AWS IoT

|  |  |
| --- | --- |
| Fecha | 20210117 |
| Autor | Ramón Junquera |

**Índice**

[​ Objetivo 1](#__RefHeading___Toc291_451637037)

[​ Configuración de AWS 1](#__RefHeading___Toc1353_4144735185)

[​ Probando en AWS 3](#__RefHeading___Toc1355_4144735185)

[​ Probando en mosquitto 3](#__RefHeading___Toc1357_4144735185)

[​ Sombras 3](#__RefHeading___Toc181_3212176281)

[Tópico update 7](#__RefHeading___Toc557_4191616741)

[Tópico get 11](#__RefHeading___Toc559_4191616741)

[Tópico delete 12](#__RefHeading___Toc561_4191616741)

# Objetivo

El objetivo del documento es crear un dispositivo virtual en AWS y trabajar con él a través del protocolo MQTT.

Utilizaremos **mosquitto** como cliente.

Es conveniente tener conocimientos acerca del protocolo MQTT y del funcionamiento de la aplicación **mosquitto**, tanto como servidor como sus consolas.

# Configuración de AWS

AWS (Amazon Web Services) es una red preparada para la gestión de dispositivos IoT.

Comenzaremos creando la cuenta:

* Visitamos <https://aws.amazon.com/es/> y creamos una cuenta
* Desde la consola de administración (<https://console.aws.amazon.com/console/home?region=us-east-1>) podemos acceder a todos los servicios que tenemos contratados
* Localizamos el servicio "IoT Core" ([https://console.aws.amazon.com/iot/home?region=us-east-1#/](https://console.aws.amazon.com/iot/home?region=us-east-1" \l "/))

Desde IoT Core podemos gestionar todo lo que necesitamos para este proyecto.

Crearemos un dispositivo virtual y sus certificados:

* Desde el menú del panel izquierdo, seleccionamos: Administración/Objetos
* Seleccionamos "Registrar un objeto".
* Seleccionamos "Crear un solo objeto".
* Completamos el campo "Nombre" con "Atom\_obj".
* Pulsamos el botón "Siguiente".
* Pulsamos el botón "Crear certificado". Aparece una pantalla con 3 líneas: el certificado del objeto, su clave pública y su clave privada.
* Descargamos los 3 archivos a local mediante sus enlaces.
* Bajo las 3 líneas encontramos un enlace para descargar el certificado raíz para AWS IoT. Lo pulsamos.
* Se abre una nueva pestaña. Podemos localizar el enlace para descargar el certificado raíz "Amazon Root CA 1". Lo pulsamos y se abre una nueva pestaña con el contenido del certificado (sólo texto).
* Click con el botón derecho sobre el texto del certificado y guardar como...
* Dejamos el mismo nombre que propone (AmazonRootCA1.pem) y lo guardamos junto con los otros 3 archivos descargados.
* Cerramos la pestaña del texto del certificado raíz.
* Cerramos la pestaña con los enlaces para el certificado raíz y regresamos a la pestaña que nos permitía descargar el certificado del objeto y sus claves.
* Lo activamos pulsando el botón "Activar". Se muestra un mensaje indicando que el certificado se ha creado.

Renombrar archivos:

Los archivos de claves y certificados tienen un prefijo que corresponde con el identificador interno del objeto en AWS. Esta información no es útil para nosotros.

Renombraremos los archivos sustituyendo este prefijo por el nombre de nuestro objeto, que es este ejemplo es Atom.

El resultado final quedaría con los siguientes nombres de archivos:

* AmazonRootCA1.pem
* Atom-certificate.pem.crt
* Atom-private.pem.key
* Atom-public.pem.key

Crearemos una política:

* Desde el menú del panel izquierdo, seleccionamos: Seguridad/Políticas.
* Pulsamos el botón "Crear una política".
* Completamos el campo "Nombre" con "Atom\_pol".
* Completamos el campo "Acción" con "\*".
* Completamos el campo "ARN de recurso" con "\*"
* Marcamos el checkbox "Permitir".
* Pulsamos el botón "Crear".

