Comentario técnico: CTC-104
Componente: **Amazon Web Services**
Autor: Sergio R. Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	31/08/20	

En el CTC-086 charlamos sobre la “Internet de *nuestras* cosas” y propusimos una solución escalable basada en MQTT, a la vez que recomendamos recurrir a los grandes proveedores para una red de envergadura. Analizaremos ahora una parte de la implementación de ese tipo de soluciones en uno de esos grandes proveedores. En esta oportunidad es Amazon, y nos referimos a los Amazon Web Services. En particular, desarrollaremos la utilización de AWS IoT, el conjunto de servicios para la Internet de las cosas, y nos enfocaremos de manera más específica en AWS IoT Core, el producto de conectividad basado en MQTT y HTTP que nos permite concentrar la información proveniente de nuestros dispositivos.

Índice de contenido

Breve descripción de AWS (desde nuestro punto de vista)

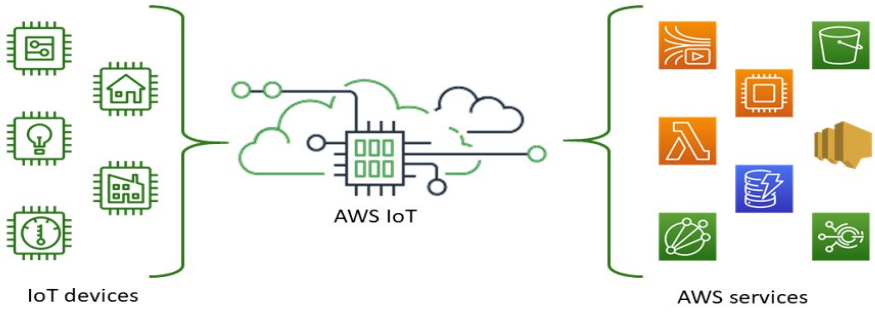
Amazon Web Services es una plataforma de servicios, los cuales se proveen mediante la red de datacenters de Amazon. Se trata de una compleja red de servicios de infraestructura de sistemas, content delivery, gestión,

autenticación, etc. Muchos de esos servicios cuentan con interfaces de modo de poder recibir y entregar sus datos de y a otros servicios, formando así toda una plataforma sobre la que podemos desarrollar nuestro sistema.

Por otra parte, el tipo de sistemas que nos interesa como desarrolladores de productos de hardware y bajo la óptica de la Internet de las cosas (nuestras), es aquél con conectividad. Existe un conjunto de servicios, AWS IoT, que atiende la mayor parte de los requerimientos relacionados a la IoT, desde el ingreso de datos hasta ejecutar acciones en respuesta al contenido de los mismos y graficarlos en una consola. Dentro de este conjunto se encuentra AWS IoT Core. Este servicio permite enviar y recibir mensajes de y a los dispositivos mediante un *message broker* que acepta MQTT y MQTT sobre Websocket, y HTTP. Ingresado el mensaje, un motor de reglas nos permite reenviar la información a otro servicio, como por ejemplo almacenar en tablas o bases de datos, o re-publicarlo nuevamente hacia los dispositivos. Pero la cosa no termina aquí, si bien las necesidades de cada aplicación son particulares, todas las aplicaciones de tipo IoT tienen una serie de necesidades comunes, y en vez de diseñarlas para cada una, AWS IoT Core nos provee una serie de servicios ya desarrollados para que podamos configurar los dispositivos, enviarles instrucciones, administrarlos. Dentro de AWS IoT, a su vez, existen servicios más complejos como para, por ejemplo, realizar manejo de flota de dispositivos.[1]

Sintetizando:

- AWS IoT concentra los servicios relacionados con la IoT.
- AWS IoT Core recibe los datos de telemetría y estado y maneja los dispositivos. Al ser un *message broker*, la comunicación es bidireccional e instantánea.
- AWS IoT Core cuenta con un motor de reglas que permite filtrar, redireccionar, e inyectar esos datos en el resto de la plataforma.
  - A partir de aquí, por ejemplo, podemos acceder a Dynamo DB, ejecutar funciones Lambda, procesar en Kinesis, Quicksight, etc.
- Dentro de AWS IoT, contamos con un servicio que nos permite de manera muy simple y rápida almacenar y visualizar los datos: AWS IoT Analytics.



