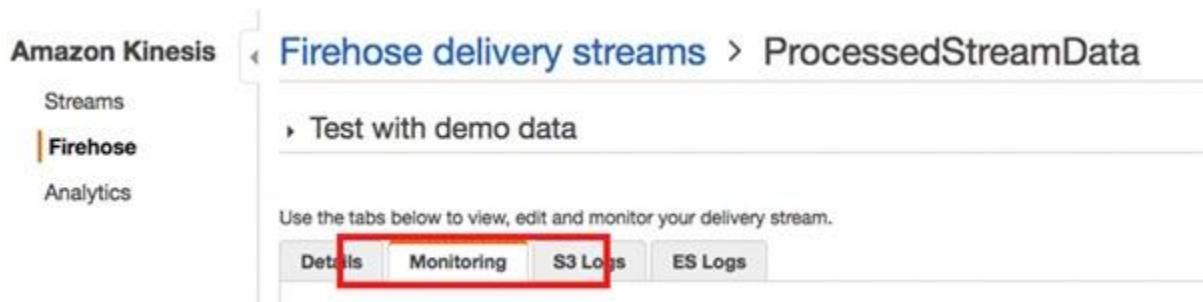
☒ JSON
☐ CSV

Permission to access the stream*

☒ Create / update IAM role `kinesis-analytics-AnomalyApp-us-east-1`
☐ Select from IAM roles that Kinesis Analytics can assume

CancelSave 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.

Delivery Stream metrics

Time range: Last 1 hour Period: 1

Go to [CloudWatch](#) for a complete list of Firehose metrics. [Learn more.](#)
All graphs are displayed in UTC time zone.

IncomingBytes (Sum)



IncomingRecords (Sum)



DeliveryToS3 DataFreshness (Maximum)



DeliveryToS3 Bytes (Sum)



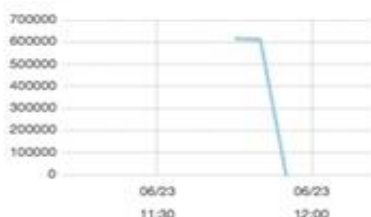
DeliveryToS3 Records (Sum)



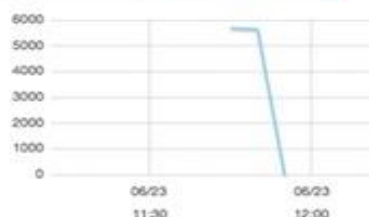
DeliveryToS3 Success (Average)



DeliveryToElasticsearch Bytes (Sum)



DeliveryToElasticsearch Records (Sum)



DeliveryToElasticsearch Success (Average)



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.

Amazon Elasticsearch Service dashboard

Create a new domain

My Elasticsearch domains

Domain	Elasticsearch version	Searchable documents	Cluster health ⓘ	Free storage space ⓘ	Minimum free storage space ⓘ	Configuration state
anomalydata	5.3	2,892	Yellow	7.24 GB	7.24 GB	Active

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.

Configure cluster Modify access policy Manage tags

Domain status **Active**

Elasticsearch version 5.3

Endpoint search-t1anomalydata- i.us-east-1.es.amazonaws.com

Domain ARN arn:aws:es:us-east-1:255742051750:domain/t1anomalydata

Kibana search-t1anomalydata- i.us-east-1.es.amazonaws.com/_plugin/kibana/

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

kibana 1,385 hits

iotvalue=10

Discover

Visualize

Dashboard

Timeline

Dev Tools

Management

Selected Fields

? _source

Available Fields

ANOMALY_SCORE

t _id

t _index

_source

iotName: DemoSensor iotValue: 10 ANOMALY_SCORE: 0 created: 2017-06-12 962327277570.0 _type: sensortype _index: anomalydata _score: 2.339

iotName: DemoSensor iotValue: 10 ANOMALY_SCORE: 0 created: 2017-06-12 975007416322.0 _type: sensortype _index: anomalydata _score: 2.339

iotName: DemoSensor iotValue: 10 ANOMALY_SCORE: 0 created: 2017-06-12

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.

