IoT Device Simulator

AWS Implementation Guide

Sean Senior

May 2018

Contents

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the IoT Device Simulator on the Amazon Web Services (AWS) Cloud. It includes a link to a 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, administrators, and DevOps professionals who have practical experience with IoT devices, and the AWS Cloud.

Overview

Amazon Web Services (AWS) provides many services to help customers build serverless IoT applications that gather, process, analyze, and act on connected device data, without having to manage any infrastructure. With AWS, customers also build a secure, agile, and scalable backend for their IoT applications. This eliminates the need for customers to develop and manage their own backend resources and can help reduce costs and increase productivity and innovation. But, it can be a challenge to test IoT applications and backend services without a large pool of physical, connected devices.

To help customers more easily test device integration and IoT backend services, AWS offers the IoT Device Simulator solution. This solution provides a web-based graphical user interface (GUI) console that enables customers to create and simulate hundreds of connected devices, without having to configure and manage physical devices, or develop time-consuming scripts. This solution is designed to work out-of-the-box, or you can use this

solution as a reference implementation to build a custom simulation engine for your specific use case.

IoT Device Simulator provides a console that enables users to build a large fleet of virtual connected devices (widgets) from a user-defined template and simulate those widgets publishing data at regular intervals to AWS IoT. You can also monitor individual widgets from the simulator or observe how backend services are processing the data.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost for running this solution depends on the amount of data being simulated, collected, stored, processed, and presented. 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.

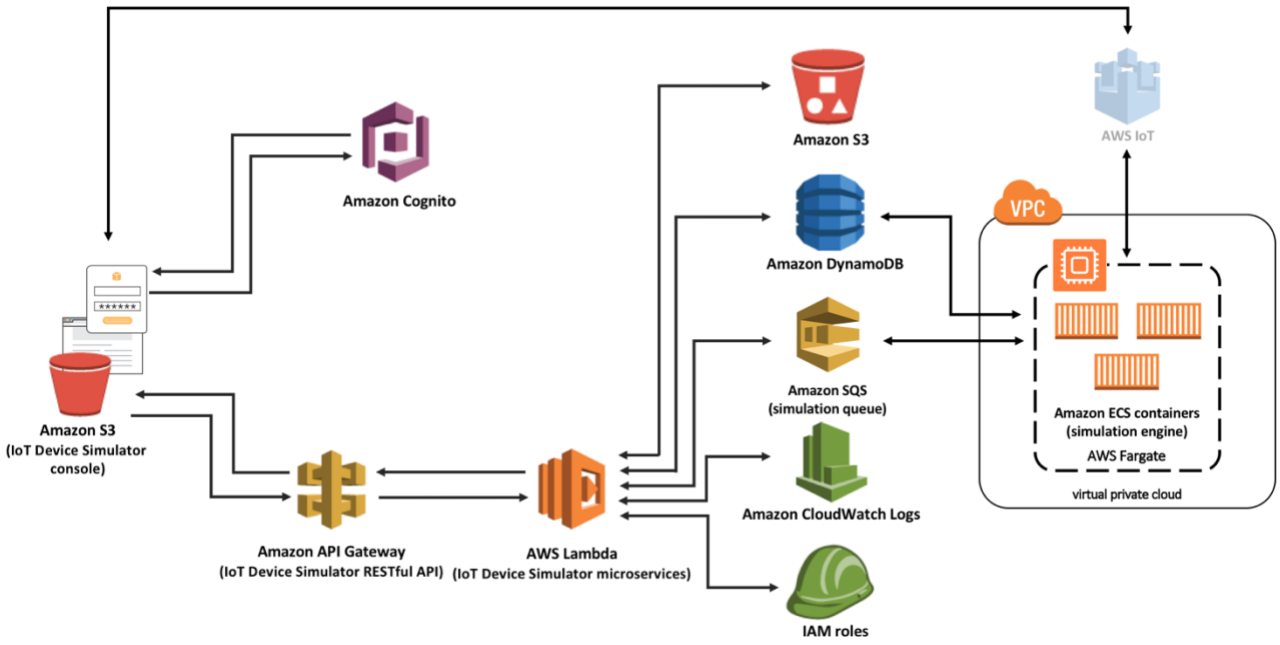

Figure 1: IoT Device Simulator architecture

The AWS CloudFormation template deploys a device simulator API, which leverages Amazon API Gateway to invoke the solution's microservices (AWS Lambda functions). These microservices provide the business logic to perform CRUD operations on virtual devices and device types, record simulation metrics, and perform administration tasks. These microservices interact with Amazon Simple Storage Service (Amazon S3), Amazon

DynamoDB, AWS Identity and Access Management (IAM), and Amazon CloudWatch Logs to provide data storage, management, and logging functions.