Una vez dentro de la plataforma, los datos pueden ser procesados por otros productos, analizados, graficados, o lo que necesitemos hacer con ellos con cualquiera de los productos disponibles en la plataforma, de los cuales los expertos de sistemas conocen mucho más de lo que podamos comentar aquí.

Breve descripción de AWS IoT

Como comentamos, AWS IoT es el conjunto de servicios que nos permite operar para responder a las inquietudes típicas de negocio relacionadas con la IoT. No sólo concentrar la información proveniente de nuestros dispositivos y disponer de un modo de manejarlos, sino además procesar y visualizar la información, y responder a ella.

Identificamos algunos de los servicios que lo componen:

1 https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html

- AWS IoT Core: el punto de entrada
- AWS IoT Device Defender: soporte para monitoreo de la seguridad de nuestra red
- AWS IoT Device Management: manejo y aprovisionamiento de dispositivos, a escala
- AWS IoT Analytics: procesamiento y visualización de la información
- AWS IoT Events: análisis y respuesta al contenido de la información
- AWS IoT SiteWise: esquema completo para aplicaciones industriales
- AWS IoT Things Graph: flujo de procesamiento

Breve descripción de AWS IoT Core

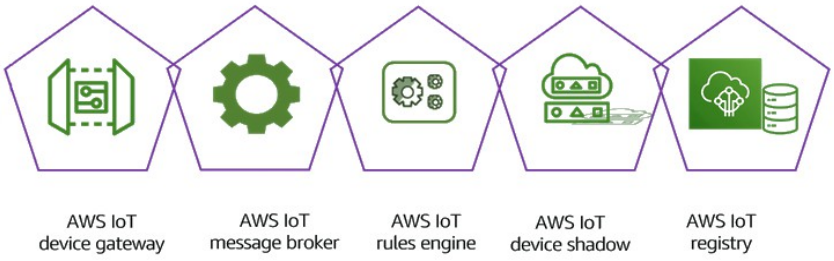
Como comentamos, AWS IoT Core[2] contiene el servicio de conectividad que nos permite concentrar la información proveniente de nuestros dispositivos y disponer de un modo de manejarlos.

Cuenta con un *message broker* que soporta MQTT, MQTT sobre Websocket, y HTTP (sólo para publicar mensajes); aunque en el resto de este texto nos concentraremos solamente en MQTT[3].

Como centro de manejo de dispositivos, nos presenta una base de dispositivos (*registry*), que nos provee herramientas administrativas de gestión de nuestras "cosas" (Things), como se las llama en el entorno.

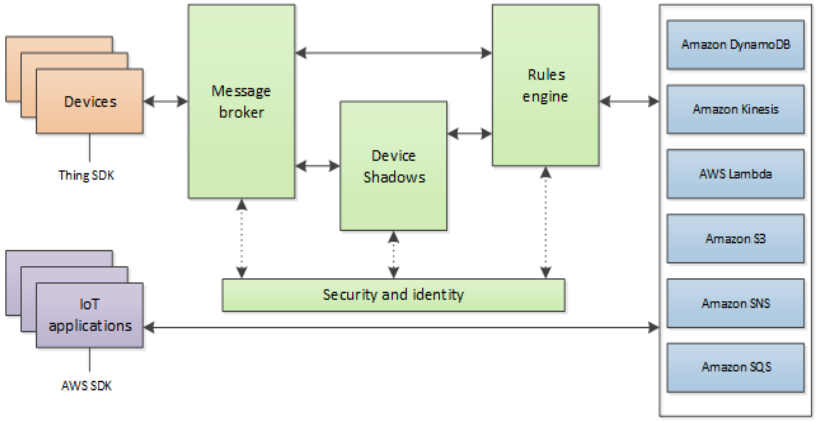
Para mantener el estado y configuración de los dispositivos ante desconexiones, provee un servicio de *device shadow*. El código para operar desde nuestro dispositivo podemos obtenerlo mediante las opciones de SDK disponibles para varias arquitecturas.

