Real-Time IoT Device Monitoring with Kinesis Data Analytics

Implementation Guide

Real-Time IoT Device Monitoring with Kinesis Data Analytics: Implementation Guide

Table of Contents

Home 1
Overview 2
    Cost 2
    Architecture 2
Components 4
    Amazon Kinesis Data Analytics Application 4
    Amazon DynamoDB 4
    Dashboard 4
    Anomaly Detection 5
Considerations 6
    Solution Updates 6
    Regional Deployment 6
Template 7
Deployment 8
    Launch the Stack 8
Security 10
    Encryption 10
    Amazon CloudFront 10
Resources 11
Appendix A: Code Components 12
    SQL Query 12
    Lambda 12
    JavaScript 13
    HTML Element 13
Appendix B: Customizing the Dashboard 14
    Step 1. Update the Demo Script to Send New Data 14
    Step 2. Add the Metric to the Source Schema 14
    Step 3. Modify the Application's SQL Code 15
    Step 4. Update the Lambda Code 15
    Step 5. Update the JavaScript Code 16
    Step 6. Update the Website Assets 17
Appendix C: Anonymous Data 18
Source Code 19
Revisions 20
    20

Deploy a reference implementation to collect, process, analyze, and visualize IoT device connectivity

May 2018 (last update (p. 20): December 2019)

This implementation guide discusses architectural considerations and configuration steps for deploying Real-Time IoT Device Monitoring with Kinesis Data Analytics on the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, developers, and DevOps professionals who have practical experience architecting on the AWS Cloud.

Overview

Monitoring IoT devices in real-time can provide valuable insight that can help you maintain the reliability, availability, and performance of your IoT devices. You can track time series data on device connectivity and activity. This insight can help you react quickly to changing conditions and emerging situations.

Amazon Web Services (AWS) offers a comprehensive set of powerful, flexible, and simple-to-use services that enable you to extract insights and actionable information in real time. Amazon Kinesis is a platform for streaming data on AWS, offering key capabilities to cost-effectively process streaming data at any scale. Kinesis capabilities include Amazon Kinesis Data Analytics, the easiest way to process streaming data in real time with standard SQL without having to learn new programming languages or processing frameworks.

To help customers more easily leverage Kinesis Data Analytics, AWS offers the Real-Time IoT Device Monitoring with Kinesis Data Analytics solution, a reference implementation that automatically provisions the services necessary to collect, process, analyze and visualize IoT device connectivity and activity data in real-time. This solution is designed to provide a framework for analyzing and visualizing metrics, allowing you to focus on adding new metrics rather than managing the underlying infrastructure.

Real-Time IoT Device Monitoring with Kinesis Data Analytics uses AWS IoT to ingest device data, Amazon Kinesis Data Firehose to archive the data, Kinesis Data Analytics to compute metrics in real-time, and Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB to durably store metric data. The solution features a dashboard that visualizes your device connectivity metrics in real-time.

Cost

You are responsible for the cost of the AWS services used while running this reference deployment. As of the date of publication, the baseline cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$500 per month**. This cost estimate assumes the solution will monitor 1,000 devices that are always connected, and that send data once every 10 seconds (about 250M messages per month). Note that the monthly cost will vary depending on the number of connected devices and how often those devices send data. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