The solution also deploys an Amazon Virtual Private Cloud (Amazon VPC) network topology with two public subnets and two private subnets that contains the solution's simulation engine, which runs in Amazon Elastic Container Service (Amazon ECS) containers provisioned by AWS Fargate. The VPC also includes a NAT gateway.

The solution creates a web console and deploys it into an Amazon S3 bucket configured for static website hosting. During initial configuration, the solution also creates a default administrator role and sends an access invite to a customer-specified user email. The solution uses an Amazon Cognito user pool to manage user access to the console and the device simulator API.

When the device simulator API receives an authorized request, Amazon API Gateway invokes the appropriate Lambda function. The Lambda function returns the execution results to the API, which returns the results to the simulator console.

When a device simulation request is received, the device microservice sends the request to a simulation queue in Amazon Simple Queue Service (Amazon SQS). The simulation engine polls the simulation queue for simulation start and stop requests.

When a start simulation request is received, the solution will spawn a virtual device based on the request and start publishing simulated data to the defined AWS IoT endpoint for the duration defined in the device type definition. Each simulation runs until the defined execution duration expires or a stop request is received. When a stop simulation request is received, the solution will terminate the simulation based on the request and update the device catalog.

Solution Components

Device Simulator Microservices

The IoT Device Simulator microservices are a series of AWS Lambda functions that provide the business logic and data access layer for all device simulation operations. Each Lambda function assumes an AWS Identity and Access Management (IAM) role with least privilege access (minimum permissions necessary) to perform its designated functions.

Admin Microservice

The `iot-sim-admin-service` Lambda function processes device simulator API requests sent to the `/admin/*` endpoints. All `/admin/*` API endpoints are configured as Lambda

proxy endpoints that pass the full request payload to the `iot-sim-admin-service` function. The admin microservice handles all administrative services, including users and general settings.

Metrics Microservice

The `iot-sim-metrics-service` Lambda function processes device simulator API requests sent to the `/metrics/*` endpoints. All `/metrics/*` API endpoints are configured as Lambda proxy endpoints that pass the full request payload to the `iot-sim-metrics-service` function. The metrics microservice handles all metrics operations for the IoT Device Simulator.

Device Microservice

The `iot-sim-device-service` Lambda function processes device simulator API requests sent to the `/device/*` endpoints. All `/device/*` API endpoints are configured as Lambda proxy endpoints that pass the full request payload to the `iot-sim-device-service` function. The device microservice handles all device and device type operations, including list, add device, remove device, list device types, add device type, update device type, start simulation, and stop simulation.

Profile Microservice

The `iot-sim-profile-service` Lambda function processes device simulator API requests sent to the `/profile/*` endpoints. All `/profile/*` API endpoints are configured as Lambda proxy endpoints that pass the full request payload to the `iot-sim-profile-service` function. The profile microservice handles reading user profile information for the IoT Device Simulator.

Amazon DynamoDB

The IoT Device Simulator uses Amazon DynamoDB to persist metadata for devices, device types, settings, and metrics. The solution creates four DynamoDB tables: `iot-sim-device-widgets`, `iot-sim-device-types`, `iot-sim-metrics`, and `iot-sim-settings`.