Finalmente, un motor de reglas (*rules engine*) nos permite re-publicar el mensaje en otro tópico, filtrarlo y almacenarlo en una tabla, o enviarlo a otro servicio de AWS.



El siguiente diagrama en bloques muestra una interrelación entre los diversos componentes de AWS IoT Core y el resto de la aplicación IoT:

2 https://docs.aws.amazon.com/iot/latest/developerguide/aws-iot-how-it-works.html
3 para quienes deseen conocer más detalles sobre MQTT, remitimos al CTC-087



Message broker

La implementación de MQTT es standard, se trata de un broker con las características esperables. Las diferencias al momento de escribir este documento son:[4]

- No soporta retención de mensajes
- La versión de MQTT requerida es 3.1.1
- No soporta QoS 2
- No garantiza orden en la entrega

Autenticación y seguridad

Se utiliza TLS mediante certificados X.509. Es decir, autenticación por esquema asimétrico (clave pública/privada). Existen otras alternativas para HTTP y MQTT sobre Websocket.

- La conexión es encriptada mediante TLS 1.2
- Server y dispositivo autentican al otro mediante su certificado.

Amazon provee los certificados; los mismos se generan automáticamente al agregar un dispositivo. Algunos servicios de manejo de flota permiten o requieren la utilización de una CA por parte del usuario.
Autenticado el dispositivo, una serie de políticas determinan qué puede hacer en la red. Dado que no existe limitación en los tópicos posibles en el *message broker*, podemos limitar mediante estas políticas, a qué tópicos un dispositivo puede suscribirse y en cuáles puede publicar.

Tópicos

No existe limitación en los tópicos, los mismos son únicos para cada cuenta y corren por cuenta del usuario. Dado que es posible publicar y suscribirse a cualquier tópico, podemos armar nuestra implementación a gusto y por ejemplo comunicar los dispositivos mediante el *message broker* sin necesidades adicionales.
El servicio de *device shadows* emplea tópicos reservados, así como también lo hacen otros servicios de aprovisionamiento y manejo de flota. Los tópicos reservados comienzan con '$', es posible suscribirse y publicar pero no crear nuevos tópicos.

Contenido

A fin de que el motor de reglas pueda realizar su cometido, el contenido de los mensajes debe ser un objeto JSON codificado en UTF-8. Los mensajes de *device shadow* son también objetos JSON, aunque de su procesamiento se encargará el código provisto en el SDK que utilicemos. Algunos servicios emplean además CBOR (Concise Binary Object Notation).
El tamaño máximo de mensaje es de 128KB.

4 https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html

Device shadow

Es un servicio de "gemelo digital" (digital twin), una copia del estado y configuración del dispositivo en formato JSON que reside en la nube y es accesible siempre. Mediante el *message broker* se sincroniza con el dispositivo en ambos sentidos, es decir, el dispositivo publica sus cambios y recibe los que otros dispositivos o aplicaciones realicen sobre su *shadow*. Si el dispositivo se desconecta de la red, y aplicaciones cambian su estado deseado (en la *shadow*), se producirá la sincronización al momento de reconexión. De forma simplificada, cuando se produce un cambio AWS IoT Core registra un timestamp y envía mensajes *delta* a los suscriptores; estos mensajes contienen la diferencia entre el estado deseado (*desired*, el configurado en la *shadow* por una aplicación, por ejemplo) y el reportado (*reported*, el último que el dispositivo indica). Los suscriptores, incluyendo al dispositivo, pueden obrar en consecuencia actualizando por ejemplo el estado, en caso del dispositivo, o una interfaz de usuario remota o virtual, en el caso de una aplicación móvil, por ejemplo.
Las aplicaciones pueden acceder a la *shadow* de un dispositivo mediante una API HTTP de tipo REST, además de los tópicos prefijados en el broker MQTT que usan los dispositivos. La especificación de la operatoria requerida y las formas de acceder, así como los tópicos, se encuentran en la documentación[5].

