Workshop: Building a Streaming Data Platform on AWS

AWS Pop-up Loft | San Francisco

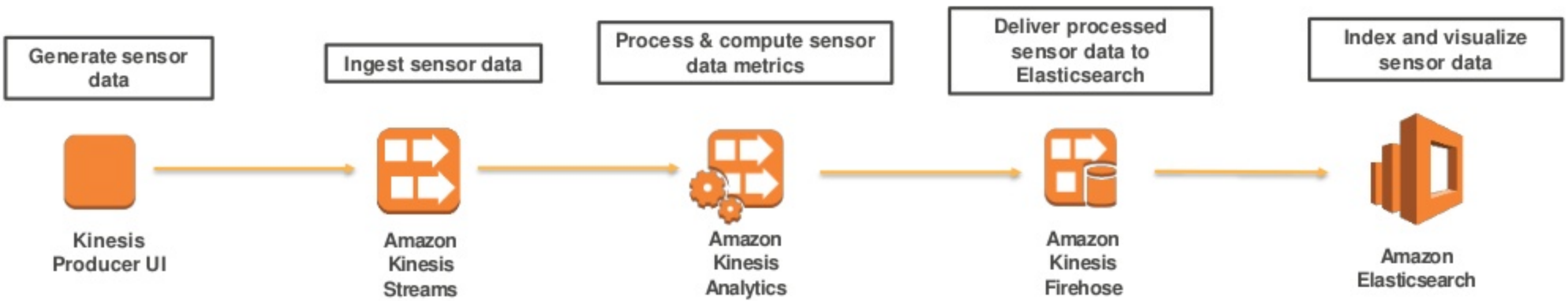
4/5/17

Before we get started

Download this guide:

amzn.to/loft-workshop

Your Streaming Application Architecture



Prep step
Create an Amazon Elasticsearch Cluster

Prep – Create an Amazon Elasticsearch Cluster

1. Open the Amazon ES console at https://console.aws.amazon.com/es/home?region=us-west-2
2. If you have not previously created an Amazon ES Domain, choose Get started. Otherwise, choose **Create a new domain**.
3. On **Define domain** screen:

a. For **Elasticsearch domain name**, enter `kinesis-workshop`

b. For **Elasticsearch version**, leave it set to the default value (5.1)

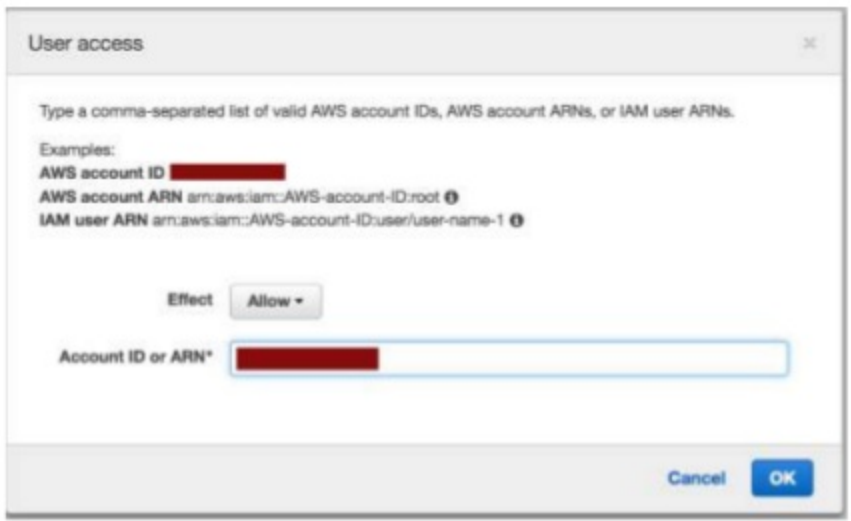
Prep – Create an Amazon Elasticsearch Cluster

4. On the **Configure cluster** screen (see below), leave all settings as their default values and click **Next**.

Prep – Create an Amazon Elasticsearch Cluster

5. In **Set up access policy** page, select **Allow or deny access to one or more AWS accounts or IAM users**

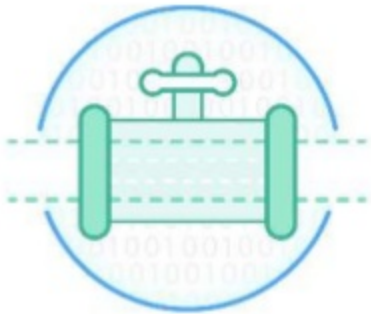
6. In the pop-up window, type in your account ID, click **OK**



7. Review the cluster configuration in the next page, confirm and create the cluster.

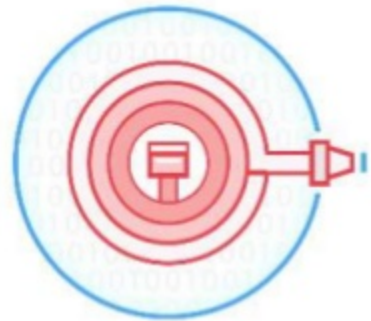
Amazon Kinesis Platform Overview

Amazon Kinesis makes it easy to work with real-time streaming data



Amazon Kinesis Streams

- For Technical Developers
- Collect and stream data for ordered, replayable, real-time processing



Amazon Kinesis Firehose

- For all developers, data scientists
- Easily load massive volumes of streaming data into Amazon S3, Redshift, ElasticSearch

Amazon Kinesis Analytics

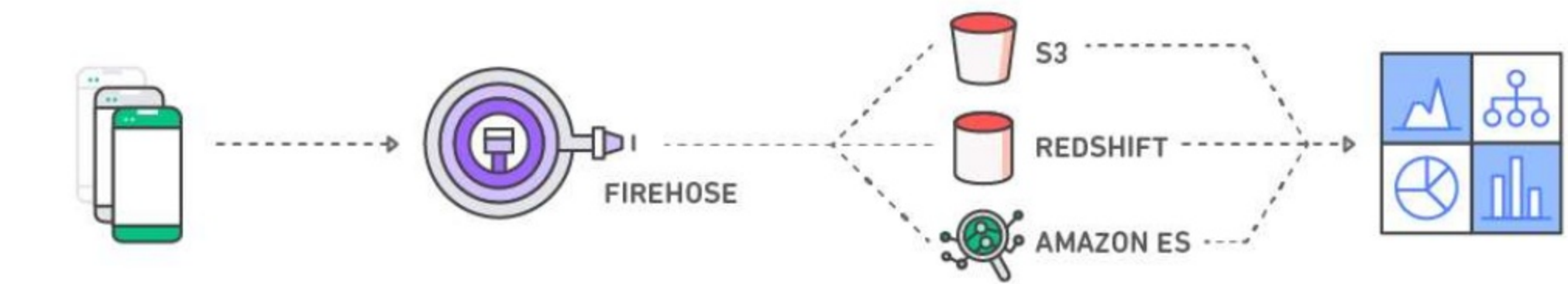
- For all developers, data scientists
- Easily analyze data streams using standard SQL queries

Amazon Kinesis Streams

- Reliably ingest and durably store streaming data at low cost
- Build custom real-time applications to process streaming data

Amazon Kinesis Firehose

- Reliably ingest and deliver batched, compressed, and encrypted data to S3, Redshift, and Elasticsearch
- Point and click setup with zero administration and seamless elasticity



Amazon Kinesis Analytics

- Interact with streaming data in real-time using SQL
- Build fully managed and elastic stream processing applications that process data for real-time visualizations and alarms

Capture streaming data with Kinesis Streams or Kinesis Firehose

Run standard SQL queries against data streams

Kinesis Analytics can send processed data to analytics tools so you can create alerts and respond in real-time

Amazon Kinesis Analytics: Service Overview

Kinesis Analytics

Easy to use

Automatic elasticity

Real-time processing

Pay for only what you use

Standard SQL for analytics

Use SQL to build real-time applications

Connect to streaming source

Easily write SQL code to process streaming data

Continuously deliver SQL results

Connect to streaming source

- Streaming data sources include Kinesis Firehose or Kinesis Streams
- Input formats include JSON, .csv, variable column, unstructured text
- Each input has a schema; schema is inferred, but you can edit
- Reference data sources (S3) for data enrichment

Write SQL code

- Build streaming applications with one-to-many SQL statements
- Robust SQL support and advanced analytic functions
- Extensions to the SQL standard to work seamlessly with streaming data
- Support for at-least-once processing semantics

Continuously deliver SQL results

- Send processed data to multiple destinations
  - S3, Amazon Redshift, Amazon ES (through Firehose)
  - Streams (with AWS Lambda integration for custom destinations)
- End-to-end processing speed as low as sub-second
- Separation of processing and data delivery

Activity 1
Create and Populate a Kinesis Stream

Activity 1: Create and populate a stream

We are going to:

A. Create an Amazon Kinesis stream
B. Set up Amazon Kinesis Data Generator and Cognito user with CloudFormation
C. Write data to a Kinesis Stream

The Amazon Kinesis stream will durable ingest our IoT records and provide a temporal buffer (a stream) of records to be consumed by our Amazon Kinesis Analytics application. We will use a sample data producer called the “Amazon Kinesis Data Generator” to simulate IoT records and write them to the stream.

Activity 1-A: Create an Amazon Kinesis stream

Amazon Kinesis Streams enables you to build custom applications that process or analyze streaming data for specialized needs. You can continuously add various types of data such as clickstreams, application logs, and social media to an Amazon Kinesis stream from hundreds of thousands of sources. Within seconds, the data will be available for your Amazon Kinesis Applications to read and process from the stream.