The `iot-sim-device-widgets` table stores the following information on virtual devices:

- **Category:** The category of the device widget. This is defined by the device type.
- **CreatedAt:** The date and time (in UTC) when the device widget was created
- **EndedAt:** The date and time (in UTC) when the last simulation run ended
- **Id:** The unique identifier for the device widget. This is generated automatically.
- **Metadata:** The metadata associated with the device widget

- **Runs:** The number of simulation runs performed by the device widget
- **Stage:** The current simulation stage of the device widget. For more information, see Simulation Stage.
- **StartedAt:** The date and time (in UTC) when the last simulation run started
- **Subcategory:** The sub category of the device widget. This is defined by the device type.
- **TypeId:** The device type identifier
- **UpdatedAt:** The date and time (in UTC) when the device widget was updated
- **UserId:** The user id of the owner of the device widget


```
{
  "category": "custom widget",
  "createdAt": "2018-04-09T20:01:58Z",
  "endedAt": "2018-04-09T20:04:13Z",
  "id": "B1OJ6UHtoG",
  "metadata": {},
  "runs": 1,
  "stage": "sleeping",
  "startedAt": "2018-04-09T20:02:10Z",
  "subCategory": "weather station",
  "typeId": "rkbZDGsOf",
  "updatedAt": "2018-04-09T20:04:13Z",
  "userId": "sample_user"
}
```


Figure 2: Sample device record

The `iot-sim-device-types` table stores the following information on device types:

- **CreatedAt:** The date and time (in UTC) when the device type was created
- **Custom:** Identifies whether the device type is user created (true) or system generated (false)
- **Name:** The name of the device widget
- **Spec:** The specification of the device type
- **Duration:** The duration a simulation will run for a particular device type
- **Interval:** The interval at which devices will publish data to AWS IoT
- **Payload:** The JSON definition of the attributes to be simulated
- **Topic:** The AWS IoT topic where the device will publish data
- **TypeId:** The device type identifier
- **UpdatedAt:** The date and time (in UTC) when the device type was updated

- **UserId:** The user id of the owner of the device type


```
{
  "createdAt": "2018-03-05T19:14:00Z",
  "custom": true,
  "name": "weather station",
  "spec": {
    "duration": "120000",
    "interval": 2000,
    "payload": [
      {
        "_id_": "HJpJGapuf",
        "name": "stationid",
        "smax": 20,
        "smin": 10,
        "static": true,
        "type": "string"
      },
      {
        "_id_": "H1ea1fTTuM",
        "dmax": 99,
        "dmin": 0,
        "imax": 200,
        "imin": 0,
        "name": "temperature",
        "precision": 2,
        "type": "float"
      },
      {
        "_id_": "H1f61z6aOz",
        "max": 100,
        "min": 0,
        "name": "humidity",
        "type": "int"
      },
      {
        "_id_": "HkQa1G6p_M",
        "name": "timestamp",
        "tsformat": "default",
        "type": "timestamp"
      },
      {
        "_id_": "r1IfpsQ3M",
        "arr": [
          "running",
          "stopped",
          "starting",
          "error",
          "warning"
        ],
        "name": "state",
        "type": "pickOne"
```

```
        }
      ],
      "topic": "/weather/data"
    },
    "typeId": "rkbZDGsOf",
    "updatedAt": "2018-04-17T17:21:33Z",
    "userId": "sample_user"
  }
```


Figure 3: Sample device type record

The `iot-sim-metrics` table stores the following logging information:

- **CreatedAt:** The date and time (in UTC) when the metric record was created
- **DeviceBreakdown:** A breakdown of the total number of simulation runs per device type by a specific user for the current month
- **Id:** The unique identifier for the device widget. This is generated automatically.
- **Simulations:** The number of simulation runs broken down by category
- **MonthlyRuns:** A breakdown of the last six months of simulation runs
- **TotalDuration:** The total number of simulation minutes by a user
- **TotalRuns:** The total number of simulation runs by a user
- **UpdatedAt:** The date and time (in UTC) when the metric record was updated
- **UserId:** The user id of the simulation user


```
{
  "createdAt": "2018-03-08T16:47:30Z",
  "deviceBreakdown": {
    "id": "201804",
    "simulations": [
      {
        "category": "weather station",
        "runs": 57
      },
      {
        "category": "vehicle",
        "runs": 2
      },
      {
        "category": "Sub Sensor",
        "runs": 5
      },
      {
        "category": "torque sensor",
        "runs": 25
      }
```

```
      ]
    },
    "monthlyRuns": [
      {
        "auto": 242,
        "generic": 164,
        "id": "201803",
        "month": "Mar",
        "runs": 147
      },
      {
        "auto": 0,
        "generic": 0,
        "id": "201802",
        "month": "Feb",
        "runs": 0
      },
      {
        "auto": 59,
        "generic": 8830,
        "id": "201804",
        "month": "Apr",
        "runs": 89
      }
    ],
    "totalDuration": 9295,
    "totalRuns": 236,
    "updatedAt": "2018-04-17T17:29:47Z",
    "userId": "sample_user"
}
```