Rules engine

Se trata de un motor de reglas con una sintaxis similar a SQL que permite disparar una acción en respuesta al contenido de los mensajes que publican los dispositivos. Entre las acciones posibles encontramos:

- enviar a una tabla en Dynamo DB
- almacenar en un bucket S3
- ejecutar una acción POST en un URL HTTPS
- ejecutar una función Lambda
- enviar a AWS IoT Analytics, AWS IoT Events, AWS SiteWise
- enviar a un stream Kinesis Data Firehose
- enviar notificaciones push en Amazon SNS
- republicar en otro tópico

Manejo de datos

No existe aquí el concepto de estados como en GCP, la información correspondiente al estado del dispositivo se sincroniza con la *shadow* y se obtiene de ésta. En cuanto a los datos publicados, no existe diferenciación en tipos, y, como hemos visto, no hay limitaciones en los nombres de los tópicos. Sin embargo, existen dos formas alternativas de operar.

Message broker

Es la forma que hemos comentado, en la cual los mensajes se publican en un tópico al cual es posible suscribirse y así redistribuir la información entre los consumidores. Los mensajes a su vez son analizados por el motor de reglas, el cual decide si ejecutar alguna que puede reenviar dichos mensajes hacia otros productos o incluso republicar en otros tópicos. Los consumidores interesados pueden obtener los datos suscribiéndose a los respectivos tópicos o accediendo a la API de alguno de los productos destino de alguna regla.

Basic ingest

Es posible saltearse el *message broker* y publicar en tópicos prefijados que comunican directamente con una regla; es decir, se envía el mensaje directamente a una regla específica del motor de reglas. Esto permite

5 https://docs.aws.amazon.com/iot/latest/developerguide/iot-device-shadows.html

reducir costos (los asociados a mensajería) a cambio de perder la flexibilidad del *message broker*. No es posible suscribirse a dichos tópicos prefijados y por consiguiente estos mensajes sólo pueden observarse por otros dispositivos si la regla en cuestión los re-publica o los obtienen mediante la API de un producto destino de dicha regla.

Manejo de dispositivos

El manejo de los dispositivos es amplio. En principio se hace a través de la *registry*; una base de dispositivos con características afines. Allí podemos crear y borrar “cosas” (things) y aplicar información adicional en forma de atributos. Agrupándolas en tipos y grupos disponemos de mayor capacidad de manejo. Habilitando el servicio de indexado de flota (Fleet indexing service) disponemos de mayor flexibilidad a la hora de ubicar dispositivos o grupos de éstos mediante características particulares, con un cargo adicional.
No existe el concepto de comandos como en GCP, aunque nada impide implementar un servicio similar mediante el *message broker*; dado que tenemos total libertad en la creación de tópicos en los que publicar o suscribirse. Existe además un servicio mediante el cual es posible establecer un túnel seguro con el dispositivo, de modo de poder operar de la forma en que éste permita hacerlo. Adicionalmente, existen otras opciones más complejas como los *jobs*, que describimos más adelante.

Configuración y estado

Todo lo relacionado a la configuración del dispositivo se realiza a través de la *shadow*; a la cual accedemos como hemos descripto.

Aprovisionamiento y manejo a escala

Es posible ingresar manualmente uno por uno los dispositivos, o incorporarlos mediante una aplicación tipo CLI (*aws*) que a su vez hace uso de una API, o podemos escribir una aplicación que la use. Para facilitar el escalado se proveen plantillas.
Sin embargo, existen esquemas de operación destinados al aprovisionamiento masivo, según el nivel de flexibilidad deseado, mucho más poderosos.
Si podemos cargar de manera segura el certificado correspondiente (y único) en cada dispositivo antes de conectarlos a la red, podemos:

