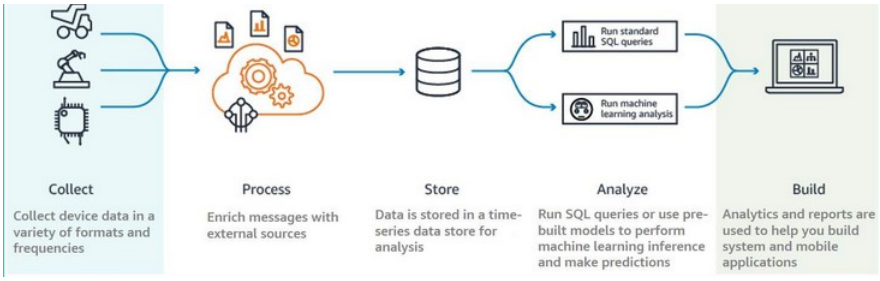
Nota de Aplicación: CAN-107
Título: **AWS IoT Analytics para ingenieros y desarrolladores de sistemas dedicados**
Autor: Sergio R. Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	18/09/20	

En el CTC-104 analizamos los Amazon Web Services (AWS) y desarrollamos la utilización de AWS IoT Core, el servicio de conectividad, utilizando MQTT. En el CTC-105 vimos como conectarnos a dicha plataforma IoT con un ESP32 y Mongoose-OS, enviando datos de telemetría; mientras que en el CTC-106 lo hicimos con módulos SIMCOM. En esta oportunidad nos adentraremos un poco en el otro lado, en el almacenamiento y presentación posterior de los datos, lo cual haremos desde la óptica de un ingeniero o desarrollador de sistemas dedicados; es decir, no con la intención de servir páginas web a proyectos de clientes sino como una herramienta más para poder observar el funcionamiento de nuestros equipos o proveer de información a quienes deben tomar decisiones de negocios en base a estos datos.

Breve descripción de uso de AWS IoT Analytics

En principio, usamos este servicio por simpleza y conveniencia; para una aplicación IoT real puede no ser la mejor opción y de esto nuestros colegas de sistemas saben mucho más de lo que podamos analizar aquí.
Se trata de un componente más de la serie de servicios de AWS IoT[1]. El mismo nos permite almacenar nuestros datos en una base a tal efecto, para luego extraerlos posteriormente con el fin de analizarlos, disponiendo de diversas herramientas de procesamiento y selección en cada una de sus etapas constitutivas.
El proceso de operación es el indicado en la figura siguiente:

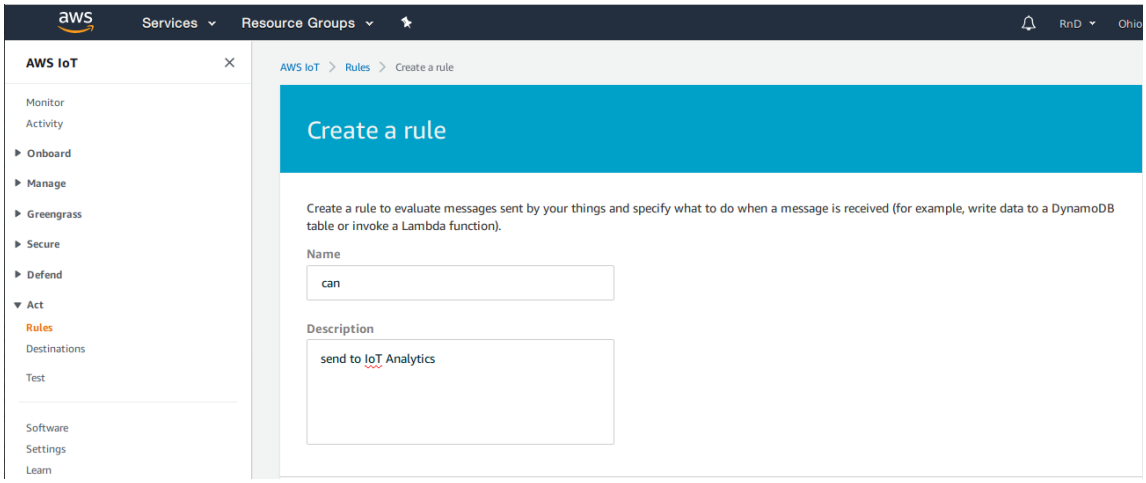


- Definimos un canal (*channel*) mediante el cual ingresan los datos. Esto lo haremos en AWS IoT Core empleando el motor de reglas para direccionar datos ingresados por MQTT a AWS IoT Analytics.
- Procesamos los datos en una *pipeline*, si necesitamos realizar conversiones de unidades o algo similar podemos realizarlo aquí
- Almacenamos los datos en un *data store*
- Analizamos los datos mediante una consulta en base de datos, obteniendo de todos los disponibles aquéllos que nos interesan y generando un *data set*
- Construimos a partir de los *data sets*, los gráficos que nos permitirán observar lo que deseamos y así poder sacar conclusiones. Esto lo haremos en Quicksight

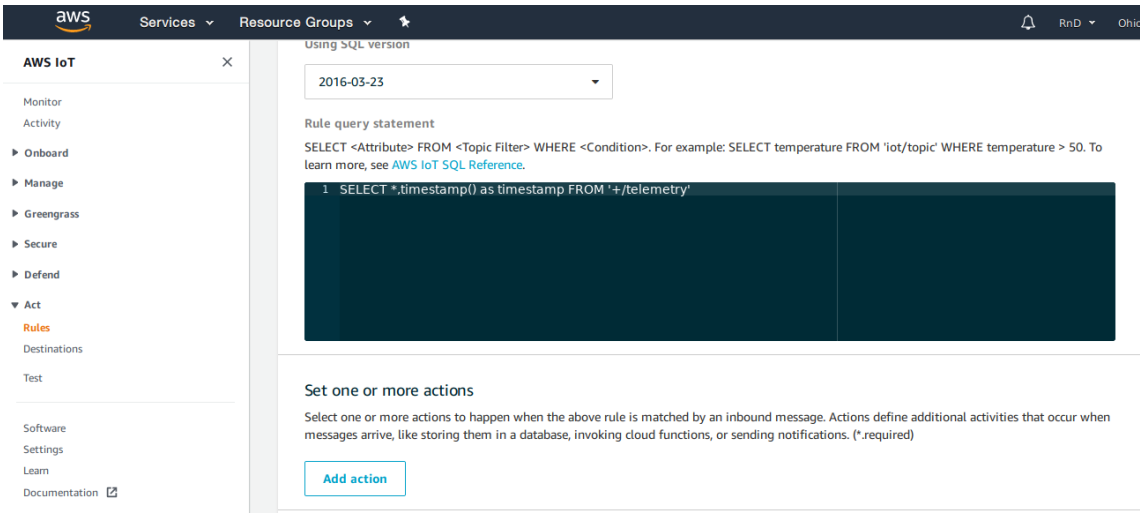
1 Es fundamental en este punto haber leído el CTC-104 para conocer de qué estamos hablando.

En AWS IoT Core

Debemos crear una regla que nos permita tomar determinados mensajes y enviarlos a IoT Analytics. En el CTC-105 generamos datos de telemetría enviando periódicamente la información de memoria libre en el ESP32; lamentablemente no incluimos un timestamp, pero podemos agregarlo aquí.[1]
Comenzamos en la consola seleccionando *Act → Rules* y allí *Create a rule*. Ingresamos el nombre y una descripción.