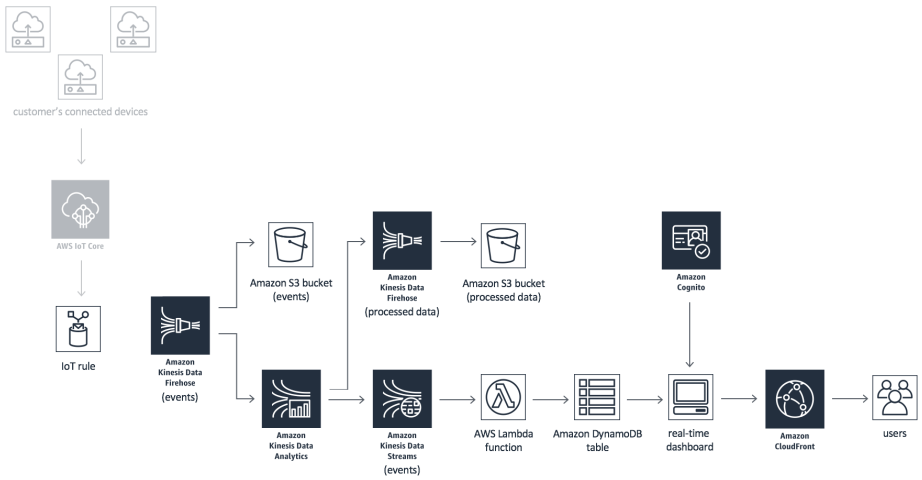

Figure 1: Real-Time IoT Device Monitoring with Kinesis Data Analytics architecture

The AWS CloudFormation template deploys an AWS IoT rule, two Amazon Kinesis Data Firehose delivery streams, Amazon Simple Storage Service (Amazon S3) buckets, a Kinesis data analytics application, an AWS Lambda function, an Amazon DynamoDB table, an Amazon Cognito user pool, an Amazon CloudFront distribution, and a real-time device monitoring dashboard to securely read and display the device connectivity and activity metrics stored in the DynamoDB table.

When AWS IoT ingests data from your connected devices, the AWS IoT rule sends the data to a Kinesis data delivery stream. The delivery stream archives the events in an Amazon S3 bucket and sends the data to a Kinesis data analytics application for processing. The application sends the processed data to a Lambda function that sends it in real-time to a DynamoDB table to be stored. The application also sends processed data to a second Kinesis data delivery stream which archives it in an Amazon S3 bucket.

The solution also creates an Amazon Cognito user pool, an Amazon CloudFront distribution, and a real-time dashboard hosted in an Amazon S3 bucket to securely read and display the device activity stored in the DynamoDB table.

Solution Components

Amazon Kinesis Data Analytics Application

This solution includes an Amazon Kinesis Data Analytics application with SQL statements that compute metrics for the built-in dashboard. The application reads records from the Amazon Kinesis Data Firehose delivery stream and runs the SQL queries to emit specific IoT device metrics including the number of unique connected devices; the average, minimum, and maximum amount of time a device has been connected; the minimum, maximum, and total number of data points sent per device over one-minute time windows; the number of unique devices that are not connected; the average, minimum, and maximum amount of time a device has been disconnected; and anomaly detection, which are stored in Amazon DynamoDB. For more information, see Appendix A (p. 12).

Amazon DynamoDB

The Real-Time IoT Device Monitoring solution creates an Amazon DynamoDB table: `AnalyticsTable` that stores the following information on metrics computed by the Kinesis data analytics application:

- **MetricType:** The name of the computed metric
- **EventTime:** The time the event was generated
- **ConcurrencyToken:** The token used in the event of updates for optimistic locking
- **Data:** The metric data, in JSON format

Device Monitoring Dashboard

The solution features a simple dashboard that loads data from Amazon DynamoDB into line charts every 10 seconds and bar charts every minute. The dashboard leverages Amazon Cognito for user authentication and is powered by web assets hosted in an Amazon Simple Storage Service (Amazon S3) bucket. Amazon CloudFront is used to restrict access to the solution's website bucket contents.

The dashboard uses the open-source `chart.js` JavaScript library to draw charts using HTML5. The `index.html` file contains the HTML elements that render the charts in the dashboard. The `dash.js` file in the `js` folder contains the JavaScript that populates the dashboard with metrics. The Kinesis data application contains the SQL queries that compute metrics. For more information, see Appendix A (p. 12).

After you successfully launch the solution, you will receive an email with instructions for logging into the dashboard.

The dashboard can also be customized to include additional metrics. For more information, see Appendix B (p. 14).