1. Go to the Amazon Kinesis Streams Console
   https://us-west-2.console.aws.amazon.com/kinesis/home?region=us-west-2#/initial-start
2. Click “Create Stream”
3. Specify a stream name and number of shards (1) and click “Create”

Activity 1-B: Working with Kinesis Data Generator

1. Go to the **Kinesis Data Generator Help** section at

https://awslabs.github.io/amazon-kinesis-data-generator/web/help.html

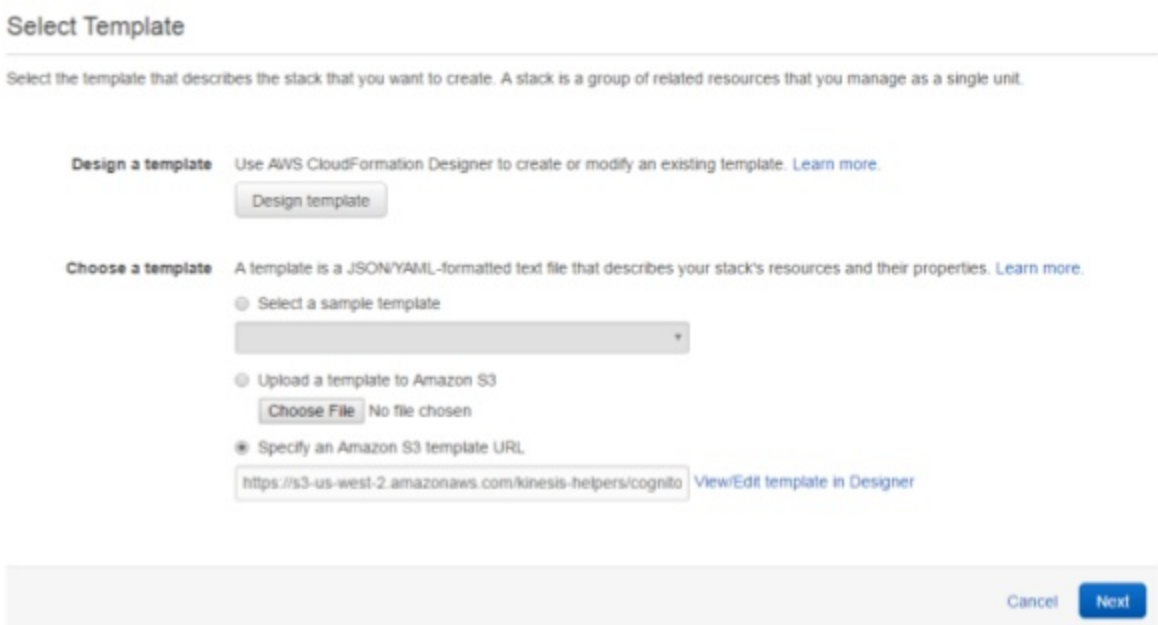
2. **Click "Create Cognito User with CloudFormation"**

This link will take you to a service called AWS CloudFormation. AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion.

We use CloudFormation to create the necessary user credentials for you to use the Kinesis Data Generator.

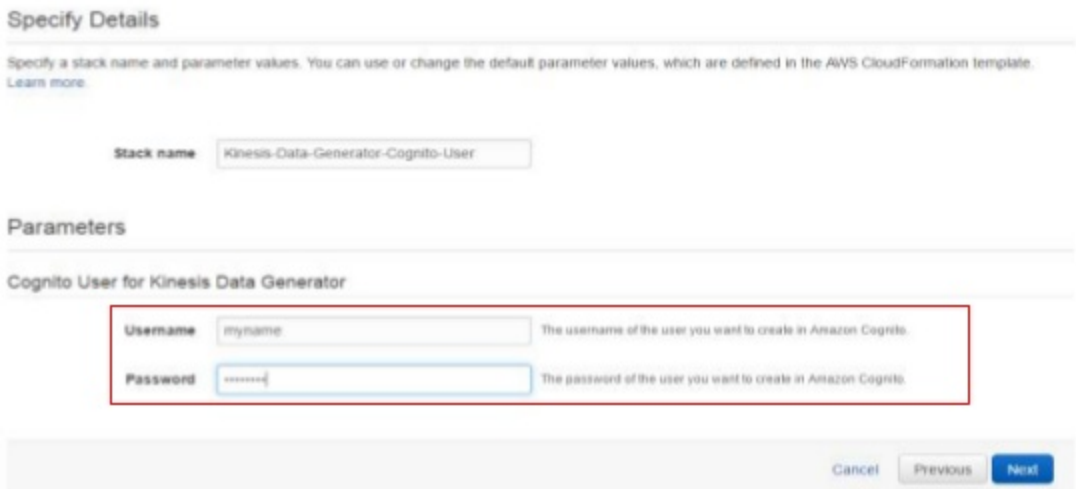
Activity 1-B: Working with Kinesis Data Generator

3. On the next screen, **click next**. (Here we are using a template stored in an Amazon S3 bucket to create the necessary credentials).



Activity 1-B: Working with Kinesis Data Generator

4. **Specify a user name and password (and remember them!), and then click next.** This user name and password will be used to sign in to the Kinesis Data Generator. **The password must be at least 6 alpha-numeric characters, and contain at least one number.**



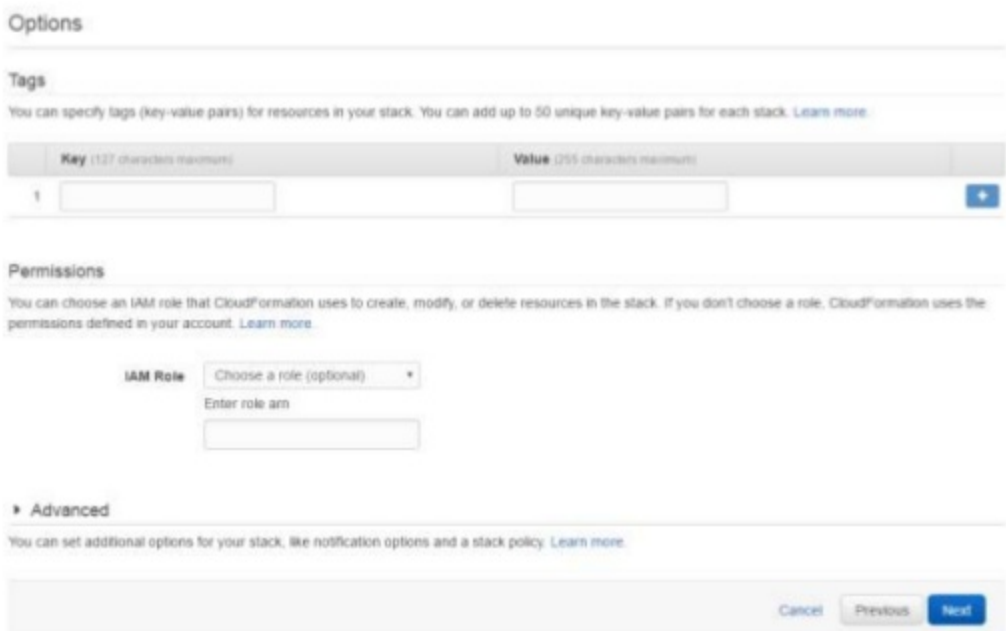
Activity 1-B: Working with Kinesis Data Generator

We use a service called Amazon Cognito to create these credentials.

Amazon Cognito lets you easily add user sign-up and sign-in to your mobile and web apps. With Amazon Cognito, you also have the options to authenticate users through social identity providers such as Facebook, Twitter, or Amazon, with SAML identity solutions, or by using your own identity system.

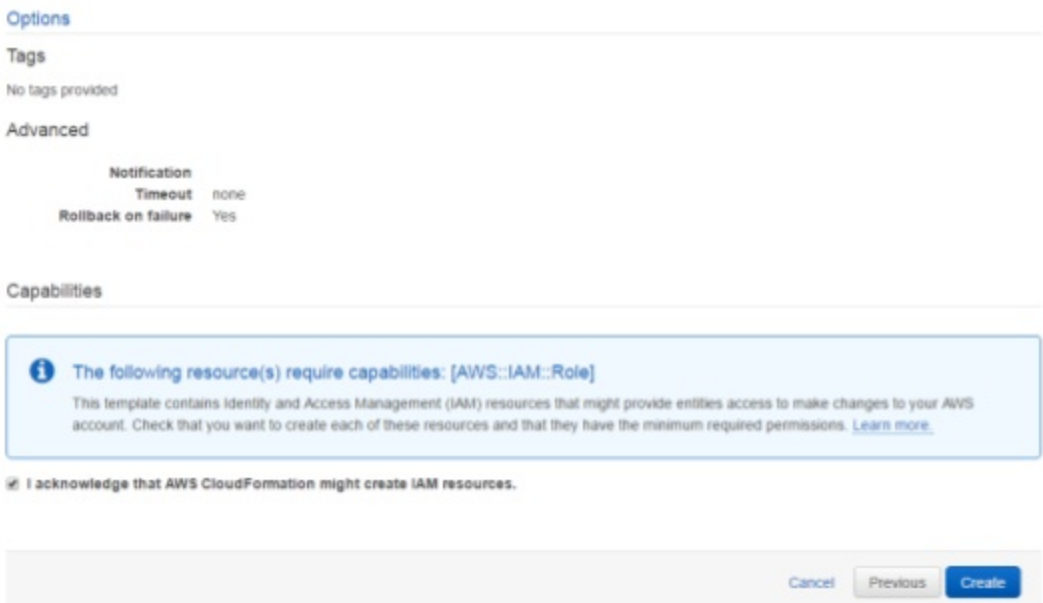
Activity 1-B: Working with Kinesis Data Generator