- Iniciar una tarea de registro masivo, que en base a una plantilla e información contenida en un archivo JSON alojado en un bucket S3 procede a registrar los dispositivos.
- Emplear JITP (Just-in-time Provisioning), que de manera simplificada podemos decir que opera de la siguiente forma: cuando un dispositivo desconocido con un certificado firmado por una CA registrada (y por lo tanto conocida) intenta conectarse, AWS IoT lo desconecta y basándose en una plantilla automáticamente registra un nuevo dispositivo. Finalizado este proceso, el dispositivo ya formará parte de la red y podrá intentar conectarse nuevamente.
- Emplear JITR (Just-in-time Registration), más flexible y complejo que JITP.

De no ser esto posible, existe una forma de realizar el registro de forma automática empleando certificados provisorios, podemos por ejemplo:

- Entregar los dispositivos con un certificado provisorio. Cuando un dispositivo se conecta, la regla asociada dispara el proceso de ingreso a la red y envía el certificado definitivo al dispositivo, que deberá desconectarse y reconectarse con el nuevo (y definitivo) certificado.
- Entregar los dispositivos sin certificados. Al momento de instalación, un usuario autorizado, por ejemplo un instalador de la empresa, utiliza una aplicación que se conecta a AWS IoT y recibe un certificado provisorio, que transfiere al dispositivo. Luego el dispositivo realiza el proceso descripto.

Para el manejo de los dispositivos existe un esquema de tareas (*jobs*) basado en tópicos prefijados, mediante el cual los dispositivos reciben indicaciones de tareas a realizar y van informando el estado y resultado de las mismas. Esto permite distribuir tareas o actualizaciones e ir realizándolas por grupos de dispositivos. El código necesario puede obtenerse en el SDK.[6]

Actualizaciones de firmware

Si bien nada impide que los dispositivos realicen conexiones a otros destinos ni acepten conexiones de otros entes, AWS IoT provee todo lo necesario para actualizar y mantener actualizado el firmware de los dispositivos, empleando el esquema de *jobs* descripto.

Forma de utilización

Antes de comenzar, recomendamos enfáticamente olvidarse de todos los tutoriales y artículos que se encuentran diseminados por la Internet. Se trata de un esquema complejo y poderoso, en el que no sólo se han definido las tareas a realizar para que no debamos reinventarlas, sino que también se nos provee del código que las realiza. Para comprender la forma de trabajar con esta plataforma debemos seguir las guías de inicio rápido (*quickstart*) oficiales de Amazon y los tutoriales, ambos disponibles en la guía para desarrolladores. Clarificaremos a continuación algunos puntos que nos han suscitado dudas

- El contenido de las pantallas que se acceden en la consola, difiere (a veces considerablemente) de lo que se indica en los tutoriales.
  - el *quickstart*[7] recibe en un tópico diferente al que publica y la información enviada no se muestra en pantalla como se indica allí.

Conexión y operación desde un dispositivo

El *quickstart*[8] nos guía a crear un dispositivo y bajar un kit de conexión (connection kit) en el lenguaje y plataforma de nuestra elección, conteniendo las credenciales creadas por AWS a tal efecto. El ejemplo no nos ofrece el detalle en bajo nivel de lo que debemos hacer, sino que todas las operaciones están contenidas dentro de una library y se espera que saquemos provecho de ella. Dicho ejemplo:

- se conecta al broker
- publica datos periódicamente en un tópico
- se suscribe a otro tópico, mostrando en pantalla lo que recibe por él

En síntesis, tiene la información necesaria para la operación básica de comunicación de datos en la plataforma, ya que en esencia eso es lo que debemos hacer tanto en un dispositivo como en cualquier consumidor que requiera observar en tiempo real los datos producidos por un dispositivo; el resto corresponde a lo que nuestra aplicación requiera.
Podemos observar los datos enviados y enviarle datos a la demo desde el cliente MQTT que nos provee la consola de AWS.