Figure 2: Real-Time IoT Device Monitoring with Kinesis Data Analytics dashboard

Anomaly Detection

The Real-Time IoT Device Monitoring with Kinesis Data Analytics solution leverages the built-in anomaly detection of Amazon Kinesis. When an anomaly score hits the threshold, the events are displayed in the solution's anomaly detection graph. For example, the solution might normally record temperatures that range from 18°C to 24°C. If the solution then records a temperature of 37°C, the solution will detect the anomaly and display it in the graph.

Design Considerations

Solution Updates

Real-Time IoT Device Monitoring with Kinesis Data Anlytics version 1.1.2 uses the most up-to-date Node.js runtime. Version 1.0 uses the Node.js 8.10 runtime, which reaches end-of-life on December 31, 2019. In January, AWS Lambda will block the create operation and, in February, Lambda will block the update operation. For more information, see Runtime Support Policy in the *AWS Lambda Developer Guide*.

To continue using this solution with the latest features and improvements, you can update the stack.

Regional Deployment

This solution uses the Amazon Kinesis Data Firehose and Amazon Kinesis Data Analytics services, which are currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where these services are available. For the most current service availability by region, see AWS service offerings by region.

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the Real-Time IoT Device Monitoring with Kinesis Data Analytics solution. It includes the following AWS CloudFormation template, which you can download before deployment:

real-time-iot-device-monitoring-with-kinesis.template: Use this template to launch the solution and all associated components. The default configuration deploys an AWS IoT rule, two Amazon Kinesis Data Firehose delivery streams, three Amazon Simple Storage Service (Amazon S3) buckets, a Kinesis data analytics application, an AWS Lambda function, an Amazon DynamoDB table, an Amazon Cognito user pool, an Amazon CloudFront distribution, and a real-time dashboard, but you can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy Real-Time Iot Device Monitoring with Kinesis Data Analytics into your account.

Time to deploy: Approximately 10 minutes

Launch the Stack

This automated AWS CloudFormation template deploys the Real-Time Iot Device Monitoring with Kinesis Data Analytics solution. Please make sure that you've verified that you have reviewed the considerations before launching the stack.

Note
You are responsible for the cost of the AWS services used while running this solution. See the Cost (p. 2) section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button below to launch the `real-time-iot-device-monitoring-with-kinesis` AWS CloudFormation template.

   Launch Solution

   You can also download the template as a starting point for your own implementation.
2. The template is launched in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the region selector in the console navigation bar.

   Note
   This solution uses the Amazon Kinesis Data Firehose and Amazon Kinesis Data Analytics services, which are currently available in specific AWS Regions only. Therefore, you must launch this solution an AWS Region where these services are available. For the most current service availability by region, see AWS service offerings by region.
3. On the **Select Template** page, verify that you selected the correct template and choose **Next**.
4. On the **Specify Details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template, and modify them as necessary.This solution uses the following default values.

Parameter	Default	Description
User Name	*<Requires input>*	User name to access the real-time dashboard
User Email Address	*<Requires input>*	Email address of dashboard user. After launch, an email will be sent to this address with dashboard login instructions.

Parameter	Default	Description
IoT topic to monitor	`iot_device-analytics`	Name of the IoT topic where your devices will send messages

6. Choose **Next**.
7. On the **Options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should see a status of **CREATE_COMPLETE** in approximately 10 minutes.

   The solution sends an email invitation to join the real-time dashboard.
10. In the email, follow the instructions to sign in to the dashboard.

Note
In addition to the primary AWS Lambda function `UpdateDDBLambda`, this solution includes the `CustomResourceHelper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.
After launching this solution, you will see these Lambda functions in the AWS console, but only the `UpdateDDBLambda` function is regularly active. However, do not delete the `CustomResourceHelper` function as it is necessary to manage associated resources.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

Encryption

By default, the Amazon Simple Storage Service (Amazon S3) buckets this solution creates are encrypted. The Amazon Kinesis Data Firehose delivery streams are not encrypted. For end-to-end encryption, we recommend restricting access to the solution's delivery streams. For more information, see Controlling Access with Amazon Kinesis Data Firehose.

Amazon CloudFront

This solution deploys a static website hosted in an Amazon S3 bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a special CloudFront user that helps restrict access to the solution's website bucket contents. For more information, see Restricting Access to Amazon S3 Content by Using an Origin Access Identity.