5. The next screen has some additional options for the CloudFormation stack which are not needed. **Click next.**



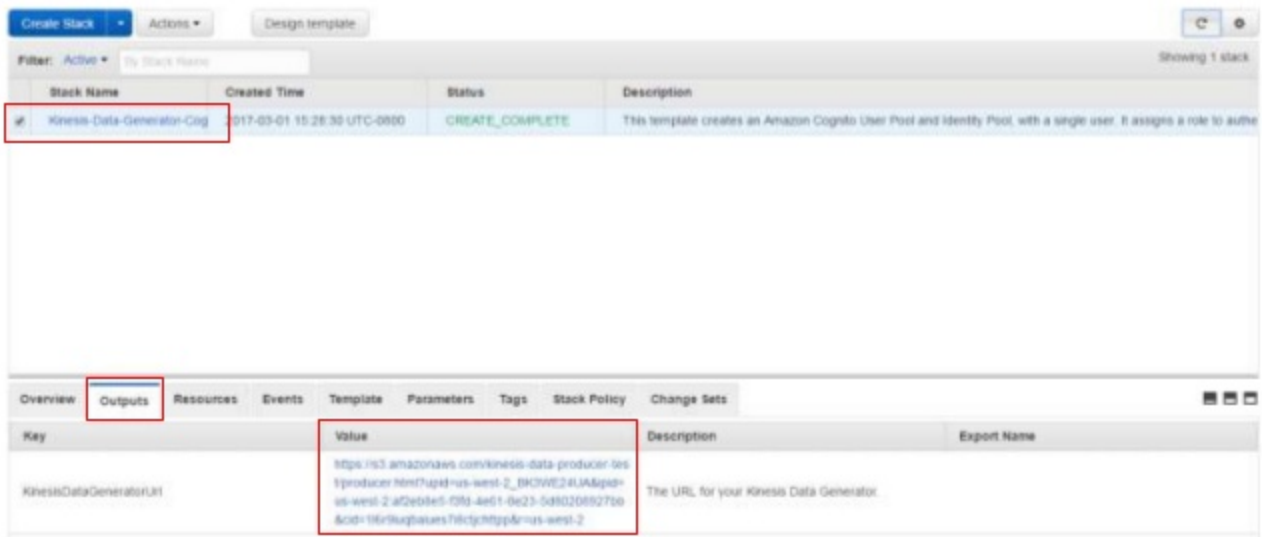
Activity 1-B: Working with Kinesis Data Generator

6. This screen is a review screen so you can verify you have selected the correct options. When you are ready, **check the "acknowledge" button and click create.**



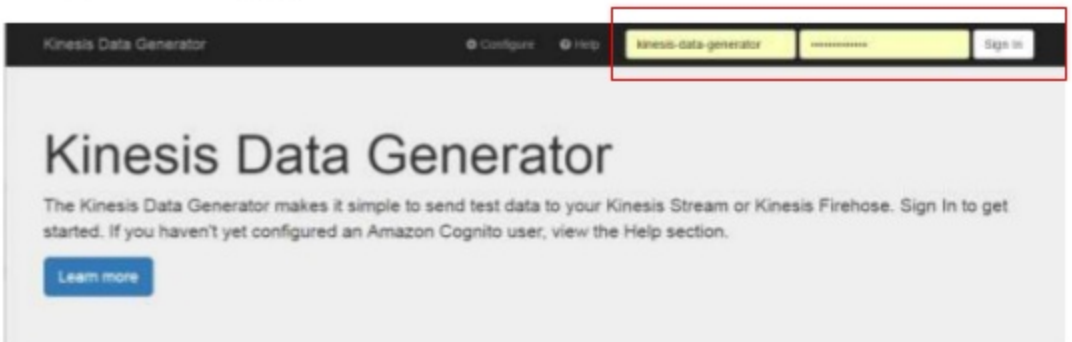
Activity 1-B: Working with Kinesis Data Generator

7. You will be taken to a screen that shows the stack creation process. (You may have to refresh the page). In approximately one minute, the stack will complete and you will see "CREATE_COMPLETE" under status. **Once this occurs, a) select the template, b) select the outputs tab, c) click the link to navigate to your very own Kinesis Data Generator.**



Activity 1-B: Working with Kinesis Data Generator

8. Login using your user name and password



9. Select the region “us-west-2” (Oregon)
10. Select the Kinesis stream you crated in Activity 1-A
11. Specify a data rate (Records per second). Please choose a number less than 10.

Activity 1-C: Write data to a Kinesis Stream

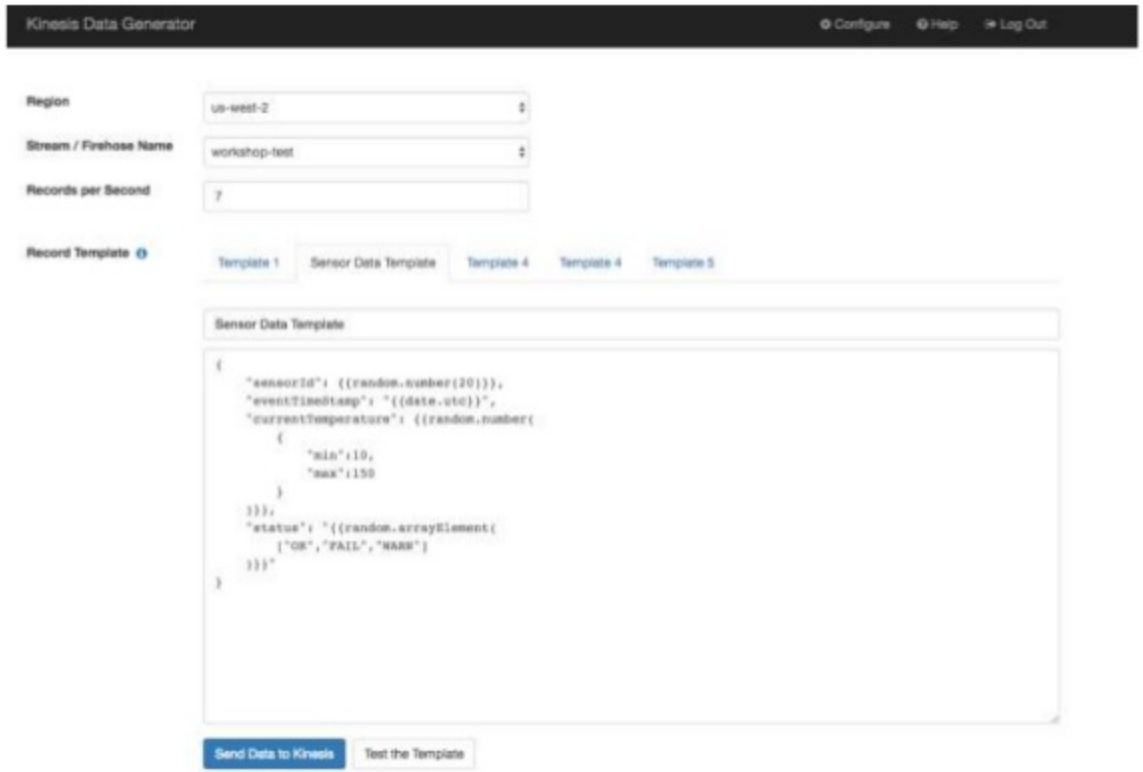
Download the data record template from here. Copy and paste the text into the text editor:

```
{
    "sensorId": {{random.number(20)}},
    "eventTimeStamp": "{{date.utc}}",
    "currentTemperature": {{random.number(
        {
            "min":10,
            "max":150
        }
    )}},
    "status": "{{random.arrayElement(
        ["OK","FAIL","WARN"]
    )}}"
}
```

Note: The tool will drain your battery. If you don't have a power cord, only keep it running while we are completing the activities.

The tool will generate records based upon the above format. It generates records based on specified rate, and sends them in a single command using the PutRecords API.

Success!


Kinesis Data Generator
Configure
Help
Log Out
Region
us-west-2
Stream / Firehose Name
workshop-test
Records per Second
7
Record Template
Template 1
Sensor Data Template
Template 4
Template 4
Template 5
Sensor Data Template
{
"sensorId": {{random.number(20)}},
"eventTimeStamp": "{{date.utc}}",
"currentTemperature": {{random.number(
{
"min":10,
"max":150
}
)}},
"status": "{{random.arrayElement(
["OK","FAIL","NAAN"]
)}}"
}
Send Data to Kinesis
Test the Template

Activity 2

Write your first query on streaming data

Activity 2: Process Data using Kinesis Analytics

We are going to:

A. Create a Kinesis Analytics applications that reads from the Kinesis stream with IoT Sensor data
B. Write several queries including de-duplication, filtering, windowed aggregation, and anomaly detection.

Download the SQL code now. You will copy and paste sections as we build the application.

Activity 2-A: Create a streaming app (pt 1)

An application is the Amazon Kinesis Analytics entity that you work with. Kinesis Analytics applications continuously read and process streaming data in real-time. You write application code using SQL to process the incoming streaming data and produce output. Then, Kinesis Analytics writes the output to a configured destination.

In Activity 2-A, we are going to create a Kinesis Analytics application that reads data from the stream you previously created. In Activity 2-B, we will write application code using SQL.