Asociamos el certificado a la política:

* Desde el menú del panel izquierdo, seleccionamos: Seguridad/Certificados.
* Encontraremos la línea que hace referencia al certificado del objeto creado. Si tenemos varios certificados, sólo hay que recordar que se muestras en orden inverso de creación. El más reciente será el primero.
* Click sobre el icono de los tres puntos de la línea del certificado de nuestro objeto y seleccionar "Asociar política". Aparece una ventana con el listado de políticas creadas.
* Marcaremos el checkbox de la política "Atom\_pol"
* Pulsamos el botón "Asociar".

# Probando en AWS

Una vez que el objeto está creado, podemos interactuar con él a través del protocolo MQTT.

Las restricciones son que sólo se permiten conexiones seguras y a través de certificados.

AWS nos ofrece un sencillo cliente integrado en la web que ya está configurado.

Haremos una prueba...

Suscripción:

* Desde el menú del panel izquierdo, seleccionamos: Prueba/Cliente de prueba de MQTT
* Completamos el campo "Tema de suscripción" con el valor "test". Nos estamos suscribiendo al tópico test (tema = tópico).
* Pulsamos el botón "Suscribirse al tema" y se crea un apartado llamado test

Publicación:

* En el mismo apartado test tenemos dos controles. El cuadro de texto con el nombre del tópico que ya viene completado con el valor test. Lo dejamos como está.
* Debajo tenemos un cuadro azul que ya contiene un texto en formato JSON.
* Modificaremos el texto del mensaje a “Hola”.
* Pulsamos el botón “Publicar en tema” a la derecha del cuadro de texto del tópico. Al momento vemos como el mensaje enviado se ha recibido.

# Probando en mosquitto

Haremos otro prueba, pero esta vez usaremos los scripts de cliente de mosquitto.

Lo primero que necesitamos conocer el la dirección del broker.

Esto lo encontramos en:

* Desde el menú del panel izquierdo, seleccionamos “Administración/Objetos”
* Localizamos el objeto “Atom\_obj” y hacemos click en su nombre. Se abre una ventana con más detalles.
* Seleccionamos la opción “Interactuar”. Bajo la sección HTTPS encontramos un cuadro azul con la URL completa. Ej: apj0acah6jgct-ats.iot.us-east-1.amazonaws.com

Supondremos que tenemos los archivos de claves y certificados en la carpeta Downloads.

Comprobamos desde el objeto Atom podemos subscribir y publicar en el broker de AWS:

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "test"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "test" -m "{\"message\":\"Hola\"}"

Podemos combinar la suscripción y publicación de mosquitto y de la web de AWS.

# Sombras

Ahora que sabemos que AWS nos facilita un broker MQTT y nos permite crear dispositivos virtuales, sólo nos queda saber qué relación guardan entre ellos.

Un dispositivos virtual no es más que la representación de uno real. En AWS se llaman “thing”.

Los dispositivos reales se diferencian en sus características o propiedades.

Ejemplos:

* Un interruptor tiene la propiedad “estado”, que puede ser “encedido” o “apagado”.
* Un led puede tener la propiedad de “intensidad”, que puede ser un número entero entre 0 y un valor máximo.
* Un led RGB tiene 3 propiedades, la intensidad de cada uno de los colores.

Con los dispositivos virtuales ocurre lo mismo. Podemos asignarles tantas propiedades como sean necesarias. La colección de valores que asignemos a sus propiedades será el estado del dispositivo.

Cuando trabajamos con un dispositivo, necesitamos dos acciones básicas:

* Consultar su estado
* Modificar su estado

Cuando trabajamos con dispositivos reales y solicitamos su estado, el dispositivo consulta el estado actual e informa.

Con un dispositivo virtual no es tan sencillo, porque siempre puede tener la duda de cuál es el estado actual del dispositivo real.