Additional Resources

AWS services

- Amazon Kinesis Data Firehose
- Amazon Kinesis Data Analytics
- AWS Lambda
- Amazon DynamoDB
- Amazon Cognito
- Amazon Simple Storage Service
- Amazon CloudWatch
- AWS CloudFormation
- Amazon CloudFront

Appendix A: Code Components

The Real-Time IoT Device Monitoring with Kinesis Data Analytics solution uses four main code components to process and display metrics on the real-time dashboard. The Amazon Kinesis Data Analytics application (`KinesisAnalyticsApp`) runs SQL queries against the in-application streams and emits the results. An AWS Lambda function (`UpdateDDBLambda`) sends processed data to an Amazon DynamoDB table to be stored. A JavaScript file (`dash.js`) populates the chart with the results of the queries, and an HTML file (`index.html`) renders the chart on the dashboard in real-time.

SQL Query

This SQL query calculates the maximum data point (temperature) per connected device in one-minute intervals. The result is stored in an output in-application stream (`FAN_OUT_STREAM`) with the name of the metric (`PerDeviceMaxTemp`) and the corresponding values.

```
CREATE OR REPLACE PUMP per_device_max_pump AS
INSERT INTO FAN_OUT_STREAM
SELECT STREAM
  STEP(source_sql_stream_001."COL_time" BY INTERVAL '1' MINUTE) AS eventTimeStamp,
  'PerDeviceMaxTemp',
  "device",
  0,
  'Maximum',
  MAX("temp") AS max_value
FROM source_sql_stream_001
GROUP BY "device", STEP(source_sql_stream_001.rowtime BY INTERVAL '1' MINUTE),
 STEP(source_sql_stream_001."COL_time" BY INTERVAL '1' MINUTE);
```

Lambda

The `UpdateDDBLambda` function sends the processed `PerDeviceMaxTemp` data from the Kinesis data analytics application in real-time to a DynamoDB table to be stored.

```
type_operator_map = {
    'ConnectedDevicesCount' : max,
    'PerDeviceMaxTemp' : max,
    'PerDeviceMinTemp': min,
    'PerDeviceAvgTemp': avg,
    'DeviceTempAnomalyScore': max,
    'AvgTempValue': avg,
    'MinTempValue': min,
    'MaxTempValue': max,
    'MaxDisconnTime': max,
    'MinDisconnTime': min,
    'AvgDisconnTime': avg,
    'MaxConnTime': max,
    'MinConnTime': min,
    'AvgConnTime': avg
}
```

JavaScript

The JavaScript (in the `dash.js` file) populates the chart with the maximum data point (temperature) per connected device.

```
var PerDeviceMaxTempParams = retrieveParams("PerDeviceMaxTemp", maxTempPerDeviceQueryTime);

docClient.query(PerDeviceMaxTempParams, function(err, data) {
      if (err) console.log(err);
      else {
        maxTempPerDeviceQueryTime = updateHorizontalBarChart(data, 20,
 maxTempPerDeviceChart, maxTempPerDeviceQueryTime, splitFunc);
      }
    });
```

HTML Element

The HTML element (in the `index.html` file) renders the maximum temperature per device chart with the results of the SQL query.

```
<div class="row aws-mb-l">

    <div class="col-md-5 col-md-offset-1 col-xs-12">

        <div class="x_title">
            <h3>Min Temperature per Device</h3>

        </div>
        <div class="x_content">
            <canvas id="minTempCanvas"></canvas>
        </div>
    </div>
    <div class="col-md-5 col-xs-12">
        <div class="x_title">
            <h3>Max Temperature per Device</h3>
        </div>
        <div class="x_content">
            <canvas id="maxTempCanvas"></canvas>

        </div>
    </div>
</div>
```

Appendix B: Customizing the Dashboard

The Real-Time IoT Device Monitoring with Kinesis Data Analytics solution dashboard displays a default set of metrics, but you can customize the dashboard to include any metrics from IoT devices. Follow the step-by-step instructions in this section to add a metric that calculates maximum air pressure.