Figure 4: Sample metrics record

The `iot-sim-settings` table stores the following settings information:

- **SettingID:** The unique identifier of the setting record
- **CreatedAt:** The date and time (in UTC) when the setting was created
- **Setting:** The JSON description of the configuration setting details
- **Type:** The type of setting record (config)
- **UpdatedAt:** The date and time (in UTC) when the setting was updated

Device Simulation

The IoT Device Simulator leverages Amazon Simple Queue Service (Amazon SQS), Amazon Elastic Container Service (Amazon ECS), and AWS Fargate to simulate virtual devices sending data to AWS IoT endpoints.

Users make simulation requests via the included web console. Simulation requests are added to the Amazon SQS queue where they are stored until they are processed.

Amazon ECS containers provisioned by AWS Fargate contain a simulation engine that periodically polls the simulation queue for simulation requests. The simulation engine launches a virtual device and starts the device publishing simulated data to the AWS IoT endpoint. After the specified duration, the simulation engine stops the simulation, terminates the virtual device, and updates the device state and metrics in the `iot-sim-device-widgets` table.

Device Types

Device types are used to define the type of data your simulated IoT devices will send. When you create a device type, you define the structure of the data and specific attributes for each item in the payload. Device types contain the follow properties:

- **Name:** The name of the device type
- **Data topic:** The topic where the data will be sent in AWS IoT
- **Data transmission duration:** The duration, in milliseconds, your device will send data
- **Data transmission interval:** The interval, in milliseconds, at which your device will send data

Attributes

Attributes define what each payload contains. Attributes include the follow fields:

- **Attribute name:** The name of the attribute
- **Attribute data type:** The data type of the payload. The following data types are available:
  - **Boolean:** Sends a random true or false value based on seeding. You set a minimum value, a maximum value, and seed value.
  - **Float:** Sends a random decimal value. You set the precision of the value, a minimum value, a maximum value, a decimal precision minimum value, and a decimal precision maximum value.
  - **Integer:** Sends a random integer value. You set a minimum value and a maximum value.
  - **String:** Sends a random string value. You set a minimum length and a maximum length of the string.

  - **Unique Identifier:** Sends a random UUID value
  - **Unique Short Identifier:** Sends a random short UUID value
  - **UTC Timestamp:** Sends a random UTC timestamp in YYYY-MM-DDTHH:mm:ss. For example, 2018-04-10T12:21:22. You can also specify a timestamp in Unix time format. For example, 1523377324.
  - **Location:** Sends a random latitude and longitude coordinated within a radius of a defined point. You specify a latitude for the center position, a longitude for the center position, and a radius from the center position.
  - **Pick One from Array:** Sends a random string value from a user-defined, comma-separated list of strings.
- **Static:** Choose whether the value stays the same for the duration of the simulation
- **Default Value:** A fixed value that will always be sent in the payload

Widgets

Widgets are virtual IoT devices that send simulated data to AWS IoT endpoints. With the IoT Device Simulator, you can create up to 25 widgets at a time. For example, to launch 100 widgets, you can create four batches of 25 widgets. The solution allows you to run up to 1,000 simultaneous simulations across the simulation engine. If you request more than 1,000 simulations at a given time, they will be queued and executed when the number of current simulations is less than 1,000.

Simulation Stage

Device widgets can be in one of three stages:

- **Simulating:** The device widget is sending simulated data to the IoT endpoint
- **Provisioning:** The device widget is launching
- **Sleeping:** The device widget is stopped and not sending simulated data

You can view each widget's simulation stage as well as a breakdown of the total number of widgets in each stage in the web console's device view.

Logging Level

By default, this solution logs informational messages for the simulation engine. To change the logging level, open the IoT Device Simulator console and navigate to the **Settings** menu option. On the **Simulation Engine** tab, you can change the **Logging Level** to `INFO`, `ROBUST`, or `DEBUG`.