Para intentar evitar estos problemas haremos dos cosas:

* Si el dispositivo real cambia su estado (valor de alguno de sus parámetros), informará al instante al dispositivo virtual.
* Si queremos modificar el estado del dispositivo real, solicitaremos un cambio de estado al dispositivo virtual (solicitaremos el estado deseado). Éste a su vez, solicitará un cambio de estado al dispositivo real. Las siguientes acciones ya vienen cubiertas por el punto anterior, porque el dispositivo real informará al virtual de cualquier cambio.

Por lo tanto, un dispositivo virtual tendrá dos colecciones de parámetros (dos estados):

* El último estado informado (reported).
* El último estado solicitado (desired).

Cuando actualicemos el estado de un dispositivo virtual es muy importante que indiquemos de qué sección hablamos. Normalmente el dispositivo real modificará la sección reported y un usuario o aplicación, la sección desired.

Por supuesto, un usuario o aplicación, también podrá consulta el estado en cualquier momento.

Este sistema de trabajo se llama sombra (shadow). El dispositivo virtual es una sombra del real.

Podemos ver la sombra de un objeto directamente en AWS IoT:

* Seleccionamos en el menú del panel izquierdo la opción Administración/Objetos
* Click sobre el nombre del objeto. En nuestro caso **Atom\_obj**.
* Click sobre la opción **Sombras** del menú vertical. Sólo apararece una sombra llamada **Classic Shadow**
* Click sobre **Classic Shadow** y se mostrará una sección llamada **Estado de la sombra**.

El contenido por defecto de la sombra es:

{

"desired": {

"welcome": "aws-iot"

},

"reported": {

"welcome": "aws-iot"

}

}

Siempre se trabaja con formato JSON.

Podemos comprobar que tiene secciones para los dos estados: **desired** y **reported**.

El estado sólo tiene una propiedad llamada **welcome** con el valor **aws-iot**.

Además comprobamos que ambos estados son idénticos. Está actualizado.

Modificaremos la sombra desde la misma web de AWS.

Tenemos la opción de **Eliminar** y **Editar**.

Hacemos click sobre **Editar**.

El cuadro **Estado de la sombra** se vuelve azul y editable.

Simularemos que solicitamos un cambio de la propiedad **welcome** a **Hola**.

En la sección **desired** cambiaremos el texto de **welcome** de **aws-iot** a **Hola**.

El texto completo quedaría así:

{

"desired": {

"welcome": "Hola"

},

"reported": {

"welcome": "aws-iot"

}

}

Pulsamos en **Guardar**.

Comprobamos que la sombra se ha actualizado a:

{

"desired": {

"welcome": "Hola"

},

"reported": {

"welcome": "aws-iot"

},

"delta": {

"welcome": "Hola"

}

}

Ha quedado reflejada nuestra solicitud. Pero además ha aparecido una nueva sección llamada **delta**.

Esta sección se crea automáticamente cuando queda algo pendiente por hacer. Se han hecho solicitudes que aun no están completadas. La sección **reported** no coincide con **desired**.

Conclusión, si una sombra contiene la sección **delta** es que aun no está sincronizada con su dispositivo real.

Si ahora cambiásemos la propiedad **welcome** en la sección **reported** a **Hola**, desaparecería la sección **delta**, porque los estados serían iguales.

Pregunta: Qué tiene de especial la sombra **Classic Shadow**?

Respuesta: Nada. Simplemente es la sombra por defecto. Pero sólo puede haber una **Classic Shadow**.

Pregunta: Es obligatorio que un dispositivo virtual tenga sombra?

Respuesta: No.

Comprobaremos que esto es así.

Ya sea cuando estamos viendo el contenido de la sombra, o cuando se muestra el listado de sombras, tenemos la opción de eliminar.

Haremos la prueba y eliminaremos la sombra **Classic Shadow**.

Vemos que ahora el listado de sombras aparece vacío.

Pregunta: Podemos volver a crear la sombra **Classic Shadow** desde AWS?

Respuesta: Sí

Crearemos de nuevo la sombra **Classic Shadow**.

En la pantalla de listado de sombras, hacemos click sobre la opción **Agregar una sombra**.