Activity 2-A: Create a streaming app (pt 2)

1. Go to the Amazon Kinesis Analytics Console
   https://us-west-2.console.aws.amazon.com/kinesisanalytics/home?region=us-west-2
2. Click **Create new application** and provide a name
3. Click **Connect to a source** and select the stream you created in the previous exercise

Activity 2-A: Create a streaming app (pt 3)

4. Kinesis Analytics will discover the schema for you. Verify the schema and click **Save and Continue**
5. Click **Go To SQL Editor** and then click **Yes, Start Application**

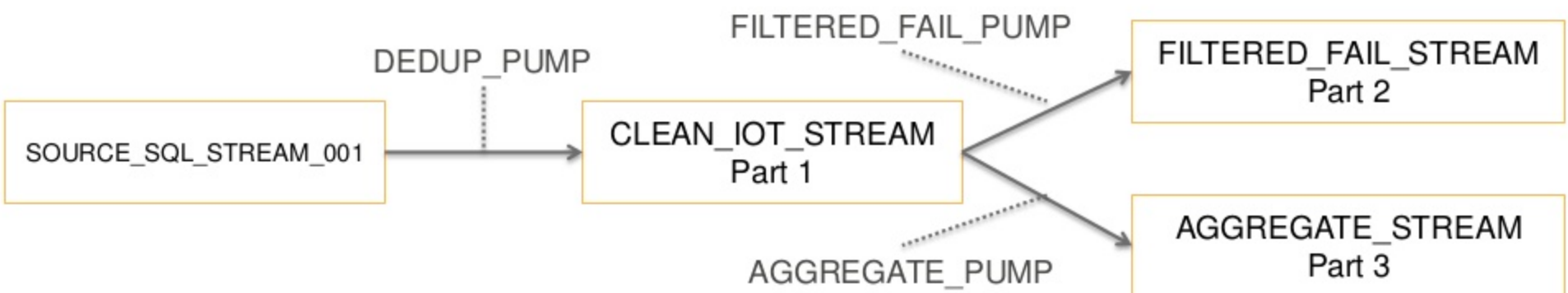
You now have a running stream processing application!

Pause and wait for the remainder of the class to reach this step.

Writing SQL over Streaming Data

Writing SQL over streaming data using Kinesis Analytics follows a two part model:

1. Create an in-application stream for storing intermediate SQL results. An in-application stream is like a SQL table, but is continuously updated.
2. Create a PUMP which will continuously read FROM one in-application stream and INSERT INTO a target in-application stream



Activity 2-B: Add your first query (part 1)

Our first query will use SELECT DISTINCT to remove any producer-side duplicates. Producer-side duplicates are caused by connectivity issues and producer retires.

In the SQL Editor, we will:

1. Create a target in-application stream to store the intermediate SQL results.
2. Create a continuous query (a “pump”) to process data from the source stream and insert it into the target in-application stream

Activity 2-B: Add your first query (part 2)

This query removes any producer side duplicates over a one second window. (In a production setting, this window would likely be larger like 10 seconds). Producer side duplicates are most often created when an HTTP request times out due to connectivity issues on the producer.

The window is defined by the following statement in the SELECT statement. Note that the ROWTIME column is implicitly included in every stream query, and represents the processing time of the application.

```
FLOOR("SOURCE_SQL_STREAM_001".ROWTIME TO SECOND)
```

(DO NOT INSERT THIS INTO YOUR SQL EDITOR)

Activity 2-B: Add your first query (part 3)

```
/* Activity 2B - Add your first query (deduplication) */
/* Step 1 - Creates a stream for intermediary SQL results */
CREATE OR REPLACE STREAM clean_iot_stream (
    sensor_id INTEGER,
    event_timestamp TIMESTAMP,
    current_temp INTEGER,
    status VARCHAR(4),
    dedup_window TIMESTAMP);

/* continued on next page (above SQL statement does nothing by itself)
*/
```

Activity 2-B: Add your first query (part 4)

```
/* Step 2 - Reads from source stream, selects distinct values over a one
second window */
CREATE OR REPLACE PUMP dedup_pump AS
INSERT INTO clean_iot_stream
SELECT STREAM DISTINCT --distinct will use the columns selected to remove
duplicates
    "sensorId",
    CAST("eventTimeStamp" AS TIMESTAMP), --casting the VARCHAR time to a
timestamp (also could have done this when we setup the schema)
    "currentTemperature",
    "status",
     FLOOR("SOURCE_SQL_STREAM_001".ROWTIME TO SECOND) --window for the distinct
keyword
FROM SOURCE_SQL_STREAM_001;
```

Activity 2-C: Add a continuous filter (part 1)

A continuous filter is a common query that uses a WHERE clause to select a portion of your data stream. For this specific query we will apply a filter on the “status” column to only select and records with equal “fail” and insert them into a subsequent stream. This common query is useful for alerting on events that match a particular query. Commonly, they are then forwarded on to other Amazon Kinesis streams, Amazon SNS topics, AWS Lambda functions, and so forth.

Again, we use two steps to accomplish this:

1. Create a target in-application stream for the filtered data.
2. Create a continuous query (a “pump”) to process data from the clean_iot-Stream and insert it into the filtered stream

Activity 2-C: Add a continuous filter (part 2)

```
/* Activity 2C - Add a continuous filter  */
/* Step 1 - Stream for intermediary SQL results */
CREATE OR REPLACE STREAM filtered_fail_stream (
    sensor_id INTEGER,
    event_timestamp TIMESTAMP,
    current_temp INTEGER,
    status VARCHAR(4));

/* Step 2 - Continuous query that filters data based upon status */
CREATE OR REPLACE PUMP filter_fail_pump AS INSERT INTO filtered_fail_stream
SELECT STREAM sensor_id, event_timestamp, current_temp, status
FROM clean_iot_stream
WHERE status = 'FAIL';
```

Activity 2-D: Calculate an aggregate metric (pt 1)

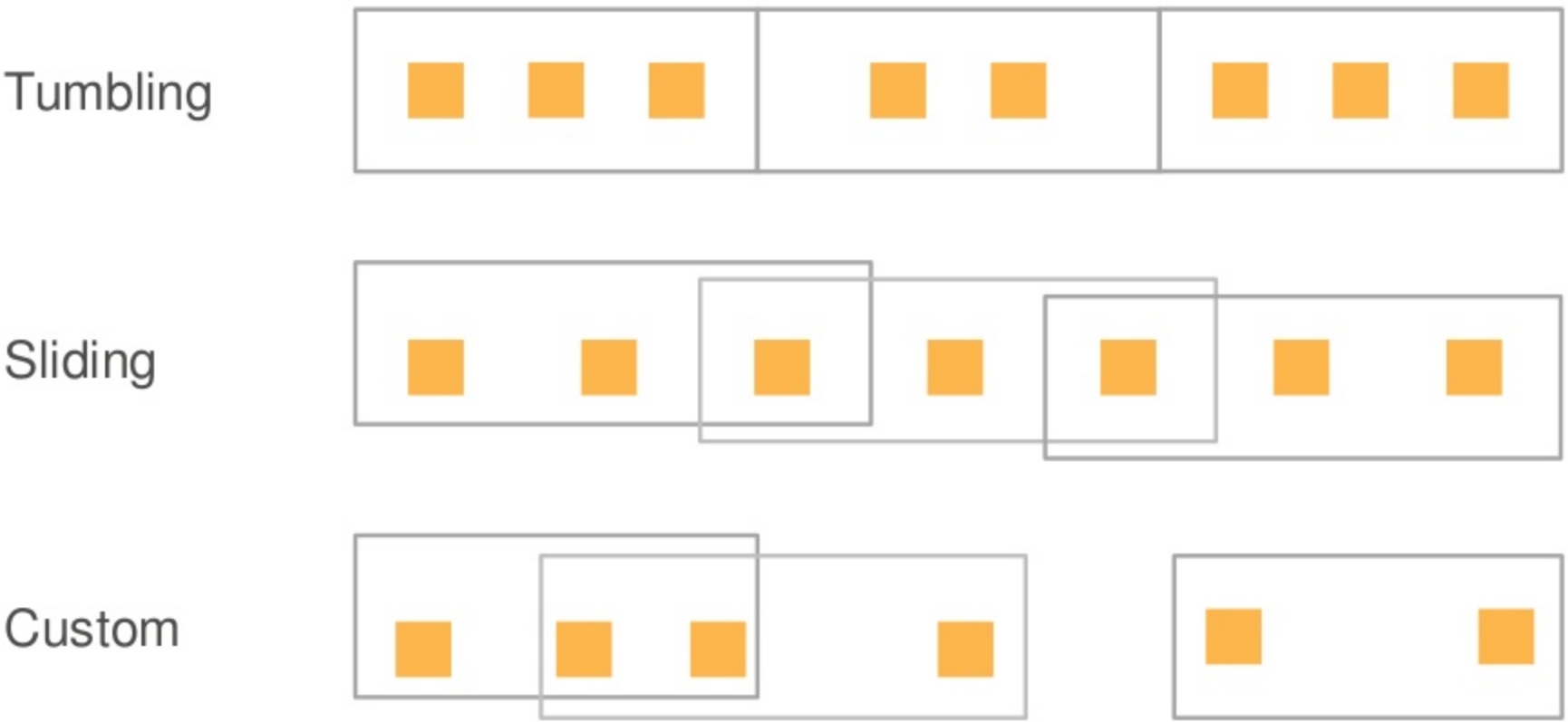
In your next query, you will calculate an aggregate metric, specifically a count using a tumbling window.

A tumbling window is similar to a periodic report, where you specify your query and a time range, and results are emitted at the end of a range. (EX: COUNT number of items by key for 10 seconds)

There are three different types of windows: Tumbling, Sliding, and Custom.