Device Simulator Console

The IoT Device Simulator features a web console that you use to manage simulations. The console displays information about virtual devices and device types, simulation states, and user profiles. You use the console to create and terminate virtual devices, start and stop simulations, and view metrics.

Dashboard

The console includes a dashboard you can use to view how many simulations you have run, how many simulation hours you have run, and a breakdown of the different device types you have created.

Figure 5: Sample dashboard

Device View

The console also includes a device view that shows all devices in your pool of created devices, as well as the simulation state of the devices.

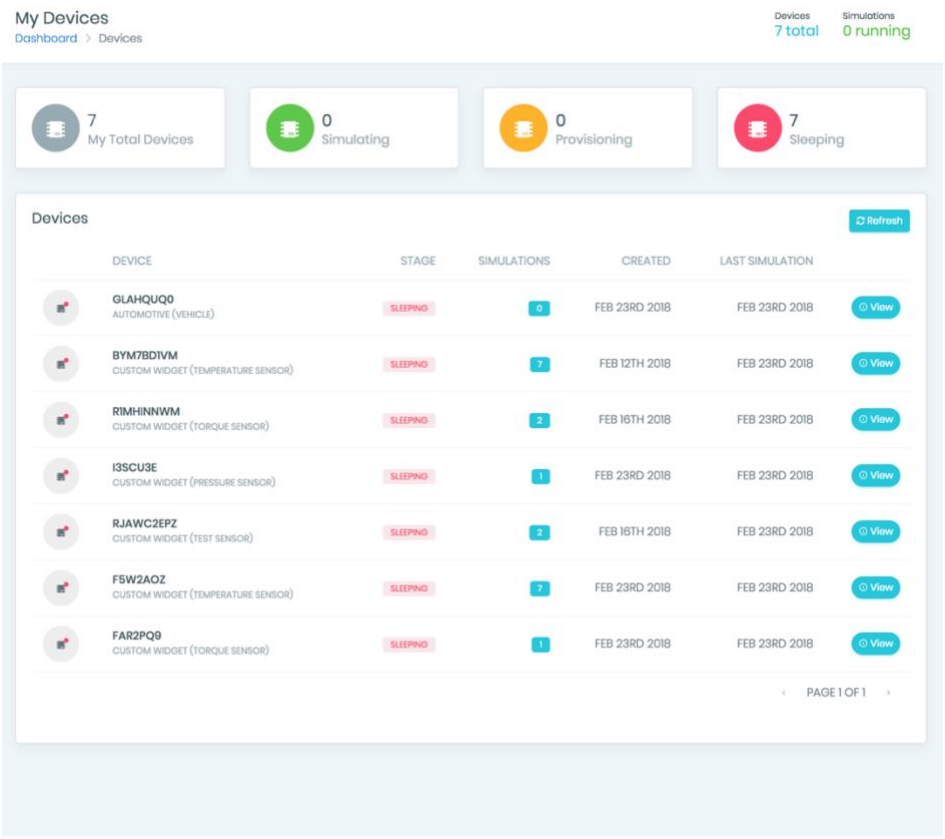

Figure 6: Sample device view

Device Types View

The console includes a device types view that displays a list of device types, and allows you to create new device types.

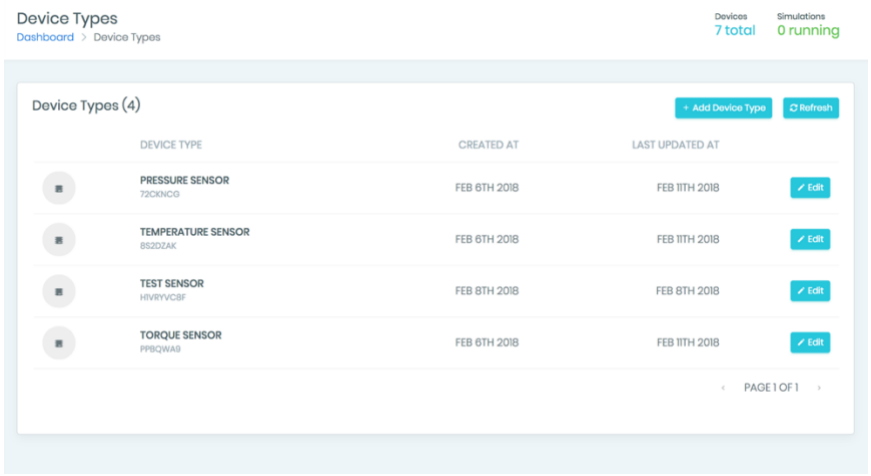

Figure 7: Sample device types view

Custom Widget View

The console includes a custom widget view that displays your list of widgets, and allows you to create and delete widgets, and start and stop simulations. This view also provides a count of the widgets in various simulation stages.