Nos solicita el nombre de la sombra. Puesto que queremos crear la **Classic Shadow**, dejaremos vacío el nombre de la sombra y pulsaremos el botón **Agregar**.

Ya tenemos de nuevo la **Classic Shadow**.

Pregunta: Puedo añadir nuevas propiedades a una sombra?

Respuesta: Sí.

Editaremos de nuevo la sombra y le añadiremos la propiedad **led** con el valor **off**, tanto en la sección de estado deseado como informado. Quedaría así:

{

"desired": {

"welcome": "aws-iot",

"led": "off"

},

"reported": {

"welcome": "aws-iot",

"led": "off"

}

}

Pregunta: He editado de nuevo la sombra par eliminar la propiedad **welcome**, pero no puedo. Me aparece siempre. Porqué?

Respuesta: Las propiedades no se pueden eliminar. Sólo pueden añadirse

Pregunta: Todos mis dispositivos están obligados a tener el parámetro **welcome**?

Respuesta: No. Los dispositivos sólo deberían tener los parámetros necesarios. Daremos respuesta a este caso más adelante cuando pasemos a la práctica.

Pregunta: Puedo crear más de una sombra para un mismo dispositivo?

Respuesta: Sí. No hay ningún problema. Cada sombra debe de tener un nombre único para poder identificarla.

Pregunta: Se comparten los estados de las sombras de un mismo dispositivo?

Respuesta: No. Cada sombra guarda su propio estado.

Entendiendo la teoría, vamos con la práctica.

Pregunta: Cómo nos comunicamos con la sombra?

Respuesta: A través de tópicos del broker.

Todos los tópicos referentes a la sombra por defecto (**Classic Shadow**) tienen el mismo prefijo:

**$aws/things/<nombre de dispositivo virtual>/shadow**

Si no es la sombra por defecto, será:

**$aws/things/<nombre de dispositivo virtual>/shadow/name/<nombre de la sombra>**

En nuestro caso como trabajaremos con la sombra clásica sería: **$aws/things/Atom\_obj/shadow**

Todos los tópicos que comienzan con $ son reservados y tienen funciones especiales.

Si tenemos la necesidad de crear nuestros propios tópicos, no hay problema, siempre que no comiencen por $.

## Tópico update

La primera acción será actualizar la sombra.

Primero lo haremos desde AWS.

Como ya vimos anteriormente, utilizaremos la opción de menú **Prueba/Cliente de prueba de MQTT**.

Nos iremos a la sección **Publicar en un tema**.

El tópico de publicación será: $aws/things/Atom\_obj/shadow/update

Y el mensaje será:

{

"state" : {

"reported" : {

"status" : “off"

}

}

}

Pulsamos el botón **Publicar en tema**

Hemos utilizado el sub-tópico **update** para publicar una actualización del estado.

En el contenido del mensaje podemos comprobar que hablábamos del estado **estate**.

Concretamente del estado informado **reported**.

E indicábamos que el parámetro **status** tiene el valor **off**.

Si abrimos en otra solapa del navegador el contenido de la sombra, podremos ver como ahora es:

{

"desired": {

"welcome": "aws-iot"

},

"reported": {

"welcome": "aws-iot",

"status": "off"

}

}

Pregunta: Porque no aparece la sección delta, si los estados deseados e informados no coinciden?

Respuesta: Porque el dispositivo ha indicado el valor de un parámetro, pero aun nadie ha solicitado ningún cambio para él. Por lo tanto la información está actualizada.

Para hacer esta misma publicación desde el cliente de Mosquitto tendremos que escribir:

mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/update" -m "{\"state\":{\"reported\":{\"status\":\"off\"}}}"

Como podemos ver, la escritura desde la línea de comandos no es tan cómoda.

Para escribir el contenido del mensaje debemos seguir utilizando el formato JSON.

No hace falta que indiquemos todos los espacios ni saltos de línea.

El receptor lo vuelve a parsear para darle un aspecto estructurado.