Recepción y utilización de datos por otras “cosas”

Dado que se trata de un broker MQTT standard, cualquier entidad registrada (una “cosa”, *thing*) con su correspondiente certificado y un permiso que la autorice se puede suscribir a los tópicos de su interés para su aplicación, sea un microcontrolador o una computadora. Podemos, por ejemplo, registrar una *thing* y con esos certificados conectar un cliente mosquitto para observar los datos o tomarlos para procesarlos en nuestra computadora, servidor, etc.

Utilización de datos en el resto de la plataforma

6 Al momento de escribir este texto, el SDK para embedded C que soporta esta facilidad es la versión 4 y se encuentra en estado beta.
7 al menos en JavaScript al momento de realizarlo, no hemos probado otras alternativas.
8 https://docs.aws.amazon.com/iot/latest/developerguide/iot-quick-start.html

Como hemos comentado, los datos pasan por un motor de reglas (*rules engine*) y de allí es posible enviarlos a cualquier otro servicio de AWS o re-publicarlos a otro tópico. En el servicio de nuestro interés tendremos los mismos datos que enviamos, aunque es posible además separar o cambiar formato de algunos campos antes de almacenarlos dentro de AWS IoT para procesarlos luego.
Entre las opciones de capacitación gratuitas disponibles, uno de los primeros cursos requeridos nos guía paso a paso a obtener datos de telemetría, almacenarlos, y graficarlos en Amazon QuickSight mediante AWS IoT Analytics.[9]

Cambios de configuración/estado

En la documentación hay una simulación paso a paso en la que manualmente operamos en un cliente MQTT como si fuéramos un dispositivo y en el CLI como si fuéramos una aplicación.[10] Esto nos permite observar de manera detallada todos los pasos de operación. De todos modos, los SDK disponibles proveen código para realizar esta función.[11]

Autenticación y Seguridad en API

El acceso mediante la API utiliza un sistema de autenticación y autorización del tipo Identity and Access Management (IAM), es posible fijar roles y asignar credenciales que permiten regular el acceso.[12] Mediante Amazon Cognito[13] es posible vincular otros sistemas de autenticación.

Logging

El servicio AWS IoT Device Management permite capturar y acceder a logs de los dispositivos.

Esquema de precios

Si bien esto parece algo fuera de lugar, es información importante a la hora de decidir qué reportamos y cómo lo hacemos. La razón es que cada proveedor cobra por el servicio de una forma diferente, y Amazon no es la excepción. La forma en que el proveedor identifica la unidad de tarifado tiene influencia en cómo deberemos empaquetar la información en caso que deseemos minimizar el costo. Nos referimos aquí solamente a AWS IoT, otros productos tienen su esquema e incluso hay disponible un estimador para poder tener una idea de los costos involucrados en nuestro proyecto.[14]
Un detalle importante a tener en cuenta es que existen algunos servicios que no tienen cargo por debajo de un determinado consumo, algunos indefinidamente y otros por un período de 12 meses desde abierta la cuenta; esto se denomina AWS Free Tier.[15]
Los siguientes son algunos detalles de AWS IoT Core al momento de escribir este texto:

- La unidad mínima es de 5Kbytes. El tamaño del mensaje determina la cantidad de “operaciones”, por ejemplo, si enviamos mensajes de 12KB son tres operaciones, si enviamos mensajes más cortos que 5KB, Amazon nos cobrará una operación.
- Las conexiones y suscripciones se cobran de igual forma que los mensajes.

9 https://www.aws.training/Details/eLearning?id=48709
10 https://docs.aws.amazon.com/iot/latest/developerguide/using-device-shadows.html
11 Se proveen implementaciones para C++, Python, Java y JavaScript, un kit para Android y iOS, y una en C para dispositivos embedded con poca capacidad de memoria (embedded C). https://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html
12 https://docs.aws.amazon.com/iot/latest/developerguide/security-iam.html
13 https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html
14 https://calculator.aws/#/
15 https://aws.amazon.com/free/?nc2=h_ql_pr_ft&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=tier%23always-free%7Ctier%2312monthsfree&awsf.Free%20Tier%20Categories=categories%23iot%7Ccategories%23databases