Step 1. Update the Demo Script to Send New Data

In the `send-messages.sh` script, update the JSON payload to send a pressure value as an integer. For this exercise, add the bold JSON to the script.

```
aws iot-data publish --topic "$TOPIC" --payload "{\"id\":\"1\",\"device\":\"$DEVICE\",
\"flow\":$FLOW,\"temp\":$TEMP,\"humidity\":$HUMIDITY,\"sound\":$SOUND, \"pressure\": 100}"
 --profile "$PROFILE" --region "$REGION"
```

Step 2. Add the Metric to the Source Schema

Use this procedure to update the source schema with the new metric. For information on using the Schema Editor, see Working with the Schema Editor in the Amazon Kinesis Data Analytics Developer Guide.

> **Note**
> If the custom metric is already added to the Amazon Kinesis Data Analytics application's in-application input stream, skip to Step 2 (p. 14).

1. Sign in to the AWS Management Console and open the Amazon Kinesis Data Analytics console.
2. Select the **KinesisAnalyticsApp** application from the list.
3. Under **Real-Time Analytics**, choose **Go to SQL results**.
4. On the **Source data** tab, choose **Actions**.
5. In the dropdown menu, choose **Edit schema**.
6. Choose **+ Add column** and enter the following:
   - For **Column name**, enter `pressure`.
   - For **Column type**, enter `integer`.
   - For **Row path**, enter `$.pressure`.
7. Choose **Save schema and update stream samples**.
8. To verify that you added the metric correctly, choose **Go to SQL results** and verify that the **Source data** tab shows the new column (`pressure`) and an applicable value.

Step 3. Modify the Application's SQL Code

Use this procedure to update the application's code with the new SQL statement. For information on using the SQL Editor, see Working with the SQL Editor in the Amazon Kinesis Data Analytics Developer Guide.

1. On the Kinesis Data Analytics application's **SQL Editor** page, select the **Real-time analytics** tab.
2. Add the bold SQL statement

```
CREATE OR REPLACE PUMP connected_device_pump AS INSERT INTO FAN_OUT_STREAM
SELECT current_timestamp as eventTimeStamp, 'ConnectedDevicesCount', 'None', 0,
 'Count', * FROM (
    SELECT STREAM * FROM TABLE(COUNT_DISTINCT_ITEMS_TUMBLING(
        CURSOR(SELECT STREAM * FROM source_sql_stream_001),
        'device',
        60
        )
    )
);

CREATE OR REPLACE PUMP per_device_max_pump AS INSERT INTO FAN_OUT_STREAM
SELECT STREAM
STEP(source_sql_stream_001."COL_time" BY INTERVAL '1' MINUTE) AS eventTimeStamp,
'PerDeviceMaxPressure',
"device",
0,
'Maximum',
MAX("pressure") AS max_value
FROM source_sql_stream_001
GROUP BY "device", STEP(source_sql_stream_001.rowtime BY INTERVAL '1' MINUTE),
 STEP(source_sql_stream_001."COL_time" BY INTERVAL '1' MINUTE);
```

   This SQL statement creates a new metric (`PerDeviceMaxPressure`) that stores maximum air pressure readings in the Amazon DynamoDB table.
3. Select **Save and run SQL**.

Step 4. Update the Lambda Code

Add the bold code to the `UpdateDDBLambda` AWS Lambda function.

```
type_operator_map = {
    'ConnectedDevicesCount' : max,
    'PerDeviceMaxTemp' : max,
    'PerDeviceMinTemp': min,
    'PerDeviceAvgTemp': avg,
    'PerDeviceMaxPressure': max,
    'DeviceTempAnomalyScore': max,
    'AvgTempValue': avg,
    'MinTempValue': min,
    'MaxTempValue': max,
    'MaxDisconnTime': max,
    'MinDisconnTime': min,
    'AvgDisconnTime': avg,
```

```
    'MaxConnTime': max,
    'MinConnTime': min,
    'AvgConnTime': avg
}
```

Step 5. Update the JavaScript Code

The solution creates an Amazon Simple Storage Service (Amazon S3) bucket (`websitebucket`) with a `js` folder that contains a `dash.js` file with the JavaScript code that populates the charts with metrics. To populate the new chart with metrics, download the `dash.js` file and follow the step-by-step instructions to modify the JavaScript.