Activity 2-D: Calculate an aggregate metric (pt 2)

Different types of windows



Activity 2-D: Calculate an aggregate metric (pt 3)

Different types of windows

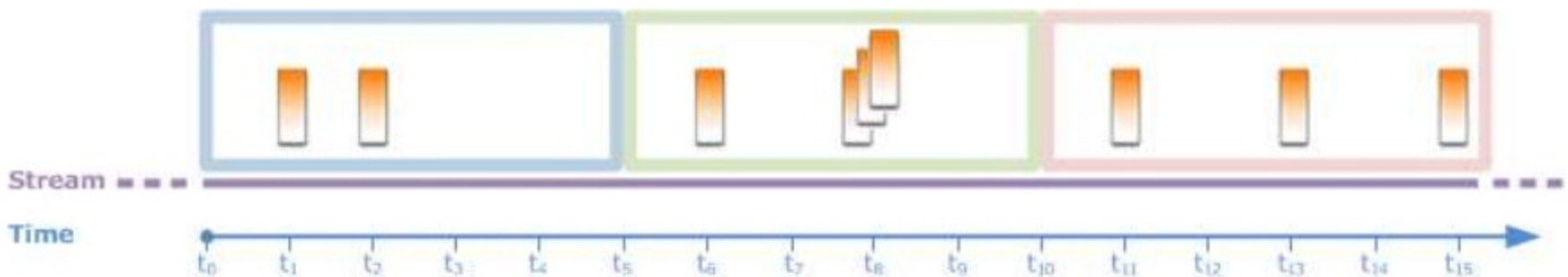
Tumbling	• Fixed size and non-overlapping • Use FLOOR() or STEP() function in a GROUP BY statement
Sliding	• Fixed size and overlapping; row boundaries are determined when new rows enter window • Use standard OVER and WINDOW clause (*i.e. count (col) OVER (RANGE INTERVAL '5' MIN*)
Custom	• Not fixed size and overlapping; row boundaries are determined by conditions • Implementations vary, but typically require two steps (Step 1 – Identify boundaries, Step 2 – perform computation)

Activity 2-D: Calculate an aggregate metric (pt 4)

Tumbling Windows

Tumbling windows are useful for periodic reports. You can use a tumbling window to compute an average number of visitors to your website in the last 5 minutes, or the maximum over the past hour. A single result is emitted for each key in the group as specified by the clause at the end of the defined window.

An important characteristic of a tumbling window is that the bounds do not overlap; the start of a new tumbling window begins with the end of the old window. The below image visually captures how a 5-minute tumbling window would aggregate results, in separate windows that are not overlapping. The blue, green, and red windows represent minutes 0 to 5, 5 to 10, and 10 to 15, respectively.



Activity 2-D: Calculate an aggregate metric (pt 5)

Tumbling windows are always included in a GROUP BY clause and use the FLOOR/STEP function.

The STEP function takes an interval to produce the periodic reports.

```
STEP(source_sql_stream_001.ROWTIME BY INTERVAL '10' SECOND)
```

You can also use the SQL FLOOR() function to achieve the same thing.

(The FLOOR function rounds to the largest value less than or equal to the specified numeric expression. To create a tumbling time window, convert a timestamp to an integer representation and put it in a FLOOR function to create the window bound.)

Activity 2-D: Calculate an aggregate metric (pt 6)

```
/* Activity 2-D - Calculate an aggregate metric  */
/* Step 1 - Stream for aggregate SQL results */
CREATE OR REPLACE STREAM aggregate_stream (
    status VARCHAR(4),
    avg_temp DECIMAL(16,2));

/* Step 2 - Continuous query for  aggregate SQL results */
CREATE OR REPLACE PUMP aggregate_pump AS INSERT INTO aggregate_stream
SELECT STREAM status, AVG(current_temp)
FROM clean_iot_stream
GROUP BY status, STEP(CLEAN_IOT_STREAM.ROWTIME BY INTERVAL '10'
SECOND);
```

Activity 2-E: Using event time (pt 1)

All of our previous window queries were using ROWTIME, which is the processing time of the application. The processing time is useful in many real-time use cases like the previous exercise (sliding window and filter). However, there are many cases where you need to use your data's event time. An event time is a timestamp generated by a data producer when the event was created.

A key problem with using event time is that events arrive out of order. You could perform sorts on the data, but sorts will leave events out which you need to reconcile later. An alternative approach is to use both processing and event time in your windows.

Activity 2-E: Using event time (pt 2)

```
/* Activity 2-E - Using event time  */
/* Step 1 - Stream for aggregate SQL results */
CREATE OR REPLACE STREAM aggregate_event_time_stream (
     event_timestamp TIMESTAMP, -- Add this column to your query
     status VARCHAR(4),
     avg_temp DECIMAL(16,2));

/* Step 2 - Continuous query for  aggregate SQL results */
CREATE OR REPLACE PUMP event_time_pump AS INSERT INTO aggregate_event_time_stream
SELECT STREAM STEP(event_timestamp BY INTERVAL '10' SECOND),
    status,
    AVG(current_temp)
FROM clean_iot_stream
GROUP BY status,
    STEP(clean_iot_stream.event_timestamp BY INTERVAL '10' SECOND),
    STEP(clean_iot_stream.ROWTIME BY INTERVAL '10' SECOND);
```

Activity 2-F: Anomaly detection (pt 1)

Kinesis Analytics includes some advanced algorithms in it that are extensions to the SQL language. These include approximate count distinct (hyperloglog), approximate top K (space saving), and anomaly detection (random cut forest).

The random cut forest algorithm will detect anomalies in real-time on multi-dimensional data sets. You pass the algorithm n number of numeric fields and the algorithm produces an anomaly score on your stream data. Higher scores are more anomalous.

The minimum anomaly score is 0 and the maximum is log2 s, where s is the subsample size parameter passed to random cut forest (third parameter). You will have to try the algorithm on your data to get a feel for the anomaly score, as the score is data-dependent.

Activity 2-F: Anomaly detection (pt 2)

```
/* Activity 2-F - Anomaly detection  */
/* Step 1 - Creates a stream for anomaly scores */
CREATE OR REPLACE STREAM anomaly_stream (
     sensor_id INTEGER,
     event_timestamp TIMESTAMP,
     current_temp INTEGER,
     status VARCHAR(4),
     anomaly_score DECIMAL(16,2));

/* Step 2 - Compute anomaly score */
-- Compute an anomaly score for each record in the input stream
-- using Random Cut Forest
CREATE OR REPLACE PUMP anomaly_pump AS INSERT INTO anomaly_stream
      SELECT STREAM sensor_id, event_timestamp, current_temp, status, anomaly_score
      FROM TABLE(RANDOM_CUT_FOREST(
              CURSOR(SELECT STREAM * FROM clean_iot_stream)));
```

Activity 2-G: clean up

For the next part of the workshop, we will stream **ANOMALY_STREAM** you created in Activity 2-F to Elasticsearch for further analysis.

In the SQL editor, you can remove code related to the other in-application streams we created:

- **FILTERED_FAIL_STREAM** from Activity 2-C
- **AGGREGATE_STREAM** from Activity 2-D
- **AGGREGATE_EVENT_TIME_STREAM** from Activity 2-E (keep the **CLEAN_IOT_STREAM** from Activity 2-B because it feeds into the **ANOMALY_STREAM**)

Activity 3

Stream results from Kinesis Analytics into Elasticsearch through Kinesis Firehose

Activity 3-A: Create a Firehose Delivery Stream (pt1)

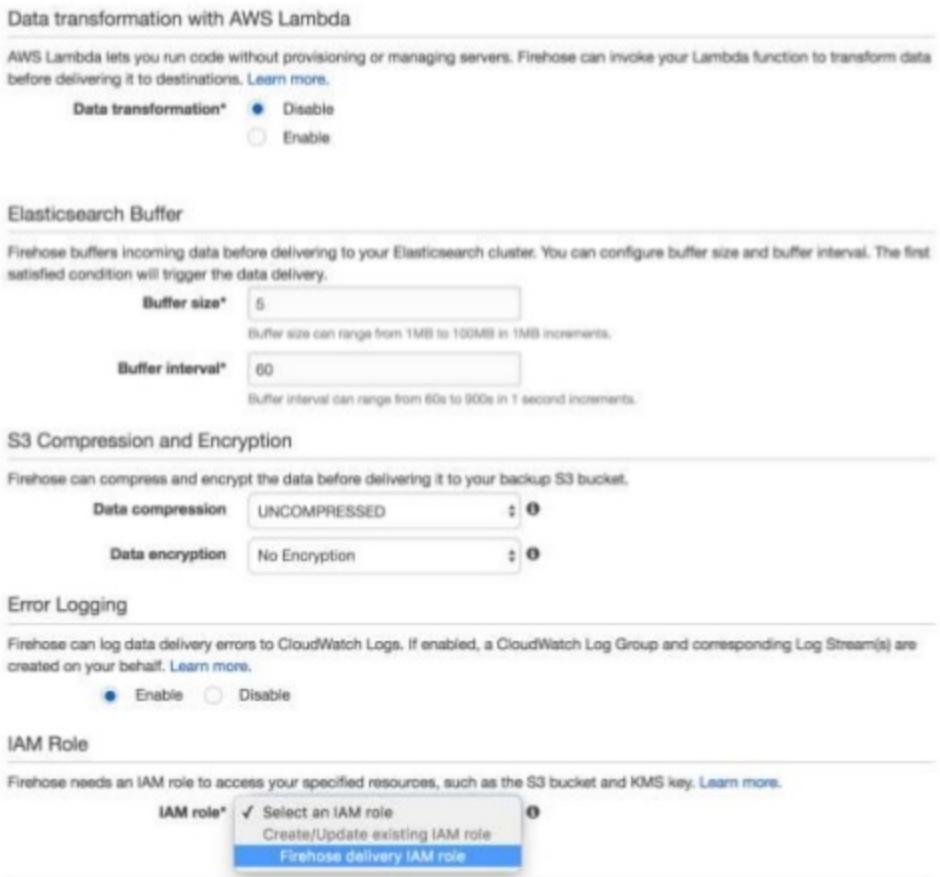
1. Open the Amazon Kinesis console at https://console.aws.amazon.com/kinesis/home?region=us-west-2
2. Click **Go to Firehose**.
3. Click **Create Delivery Stream**.
4. On the **Create Delivery Stream** screen:
   - For **Destination**, choose **Amazon Elasticsearch Service**.
   - For **Delivery stream name**, enter **`processed-sensor-data`**
   - For **Elasticsearch domain**, choose the domain you created in Prep Step (**`kinesis-workshop`**)
   - For **Index**, enter **`sensor-data`**.
   - For **Index rotation**, leave it set to “NoRotation” (default).
   - For **Type**, enter **`sensor-w-anomaly-score`**.
   - For **Retry duration (sec)**, leave it set to “300” (default).