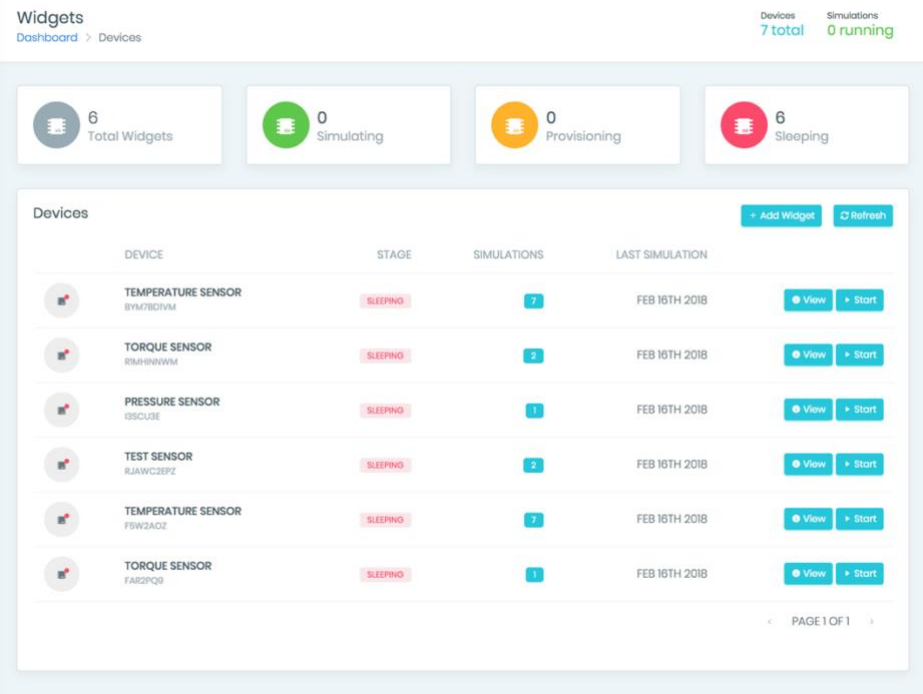

Figure 8: Sample widgets view

Authentication

This solution takes advantage of the user authentication features of Amazon Cognito User Pools. After successfully authenticating a user, Amazon Cognito issues a JSON web token that is used to allow the console to submit requests to the device simulator API. HTTPS requests are sent by the console to the simulator API with the authorization header that includes the token.

User Management

After the device simulator is deployed, administrators can invite privileged users and customize their permissions to implement granular access-control policies. Administrators can also change settings.

Logging and Metrics

The device simulator solution logs API calls, latency, and error rates to Amazon CloudWatch which you can use to set alarms based on defined thresholds. The solution also monitors traffic at the REST API level. Optionally, you can enable detailed metrics for each method of

the device simulator REST API from the Amazon API Gateway deployment configuration console. Detailed metrics will incur an extra cost.

Considerations

Regional Deployments

This solution uses the AWS Fargate service, which is currently available in specific AWS Regions only. Therefore, you must launch this solution in a region where AWS Fargate is available.[1]

AWS CloudFormation Template

This solution uses AWS CloudFormation to automate the deployment of the IoT Device Simulator. It includes the following AWS CloudFormation template, which you can download before deployment:

View template

iot-device-simulator.template: Use this template to launch the solution and all associated components. The default configuration deploys an Amazon Virtual Private Cloud network topology, Amazon API Gateway, AWS Lambda functions, Amazon Simple Storage Service buckets, Amazon DynamoDB tables, AWS Identity and Access Management roles, Amazon CloudWatch Logs, a device simulator graphical user interface, an Amazon Cognito user pool, Amazon Simple Queue Service, Amazon Elastic Container Service, and AWS Fargate. You can also customize the template based on your specific network needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture, configuration, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the IoT Device Simulator into your account.