1. Declare the variables and parameters. For this exercise, add the bold JavaScript to line 267 of the `dash.js` file.

```
var maxTempPerDeviceQueryTime = new Date(currentTime.getTime() -
 600000).toISOString().replace('T',' ').replace('Z','');

var maxTempPerDeviceChart = generateHorizontalBarChart("maxTempCanvas", "Max Temp per
 device");

var maxPressurePerDeviceQueryTime = new Date(currentTime.getTime() -
 600000).toISOString().replace('T',' ').replace('Z','');
var maxPressurePerDeviceChart = generateHorizontalBarChart("maxPressureCanvas", "Max
 Pressure per device");
```

2. Modify the `getLatestRecord` function. Add the bold JavaScript to the function.

```
var PerDeviceMinTempParams = retrieveParams("PerDeviceMinTemp",
 minTempPerDeviceQueryTime);
var PerDeviceMaxPressureParams = retrieveParams("PerDeviceMaxPressure",
 maxPressurePerDeviceQueryTime);
var AvgTempParams = retrieveParams("AvgTempValue", AvgTempValueQueryTime);

…

docClient.query(PerDeviceMaxTempParams, function(err, data) {
      if (err) console.log(err);
      else {
        maxTempPerDeviceQueryTime = updateHorizontalBarChart(data, 20,
 maxTempPerDeviceChart, maxTempPerDeviceQueryTime, splitFunc);
      }
    });

docClient.query(PerDeviceMaxPressureParams, function(err, data) {
      if (err) console.log(err);
      else {
        maxPressurePerDeviceQueryTime = updateHorizontalBarChart(data, 20,
 maxPressurePerDeviceChart, maxPressurePerDeviceQueryTime, splitFunc);
      }
    });
```

3. Upload the modified `dash.js` file to the solution's Amazon S3 bucket.

Step 6. Update the Website Assets

In the Amazon S3 bucket with the JavaScript, there is a file (`index.html`) that contains all the HTML elements that render charts on the dashboard. To add a new chart, download the `index.html` file, modify the HTML, and upload the modified file to the Amazon S3 bucket. You can replace the row of an existing chart with the new row, or add the new row to the end of the file.

For this exercise, add the following HTML element to line 287 of the `index.html` file.

```
<div class="row aws-mb-l">
    <div class="col-md-5 col-md-offset-1 col-xs-12">
        <div class="x_title">
            <h3>Max Pressure per Device</h3>
        </div>
        <div class="x_content">
            <canvas id="maxPressureCanvas"></canvas>
        </div>
    </div>
</div>
```

After you upload the modified `index.html` file to the Amazon S3 bucket, open the dashboard in a browser and verify the new chart shows metrics.

Appendix C: Collection of Operational Metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution to improve the services and the products that we offer. When enabled, the following information is collected and sent to AWS each time the dashboard is viewed:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Dashboard Views:** The number of times the dashboard is viewed
- **Processed Metrics:** The number of times metrics are updated

Note that AWS will own the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, complete one of the following tasks:

a) Modify the AWS CloudFormation template mapping section as follows:

```
Solution:
   Data:
      SendAnonymousUsageData: "True"
```

to

```
Solution:
   Data:
      SendAnonymousUsageData: "False"
```

OR

b) After the solution has been launched, find the `CustomResourceHelper` in the Lambda console and set the **SEND_ANONYMOUS_DATA** environment variable to `False`.

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
May 2018	Initial release
December 2018	Added information about the Amazon CloudFront distribution for the static website hosted in the Amazon S3 bucket
August 2019	Upgraded the solution's AWS Lambda functions to the latest Node.js runtime.
December 2019	Added information on support for Node.js update

Notices