Pero las comillas se deben respetar. Hay que recordar que en la línea de comandos de Linux, el carácter de comillas dobles se representa como \”. Y el carácter $ también es considerado especial. Y se representa como \$.

Para comprobar que realmente funciona, podemos cambiar el parámetro **status** de **off** a **on** y ver si ha cambiado en la sombra.

De la misma manera que hemos actualizado el estado de **reported**, podemos hacerlo con **desired**.

Acabamos de aprender a utilizar el sub-tópico **update**.

Pregunta: Pero cómo puedo crear una **Classic Shadow** que no tenga el parámetro **welcome**?

Respuesta: Borra la **Classic Shadow** y haz una publicación como si existiese. Se creará una nueva sólo con el parámetro (o parámetros) de la publicación.

De la misma manera que podemos publicar en el tópico **update**, podemos suscribirnos para poder estar enterados de todas las peticiones, pero normalmente no se hace, porque una petición no tiene porqué ser siempre correcta.

El tópico update tiene algunos sub-tópicos interesantes:

* **accepted** : da información sobre las actualizaciones aceptadas
* **rejected** : da información sobre las actualizaciones rechazadas
* **delta** : da información sobre los parámetros pendientes de actualizar
* **documents** : da información completa sobre el cambio (estado anterior, solicitud actual, fechas detalladas, etc).

Veamos cada uno de ellos por separado.

Sub-tópico **update/accepted**

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/update"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/update" -m "{\"state\":{\"reported\":{\"status\":\"off\"}}}"

El resultado de la suscripción es:

{

"state":{

"reported":{

"status":"off"

}

},

"metadata":{

"reported":{

"status":{

"timestamp":1610792402

}

}

},

"version":38,

"timestamp":1610792402

}

El dato timestamp es la hora de Unix: los milisegundos transcurridos desde el 1 de enero de 1970.

Sub-tópico **update/rejected**

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/rejected"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/update" -m "{\"state\":{\"reported\":{\"status\":\"off\"}}"

El resultado de la suscripción es:

{

"code":400,

"message":"Payload contains invalid json"

}

La razón es que nos hemos dejado una llave final por cerrar.

Sub-tópico **update/delta**

Suponiendo que la sombra tiene en la sección **reported** el **status** como **off**, nos suscribiremos a **update/delta** y haremos una publicación para solicitar un cambio a **status** **on**.

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/delta"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/update" -m "{\"state\":{\"reported\":{\"status\":\"on\"}}"

El resultado de la suscripción es:

{

"version":41,

"timestamp":1610793155,

"state":{

"status":"on"

},

"metadata":{

"status":{

"timestamp":1610793155

}

}

}

En la sección **state** nos indica los parámetros que están pendientes de actualización.

Sub-tópico **update/documents**

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/documents"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/update" -m "{\"state\":{\"reported\":{\"status\":\"on\"}}"

El resultado de la suscripción es:

{

"previous":{

"state":{

"desired":{

"status":"on"},

"reported":{

"status":"off"

}

},

"metadata":{

"desired":{

"status":{

"timestamp":1610793155

}

},

"reported":{

"status":{

"timestamp":1610792993

}

}

},

"version":41

},

"current":{

"state":{

"desired":{

"status":"on"

},

"reported":{

"status":"off"

}

},

"metadata":{

"desired":{

"status":{

"timestamp":1610793492

}

},

"reported":{

"status":{

"timestamp":1610792993

}

}

},

"version":42

},

"timestamp":1610793492

}

Después de ver el resultado de los 4 sub-tópicos, entendemos que un dispositivo real se suscriba solamente a sub-tópico **update/delta** y compruebe en la sección **state** los parámetros pendientes por actualizar para aplicarlos.

Los sistemas que necesitan estar al corriente del estado de una sombra, están suscritos al subtópico **update/accepted**. Y cuando les llegue el mensaje comprobarán si se trata de un cambio en el estado **reported**, que es el que importa. Los cambios en el estado **desired**, suelen ser descartados.