Activity 3-A: Create a Firehose Delivery Stream (pt2)

- Under **Backup S3 bucket**, for **Backup mode**, select **Failed Documents Only**.
- For **S3 bucket**, choose **New S3 bucket**. You will be prompted to provide additional information for the new bucket: For **Bucket name**, use a unique name. Amazon S3 bucket names are required to be globally unique.
- For **Region**, choose **Oregon**. Click **Create Bucket**.
- For **S3 prefix**, leave it blank (default).
- Click **Next**.

Activity 3-A: Create a Firehose Delivery Stream (pt3)

5. On the **Configuration** screen:
   - Under **Elasticsearch Buffer**, change the **Buffer interval** to **60**
   - Leave rest of the fields set to their default values.
   - You will need an IAM role so that Amazon Kinesis Firehose can write to your Amazon ES domain on your behalf. For **IAM role**, choose **Create/Update Existing IAM Role**.



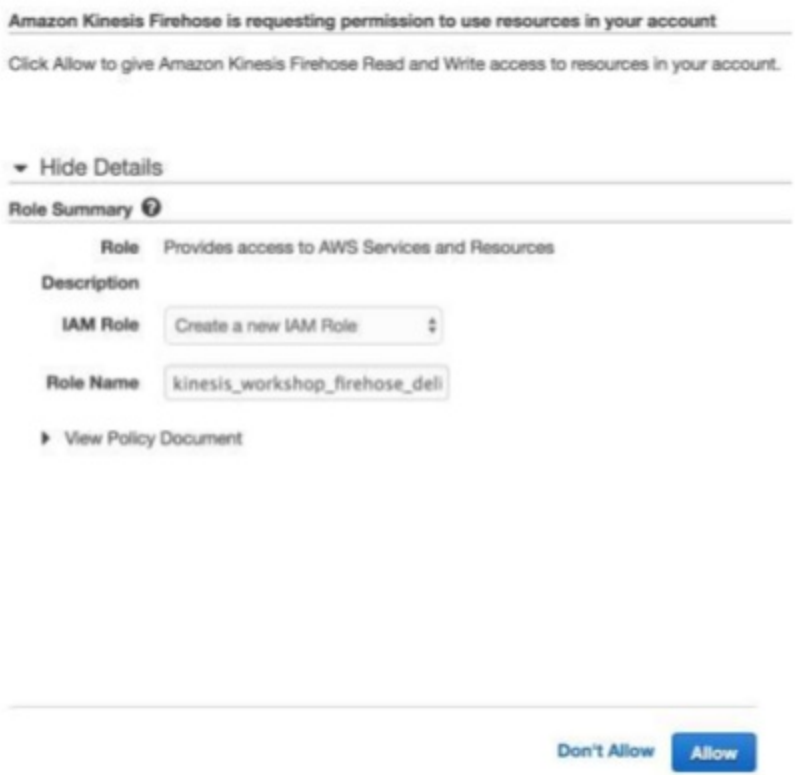
Activity 3-A: Create a Firehose Delivery Stream (pt4)

6. You will now be redirected to the IAM console:

For **IAM Role**, choose **Create a new IAM Role**.

For **Role Name**, enter **`kinesis_workshop_firehose_delivery_role`**

Click **Allow**.



Activity 3-A: Create a Firehose Delivery Stream (pt5)

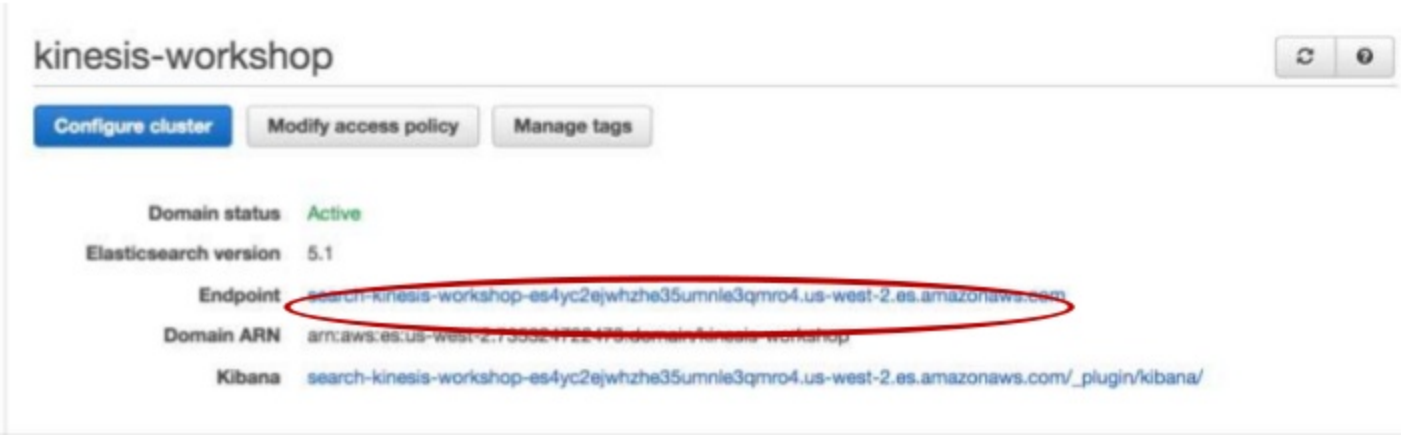
7. Click **Next**, review the details of the Amazon Kinesis Firehose delivery stream and choose **Create Delivery Stream**.

Activity 3-B: Set up Elasticsearch/Kibana Proxy (pt1)

1. Install the 'aws-es-kibana' npm module
   npm install -g aws-es-kibana

   (if you don't have npm installed, go to the Node.js website and download the latest **LTS** version)

2. Find the Endpoint of your Elasticsearch in the console



Activity 3-B: Set up Elasticsearch/Kibana Proxy (pt2)

3. If you have your AWS CLI configured with credentials, skip this step. Otherwise, set up credentials following steps here: https://github.com/santthosh/aws-es-kibana

4. Run the proxy (endpoint you copied from step 2)

   ```
   aws-es-kibana <cluster-endpoint>
   ```

Activity 3-C: Specify Elasticsearch Mapping

While the ES/Kibana proxy is running, define the **schema** for the data that will be ingested into Elasticsearch.

In a separate terminal window, type the curl command on the right. Download the text for the command from here.

```
curl -XPUT 'localhost:9200/sensor-data' -H 'Content-Type:
application/json' -d'
{
 "mappings": {
  "sensor-w-anomaly-score": {
   "properties": {
      "CURRENT_TEMP":{
         "type": "integer"
      },
      "EVENT_TIMESTAMP": {
         "type": "date",
         "format": "yyyy-MM-dd HH:mm:ss.SSS"
      },
      "SENSOR_ID": {
         "type": "integer"
      },
      "STATUS": {
         "type": "string",
         "index": "not_analyzed"
       },
      "ANOMALY_SCORE":{
         "type": "float"
      }
    }
   }
  }
}
'
```

Activity 3-D: Configure Destination Stream on Kinesis Analytics Application

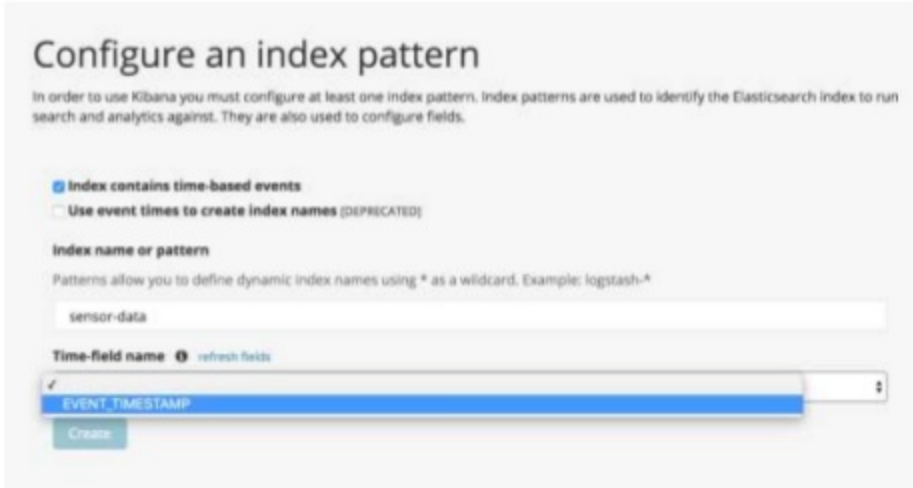
1. Go back to the Amazon Kinesis Analytics console at https://us-west-2.console.aws.amazon.com/kinesisanalytics/home?region=us-west-2
2. Go to **Application Details** of the Kinesis Analytics application you created in Activity 2
3. Under **Destination**, click the edit sign
4. In the destination configuration page:
   a. Select the **processed-sensor-data Firehose delivery stream you created in Activity 3-A**
   b. For **In your SQL, refer to this stream as,** type **ANOMALY_STREAM** (the in-application stream you created in Activity 2-F)
   c. For **Output Format**, keep the value as **JSON**
   d. For **Permission to access the stream*,** keep the default selected **Create / update IAM role** option