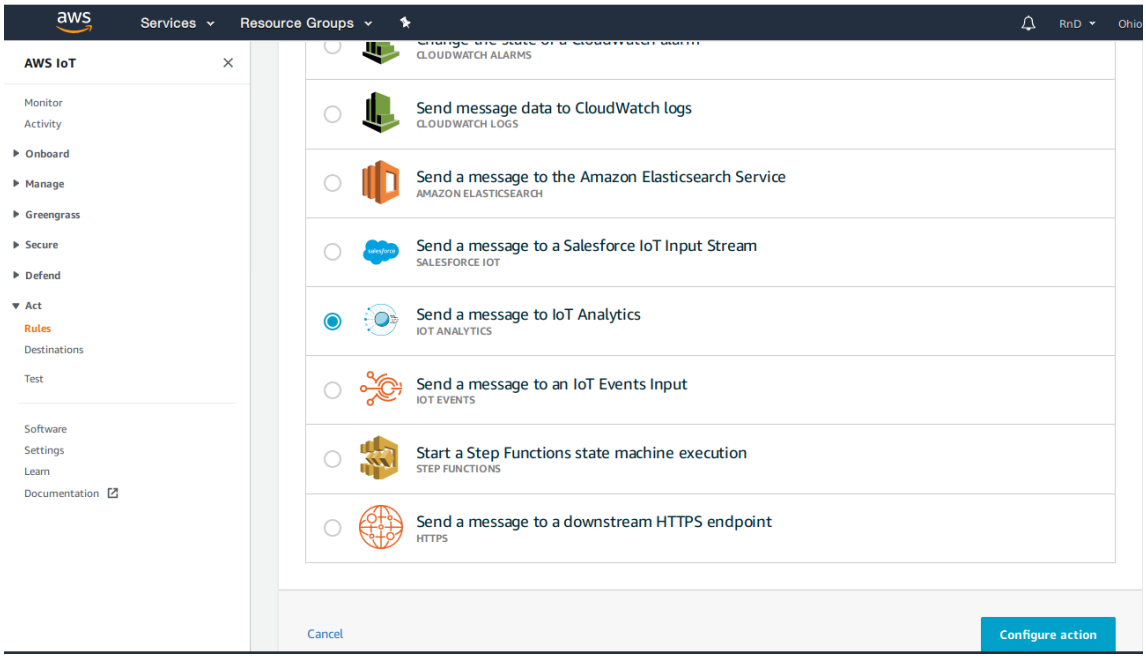


Definimos luego la consulta que vamos a hacer en base de datos. Como indicamos, debemos agregar un campo de timestamp, para lo cual nos valemos de las funciones provistas; la sintaxis es similar a SQL, nuestro query es: `SELECT *,timestamp() as timestamp FROM '+/telemetry'`

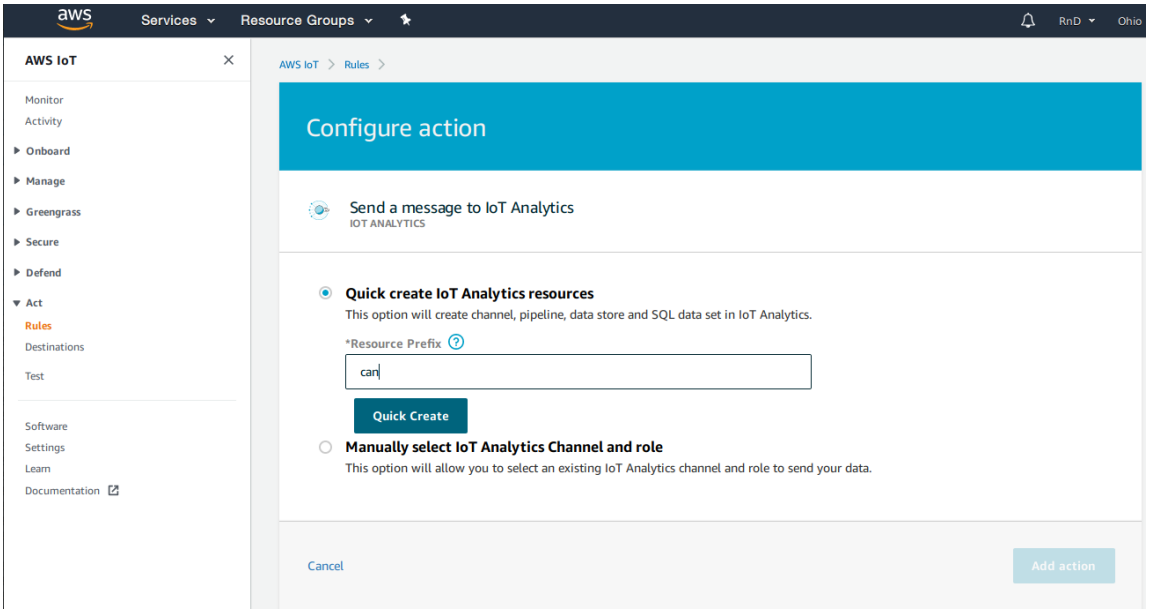


Creamos luego una acción, lo que nos permitirá hacer algo cuando se dispara la regla. La acción es “enviar un mensaje a IoT Analytics”