- Las confirmaciones de recepción (ACKs) de MQTT de los dispositivos se cobran como un mensaje, por ejemplo enviar 10 bytes a un dispositivo genera cargos por 10Kbytes (5K por el mensaje y 5K por el PUBACK). Publicar, en cambio, genera cargos por el mensaje solamente.
- Los pings usados para mantener viva la conexión, no se cobran. El tiempo mínimo entre pings es de 30 segundos, si se realiza más frecuentemente, la red puede no contestar.
- Publicar en los tópicos prefijados mediante el método de *basic ingest* no tiene cargos por mensaje.
- Cada acceso a una *shadow* por MQTT es obviamente un mensaje que se publica y se cobra como tal, pero la unidad mínima es de 1KB. Si se accede por API, tiene el cargo de API.
- El motor de reglas cobra por regla disparada y acciones ejecutadas; pero también está sujeto a los tamaños de mensajes. Una regla que no ejecuta acciones tiene cargos por una acción. Una regla que ejecuta dos acciones, disparada por mensajes de entre 5 y 10KB se cobra como dos reglas y cuatro acciones. Puede haber cargos adicionales si las acciones implican transferencias fuera de la región. Las funciones aritméticas o lógicas de la evaluación no se consideran como acciones. Si la regla es disparada por un mensaje generado por AWS IoT Core (por ejemplo un *delta* de un *shadow update*) se cobra como un mensaje de 5KB independientemente de si lo excede.
- Existe un límite *free tier* por debajo del cual no se cobra durante los 12 meses siguientes a la apertura de la cuenta.
- Amazon cobra por estar conectado, el costo es de aproximadamente US$ 0,08 por millón de minutos.
  - Cincuenta dispositivos conectados todo el mes están por debajo del límite de 2,25 millones de minutos de *free tier*.
  - Mil dispositivos conectados todo un mes tendrían un cargo de aproximadamente US$ 3,50.
- Enviar hasta mil millones de mensajes cuesta aproximadamente US$ 1 por millón de mensajes, reduciéndose a US$ 0,7 por millón si se exceden los cinco mil millones.
  - Un millón de mensajes al mes corresponde a publicar mensajes de telemetría de menos de 5KB cada 3 segundos.[16]
  - El límite de *free tier* se ubica en quinientos mil mensajes.
- Los mensajes relacionados al acceso a la *shadow* tienen un costo de aproximadamenteUS$ 1,25 por millón de mensajes, el *free tier* se ubica en los doscientos veinticinco mil mensajes.
- El motor de reglas cobra US$ 0,15 por millón de reglas y US$ 0,15 por millón de acciones. El *free tier* se ubica en doscientos cincuenta mil disparos de reglas y doscientas cincuenta mil acciones ejecutadas.
  - Cada dispositivo generando datos de telemetría cada 10 segundos, genera aproximadamente esa cantidad mensual de oportunidades de disparar reglas y acciones, por un costo de US$ 0,30.

Para más detalles se sugiere consultar la documentación oficial.[17]

En cuanto a otros servicios dentro de AWS IoT, como por ejemplo AWS IoT Device Management, podemos consultar las tarifas en la respectiva documentación[18].

16 A fin de intentar comparar con Google, que cobra por MB, un millón de mensajes puede transportar en el orden del GB, dependiendo de todo lo comentado anteriormente, hasta un máximo de 5GB. Transportar 250GB requeriría aproximadamente, redondeando, unos cien millones de mensajes, con un costo aproximado de unos US$ 100. De todos modos para comparar correctamente se debería hacer un análisis detallado y emplear los tarifadores que provee cada prestador; por ejemplo, el esquema de autenticación de Google fuerza reconexiones periódicas cuando expira el JWT, y a cada reconexión la configuración se envía hasta dos veces, lo cual debe tenerse en cuenta.

17 https://aws.amazon.com/iot-core/pricing/

18 https://aws.amazon.com/iot-device-management/pricing/