Activity 3-E: Visualize Data in Kibana

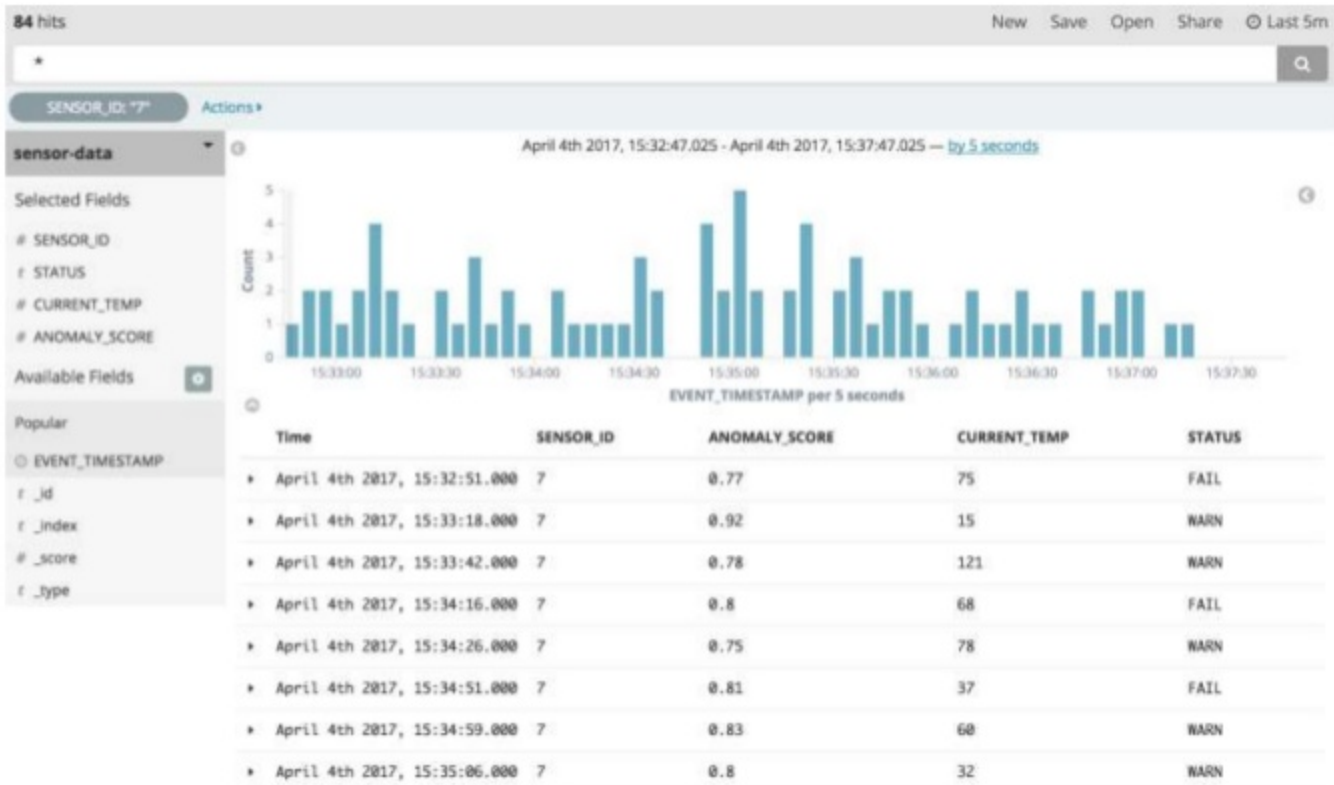
1. Open the Kibana link provided in the ES/Kibana Proxy you started in Activity 3-B in a browser:

   http://127.0.0.1:9200/_plugin/kibana/

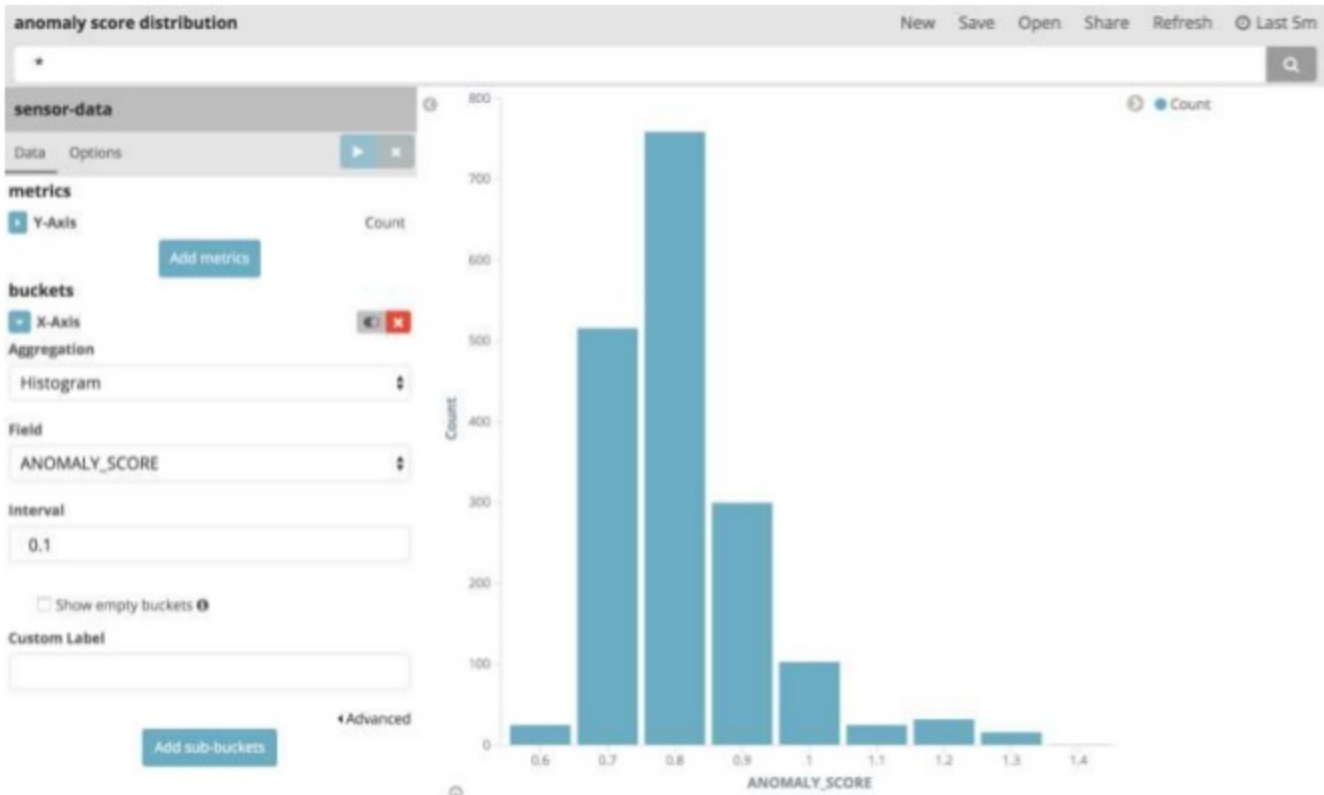
2. On **Configure an index pattern** page**:**
   - Check **Index contains time-based events**
   - For **Index name,** type **sensor-data**
   - For **Time-field name**, select **EVENT_TIMESTAMP field**