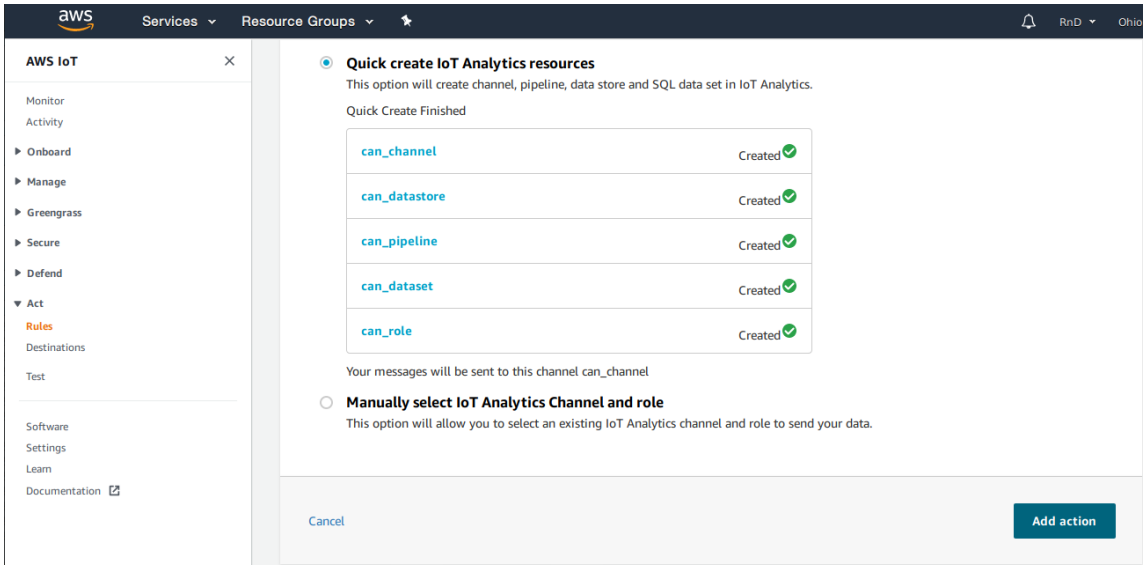
1 En principio es deseable acompañar los datos de un timestamp, dado que corresponde al momento de generación de la información. Sin embargo, esto requiere la presencia de un RTC o equivalente (SNTP y mecanismo asociado, por ejemplo) en el dispositivo, lo cual no siempre es posible por costo o limitaciones. Un paso siguiente es valerse de un recurso de la nube para introducirlo, siempre que la demora asociada sea mucho menor que el tiempo entre mensajes podemos hacerlo sin mayores inconvenientes. De todos modos, el análisis debe hacerse en función de lo que se desea de la aplicación, siendo éstas consideraciones relacionadas a una nota de aplicación genérica.