Time to deploy: Approximately 10 minutes

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

[1] For the most current AWS Fargate availability by region, see https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

Step 1. Launch the Stack

This automated AWS CloudFormation template deploys the device simulator.

> **Note**: You are responsible for the cost of the AWS services used while running this solution. See the Cost 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 to the right to launch the `iot-device-simulator` 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 AWS Fargate service, which is currently available in specific AWS Regions only. Therefore, you must launch this solution a region where AWS Fargate is 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
Administrator Name	*<Requires input>*	The user name for the initial solution administrator. After the solution is deployed, this administrator can create and manage other users, including additional administrators.
Administrator Email	*<Requires input>*	Email address of the administrator user. After launch, an email will be sent to this address with console login instructions.

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 IoT Device Simulator console.

10. In the email, follow the instructions log in to the console.

> **Note:** In addition to the AWS Lambda functions that make up the device simulator microservices, this solution includes the `iot-device-sim-helper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.
>
> When running this solution, the `iot-device-sim-helper` function is inactive. However, do not delete the `iot-device-sim-helper` function as it is necessary to manage associated resources.

Step 2. Define Your Device Types

Use this procedure to define the data each of your simulated IoT devices will send. Create the structure of the data and the specific properties and type for each item in the payload.

1. Sign in to the IoT Device Simulator console and in the navigation pane, choose **Device Types**.

2. Select + **Add Device Types**.

3. On the **Device Type Definition** page, enter the appropriate information. For more information, see Device Types. Note the value you entered for the **Data topic** attribute.

4. To format the payload, select **Add Attribute**.

5. In the **Message Attribute** window, enter the appropriate information. For more information, see Attributes.

6. Select **Submit**.

7. Repeat steps 4-6 for each attribute for your device payload.

8. Select **Save**.

Step 3. Create a Pool of Widgets

Use this procedure to define the number of simulated IoT devices (widgets) you will launch.

1. On the IoT Device Simulator console page, in the navigation menu, select **Widgets**.

2. Select + **Add Widgets**.

3. In the **Create a widget** window, specify a device type and the number of widgets you want to launch. You can create up to 25 widgets at a time.

4. Select **Submit**.

5. To view a specific device, select **View** for the specific widget. This shows metadata about the widget and the messages being received by AWS IoT during the simulation.

Step 4. Test Your IoT Backend Service

Use this procedure to test your IoT backend service.

1. Sign in to the AWS IoT Console and in the left navigation pane, choose **Test**.

2. Subscribe to the applicable topic. You can find the topic in the **Data topic** attribute in your device type definition.

3. View the simulated data flowing into the AWS IoT Core service in your account.

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.

IAM Roles

AWS Identity and Access Management (IAM) roles enable customers to assign granular access policies and permissions to services and users on the AWS Cloud. The IoT Device Simulator creates several IAM roles, including roles that grant the device simulator AWS Lambda functions access the other AWS services used in this solution. These roles are necessary to allow the services to simulate devices in your account.

Additional Resources

AWS services

- AWS Lambda
- AWS Fargate
- Amazon Elastic Container Service
- Amazon DynamoDB
- Amazon Cognito
- Amazon CloudWatch
- Amazon Virtual Private Cloud
- AWS CloudFormation
- Amazon Simple Queue Service
- AWS IoT
- Amazon Simple Storage Service
- Amazon API Gateway
- AWS Identity and Access Management

Appendix A: Automotive Module

The IoT Device Simulator includes a pre-built automotive module that you can use to simulate vehicle telemetry data using pre-defined device types. The automotive module uses a power train simulation model to generate simulated vehicle telemetry data.

Step 1: Add Your Mapbox Token (Optional)

The module leverages the location features of Mapbox to provide a map for your simulated vehicles. To display the map in the automotive module, you must register for a free Mapbox developer account. After you register, add your Mapbox token to the IoT Device Simulator.

1. On the IoT Device Simulator console page, in the navigation menu, select **Settings**.
2. In the **General** tab, enter your Mapbox token.
3. Select **Save**.