Sólo cuando se están haciendo pruebas o tramitando solicitudes manuales te suscribes a los sub-tópicos **rejected** o **documents** para asegurarte de si hay problemas y porqué. Pero cuando se trabaja de manera automática para solicitar cambios de parámetros, no hace falta suscribirse a nada.

Pregunta: Podemos hacer un **update** cuyo mensaje incluya la sección **desired** y **reported** al mismo tiempo?

Respuesta: Sí, pero no es lo más lógico. Lo normal es que la sección **desired** sea completada por los que desean aplicar un cambio en el dispositivo real y la sección **reported** por los que simplemente quieren consultar el estado del dispositivo.

## Tópico get

Aunque para conocer las actualizaciones del estado de un dispositivo, lo normal es que nos suscribamos a update/accepted, cuando ponemos en marcha el proceso, desconocemos el estado inicial.

Sólo para estos casos se utiliza el sub-tópico get. Es una solicitud del estado de una sombra.

Una solicitud de estado no tiene parámetros ni información adicional, por lo tanto la publicación en este sub-tópico tendrá un mensaje vacío.

En la documentación de AWS se indica que no se tendrá en cuenta el contenido del mensaje, pero no es del todo correcto. Si el mensaje está vacío o contiene algún número, es aceptado. En cambio si contiene alguna palabra, es rechazado.

Para asegurarnos, nosotros siempre utilizaremos un mensaje vacío.

Aunque el tópico de publicación para hacer una solicitud de estado es get, necesitaremos suscribirnos a get/accepted para comprobar el resultado de la solicitud.

Sub-tópico **get** y **get/accepted**

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/get/accepted"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/get" -m ""

El resultado de la suscripción es:

{

"state":{

"desired":{

"status":"on"

},

"reported":{

"status":"off"

},

"delta":{

"status":"on"

}

},

"metadata":{

"desired":{

"status":{

"timestamp":1610793492

}

},

"reported":{

"status":{

"timestamp":1610792993

}

}

},

"version":42,

"timestamp":1610794964

}

Nos informa de todos los detalles del estado de la sombra.

Si el que peticiona es el dispositivo real, estará interesado en la sección **delta** (si existe), para sincronizar la sombra.

Si el peticionario es un proceso, estará interesado en la sección **reported** (parámetros reales).

Sub-tópico **get** y **get/rejected**

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/get/rejected"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/get" -m "Hola"

El resultado de la suscripción es:

{

"code":400,

"message":"Payload contains invalid json"

}

La razón es porque el mensaje no está vacío y contiene una palabra.

## Tópico delete

Permite eliminar una sombra.

También tiene sus correspondientes sub-tópicos **accepted** y **rejected** para suscribirse y comprobar si nuestra petición se ha podido llevar a cabo.

Al igual que con el tópico **get**, no necesitamos pasar ningún parámetro en una solicitud de borrado de sombra, así que dejaremos el mensaje vacío.

Sub-tópico **delete** y **delete/accepted**

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/delete/accepted"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/delete" -m ""

El resultado de la suscripción es:

{

"version":42,

"timestamp":1610798238

}

Hemos borrado la **Classic Shadow** y tenemos confirmación de ello.

También podemos comprobarlo en la web de AWS.

Sub-tópico **delete** y **delete/rejected**

Suponemos que no existe la **Classic Shadow**.

**Suscripción**: mosquitto\_sub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/delete/rejected"

**Publicación**: mosquitto\_pub --cafile ~/Downloads/AmazonRootCA1.pem --cert ~/Downloads/Atom-certificate.pem.crt --key ~/Downloads/Atom-private.pem.key -h apj0acah6jgct-ats.iot.us-east-1.amazonaws.com -t "\$aws/things/Atom\_obj/shadow/delete" -m ""

El resultado de la suscripción es:

{

"code":404,

"message":"No shadow exists with name: 'Atom\_obj'"

}

No podemos borrar una sombra inexistente.

Recordamos que para crear una nueva sombra es suficiente con hacer una publicación de su estado.