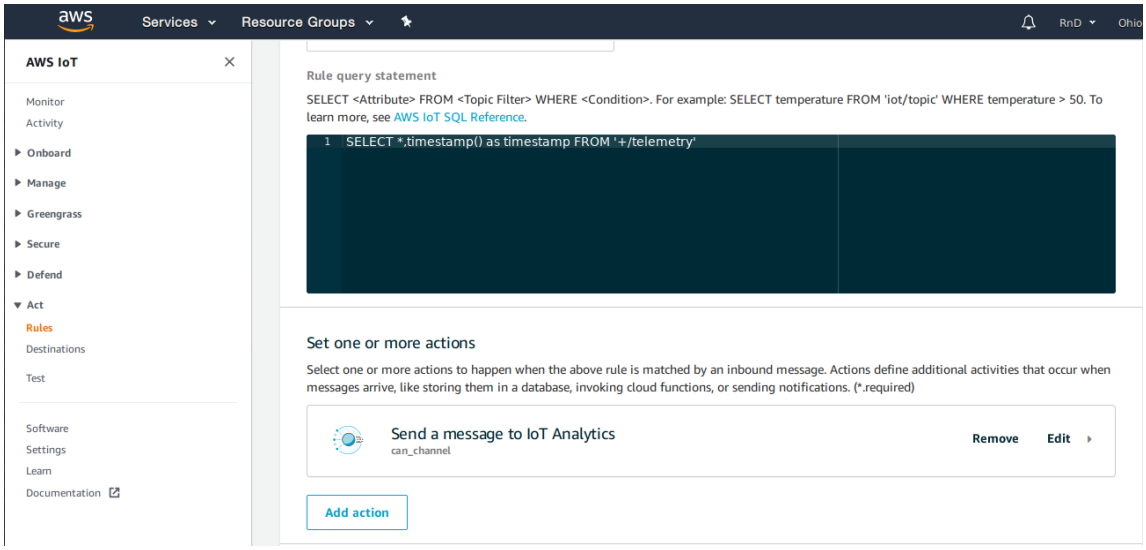


Seleccionando la forma rápida (*Quick create*) se crea todo lo necesario.

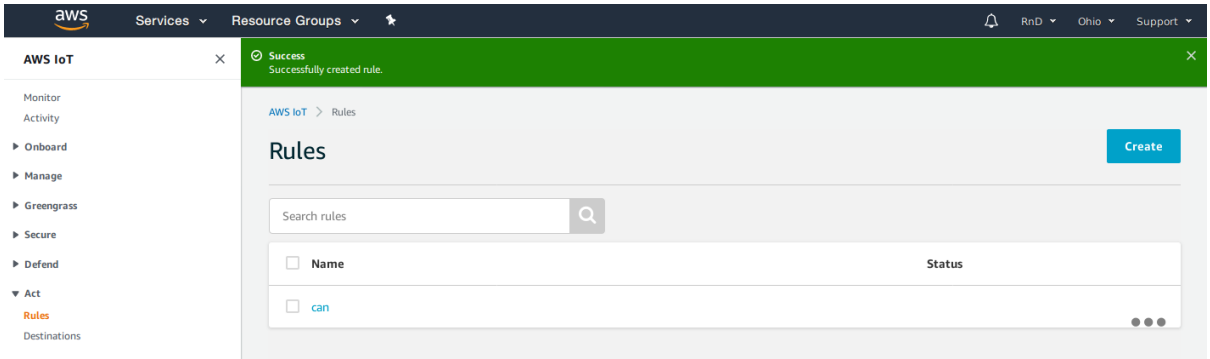


Ponemos simplemente el nombre deseado y se crean el *channel*, *pipeline*, *data store* y *data set*; y un *role* (permisos IAM necesarios para que IoT Core pueda acceder por nosotros a IoT Analytics) asociado.





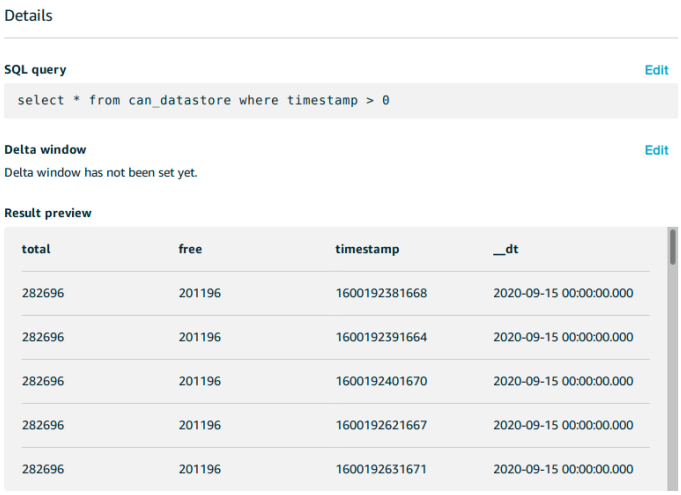
Con esto finalizamos



A partir de este momento, los mensajes MQTT que satisfagan la consulta SQL (todos los mensajes publicados en el tópico que usamos para reportes de telemetría) recibirán un timestamp y serán almacenados en el *data store* de que disponemos en IoT Analytics.