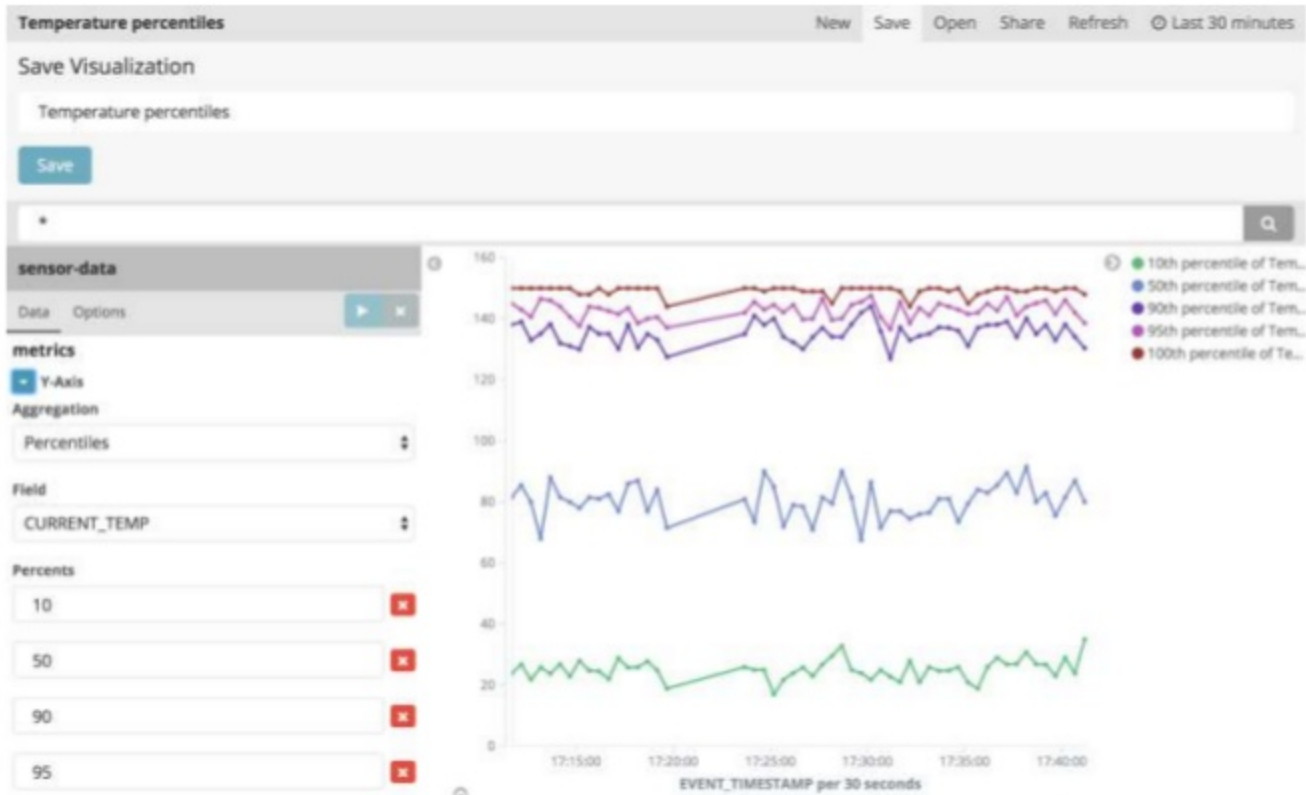
Activity 3-E: Visualize Data in Kibana



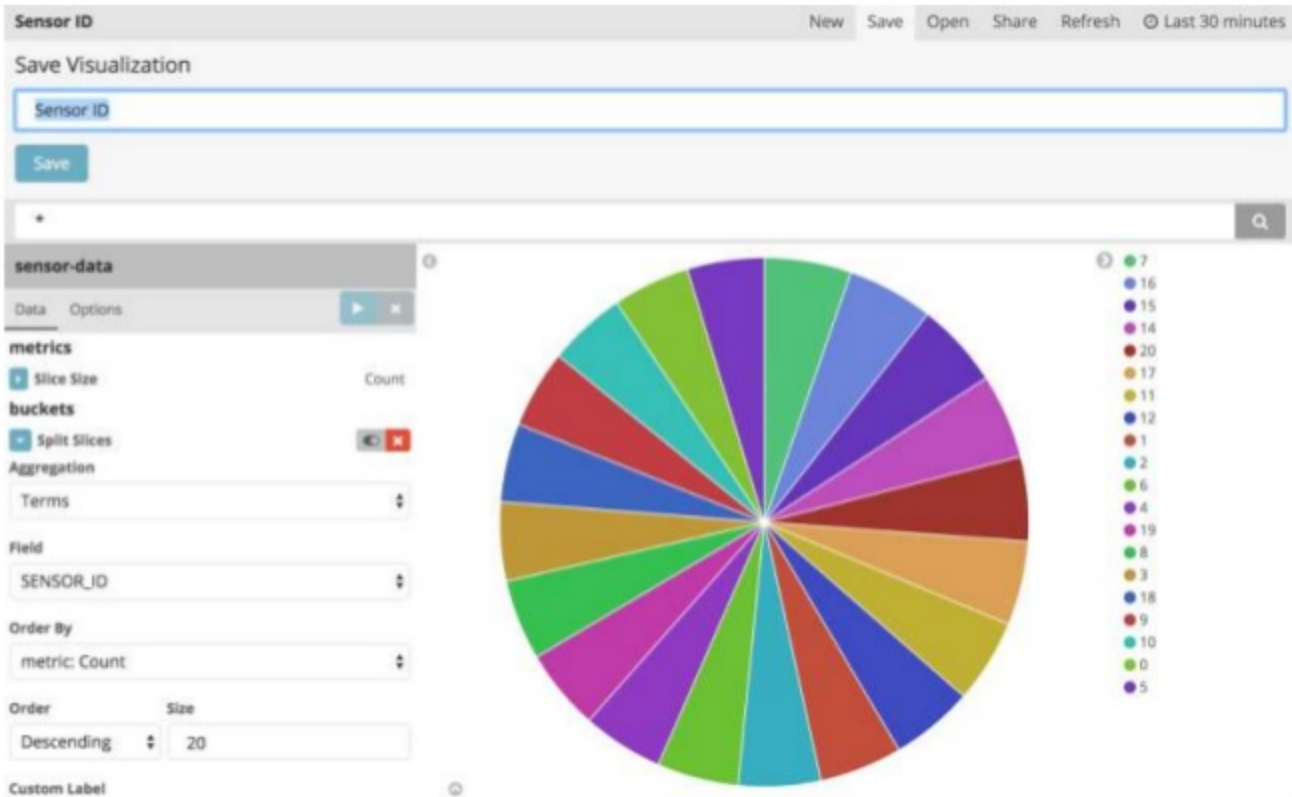
Activity 3-E: Visualize Data in Kibana



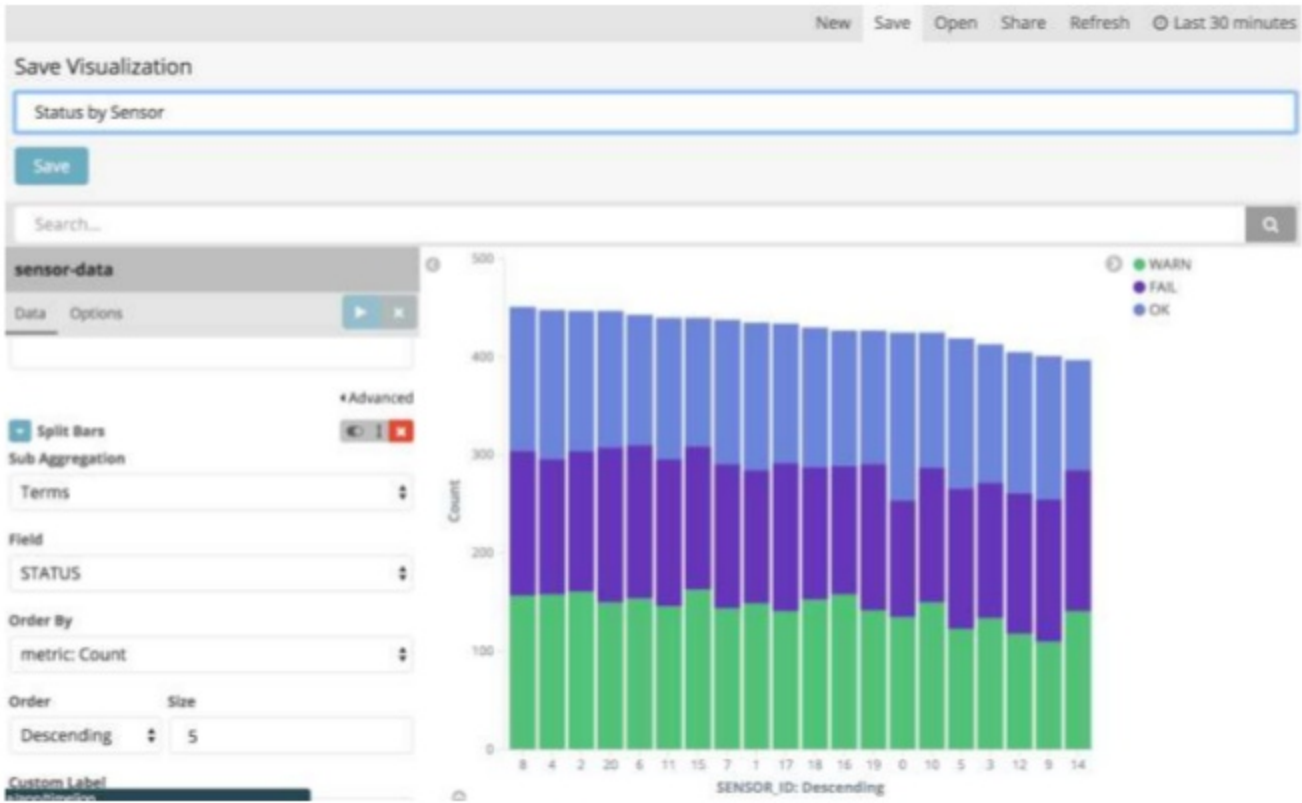
Activity 3-E: Visualize Data in Kibana



Activity 3-E: Visualize Data in Kibana



Activity 3-E: Visualize Data in Kibana



Activity 3-F: Clean up

1. Stop the data producer and ES proxy
2. Go to the Kinesis Analytics console, stop and/or delete your application
3. Go to the Kinesis Streams console, delete your stream
4. Go to the Kinesis Firehose console, delete your delivery stream
5. Go to Elasticsearch console, delete your cluster
6. Go to Cloudformation console, delete the ‘Kinesis-Data-Generator-Cognito-User’ stack