> **Note:** The **Mapbox Token** setting is cached when you log in to the device simulator console. If you change to a new token, you must log out of the console and log back in for the change to take effect.

Step 2. Launch Simulated Vehicles

Use the following procedure to launch virtual connected vehicles.

1. On the IoT Device Simulator console page, in the navigation menu, select **Automotive**.
2. Select + **Add Vehicles**.
3. In the **Create vehicles** window, specify the number of vehicles you want to simulate.
4. Select **Submit**.
5. To monitor a specific vehicle, select **View**. This shows metadata about the vehicle, the route it is driving, and the telemetry messages being received by AWS IoT during the simulation.

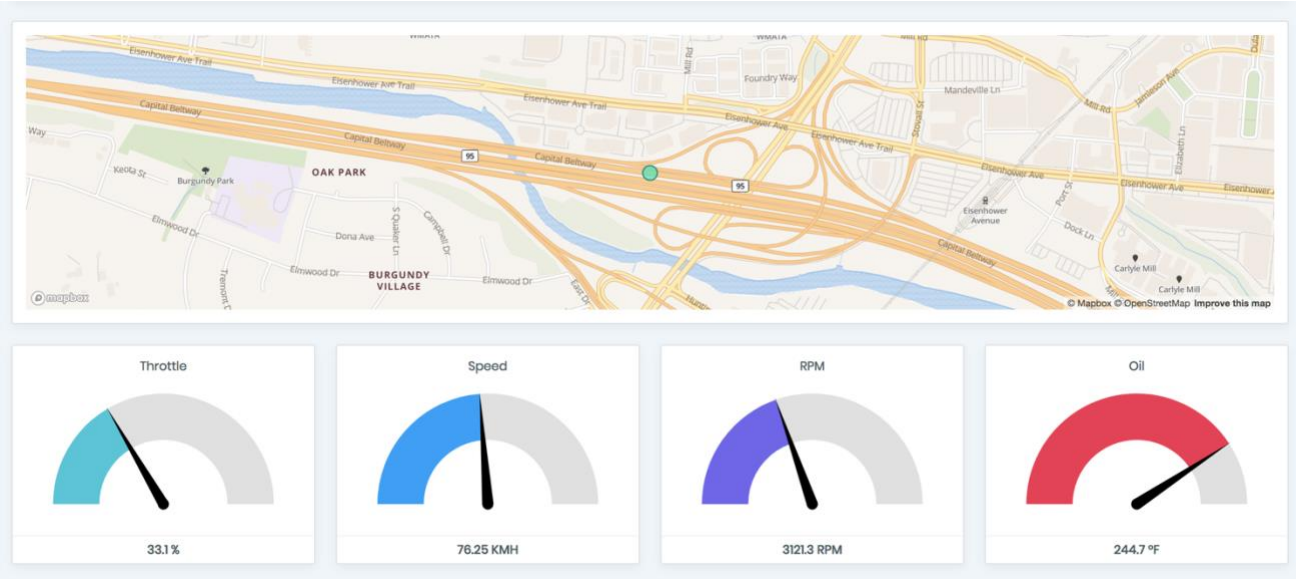

Figure 9: Sample automotive simulation dashboard

While the simulated vehicles this solution creates are designed to be used out-of-the-box, you can configure how the automotive module sends data using the **Configuration** button at the top of the screen.

Appendix B: Collection of Anonymous Data

This solution includes an option to send anonymous usage data to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment
- **Timestamp:** Data-collection timestamp
- **Simulations Run:** The number of simulations started
- **Simulation Duration:** The duration, in minutes, of the simulation
- **Simulation Category:** The device widget category of the simulation

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 task:

Modify the AWS CloudFormation template mapping section as follows:

```
Mappings:
    Send:
        AnonymousUsage:
            Data: "Yes"
```

to

```
Mappings:
    Send:
        AnonymousUsage:
            Data: "No"
```

Send Us Feedback

We welcome your questions and comments. Please post your feedback on the AWS Solutions Discussion Forum.

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	In sections
May 2018	Initial release	--

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The IoT Device Simulator is licensed under the terms of the Amazon Software License available at https://aws.amazon.com/asl/.