En AWS IoT Analytics

Todo lo necesario ha sido creado ya, sin embargo, es conveniente modificar la configuración del *data store*, a fin de que no guarde los datos por siempre sino por un determinado tiempo.
El uso de este servicio tiene un costo, por procesamiento, por consulta en base de datos y por almacenamiento.[1] Dentro de los usos mencionados aquí, estamos dentro del *free tier* por el período de doce meses desde la apertura de nuestra cuenta AWS.
De modo similar, podemos configurar la *pipeline* para eliminar o procesar datos, acorde a nuestras necesidades, y configurar consultas.
Para poblar el *data set*, podemos configurar un disparo cada un determinado tiempo. En este caso lo haremos manualmente. La acción se realiza mediante otra consulta con sintaxis SQL, en este caso como al principio teníamos datos a los que no habíamos agregado el timestamp, generamos una consulta `SELECT * FROM can_datastore WHERE timestamp > 0` para poblar el *data set*.

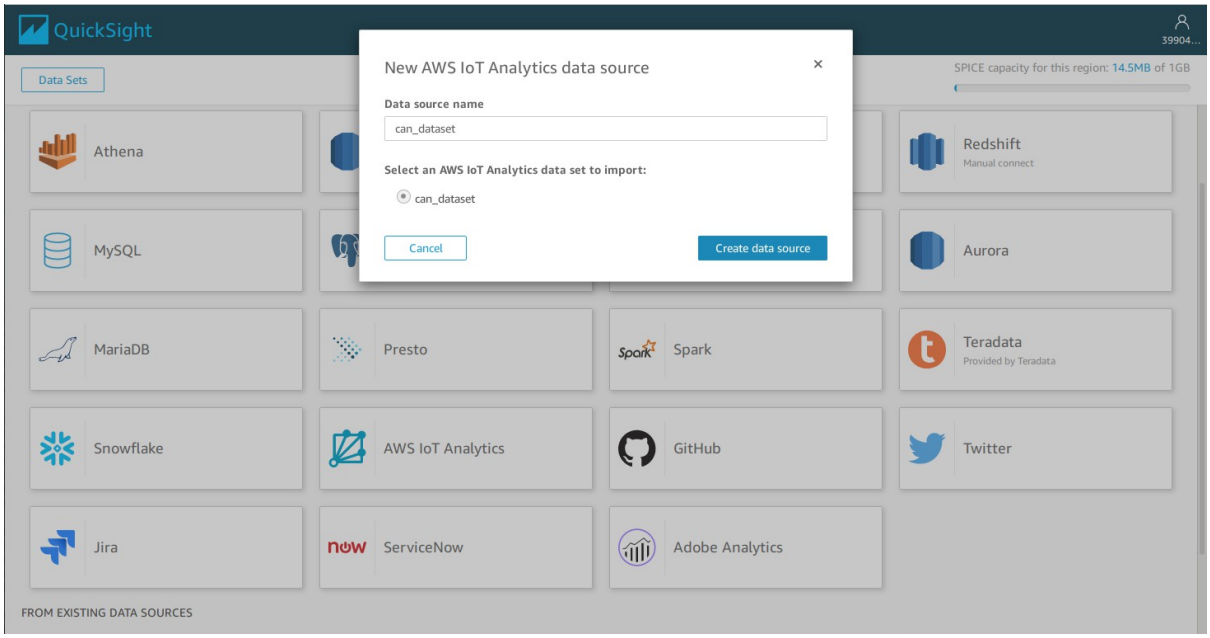


Aquí, “total” y “free” son los campos contenidos en los mensajes de telemetría que enviamos desde el ESP32 con Mongoose-OS en el CTC-105; “timestamp” lo agregamos en IoT Core en el motor de reglas, como hemos visto.[2]

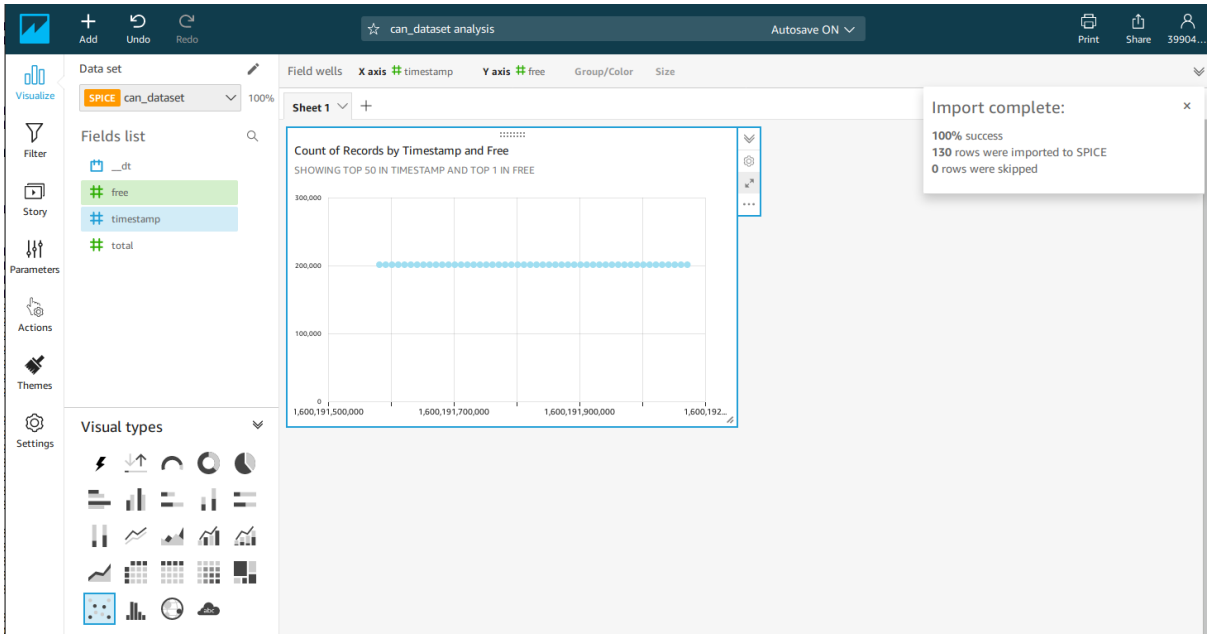
En Quicksight

Quicksight es un servicio del paquete de servicios Analytics (no IoT Analytics).
Antes de poder hacer cualquier cosa, deberemos habilitar este servicio eligiendo una forma de uso. Podemos elegir las prestaciones en función del costo, en este caso hemos elegido la opción gratuita que permite que sólo un usuario con capacidad de crear gráficos disponga de 1GB de almacenamiento.
Aquí todo es muy gráfico, en principio para realizar un gráfico simple y acorde a nuestras necesidades sólo debemos configurar que importamos de un data set en AWS IoT Analytics

1 https://aws.amazon.com/iot-analytics/pricing/
2 Corresponde por lo tanto al momento de recepción, no al de envío. En las pruebas realizadas observamos que no existe demora apreciable entre el momento en que se envía la información y el que se la observa en un cliente MQTT, dado que se trata completamente de un broker MQTT. Los mensajes han sido generados cada 10 segundos y podemos observar una diferencia levemente mayor a 10000 ms entre los respectivos timestamps. Otras plataformas, como GCP, realizan un procesamiento por lotes (en PubSub) y la operación es diferente.



para luego proceder a generar el gráfico



Como podemos apreciar, hemos logrado visualizar la evolución de una variable de telemetría en función del tiempo a lo largo de un tiempo determinado (el cual fijamos al momento de poblar el data set, en IoT Analytics).[1]

De aquí en más, es posible compartir los gráficos con otras personas y realizar cosas más complejas, para lo cual remitimos a la documentación.[2]

1 Si bien el gráfico no resulta muy interesante debido a que la variable no ha cambiado durante la medición (lo cual es bueno porque se trata de la cantidad de memoria libre en la *heap*), podemos apreciar el procedimiento y la utilidad del servicio.

2 https://docs.aws.amazon.com/quicksight/latest/user/welcome.html