

Universidad de Alcalá de Henares - EPS

# PL3 - Cloud Computing

Pablo Acereda      David E. Craciunescu      Ángel Martín  
Ángelo Moreno      Laura Pérez

May 26, 2019

## 1 Contenedores

## 2 IA & Machine Learning

## 3 IoT

Partiendo de una definición base, IoT (Internet of Things) es el conjunto de dispositivos que forman una red, la cual permite el intercambio de información y datos entre terminales.

Una vez que se tiene una cierta noción de lo que es IoT, es hora de pasar al desarrollo llevado a cabo en las dos plataformas empleadas: Microsoft Azure y Amazon Web Service.

Como última adenda a esta cuestión, decir que van a incluirse capturas de los procesos llevados a cabo, y para demostrar la realización del trabajo.

### Microsoft Azure

A continuación, van a ser descritos los pasos realizados para controlar un dispositivo conectado a IoT Hub, uno de los servicios encontrados en Microsoft Azure.

#### 3.0.1 Prerrequisitos

Como puede ser comprensible, no todo es tan sencillo como ponerse a escribir y conseguir un resultado de manera directa, primero es necesario que el dispositivo que va a emplearse está preparado con las aplicaciones y herramientas necesarias.

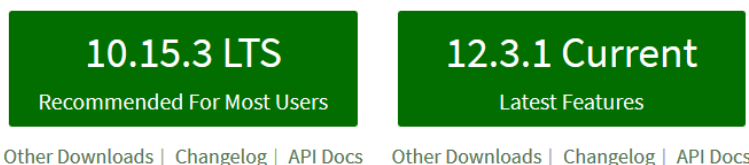
Para ello, hay que asegurarse de que haya instalado una versión de Node.js en el dispositivo. En caso de no saber si se cuenta con una instalación de Node.js, es tan sencillo como ejecutar el siguiente comando 1:

*node --version* (1)

Si se cuenta con una versión (acorde a la documentación oficial [1]) igual o posterior a 4.x.x será necesario descargarla de la que puede encontrarse en [2] 1. Cabe añadir que Node.js no es la única tecnología que puede emplearse para este desarrollo, debido a que también se acepta: .NET, Java, Python y Android; pero por elección del desarrollador de este apartado se ha decidido emplear Node.js.

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

## Download for Windows (x64)



Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

Figure 1: Descarga de Node.js desde el sitio web oficial.

Acorde a la documentación, también es necesario ejecutar el comando ??

```
azextensionadd --nameazure-cli-iot-ext (2)
```

El cual agrega la extensión IoT de Azure para la CLI a la instancia de Cloud Shell.

Por último, en caso de no contar con un dispositivo que conectar, siempre puede emplearse lo proporcionado en <https://github.com/Azure-Samples/azure-iot-samples-node/archive/master.zip>

### 3.0.2 Creación de un centro de IoT

Dentro de Azure portal, cuyo acceso no merece mención debido a la experiencia por parte del lector en este tema, se busca "IoT Hub" en la barra situada en la parte superior del portal.

Una vez dentro de IoT Hub, se crea un nuevo recurso. La pantalla debería quedar tal y como se puede ver en la captura 2. Es necesario seleccionar el grupo al cual pertenecerá el nuevo recurso y, en caso necesario, crear uno nuevo. Por último, se selecciona un nombre para el IoT Hub (en este caso se ha decidido que sea Hefesto, por ser el dios griego del fuego, la forja y los artesanos; al ser el de-

sarrollador que replique este proceso el que de forma y esculpa el funcionamiento del dispositivo).

Figure 2: Inicio del proceso de creación de recurso: asignación de Grupo y Nombre.

Una vez finalizados los pasos de la explicación anterior, se pasa al siguiente punto, definición del tamaño y la escala 3. En esta página, tal y como su nombre indica, se seleccionarán el tamaño (número de unidades) y la escala (de entre las que proporciona Microsoft Azure). De las escala, cabe añadir que se encuentran los siguientes tipos:

- Estándar (desde S1 hasta S3).
- Básico (desde B1 hasta B3).
- Gratuito. De este cabe decir que solo se emplea para pruebas y evaluaciones. Permite 500 dispositivos con el centro de IoT y hasta 8000 mensajes al día. Solo puede crearse una instancia [1].

Se puede pasar a revisar y crear la unidad IoT Hub 4.

### 3.0.3 Registrar Dispositivo

Para registrar el dispositivo que se va a emplear, es necesario acceder a la terminal de Azure. No se va a proporcionar ninguna explicación de como acceder a la misma, debido a que el lector ya es consciente de como llegar hasta ella.

Primero, se debe empezar por crear la identidad del dispositivo 5, con la que será referido a partir de ahora mediante el comando:

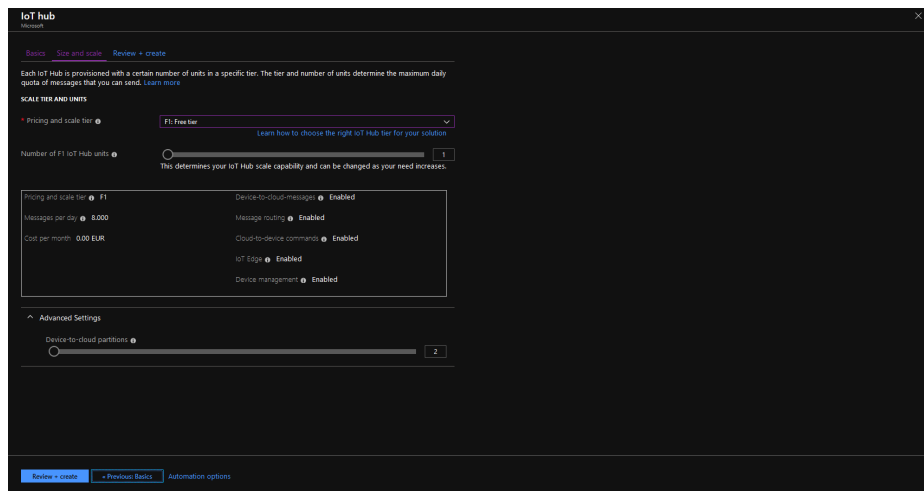


Figure 3: Seleccionar tamaño y escalabilidad

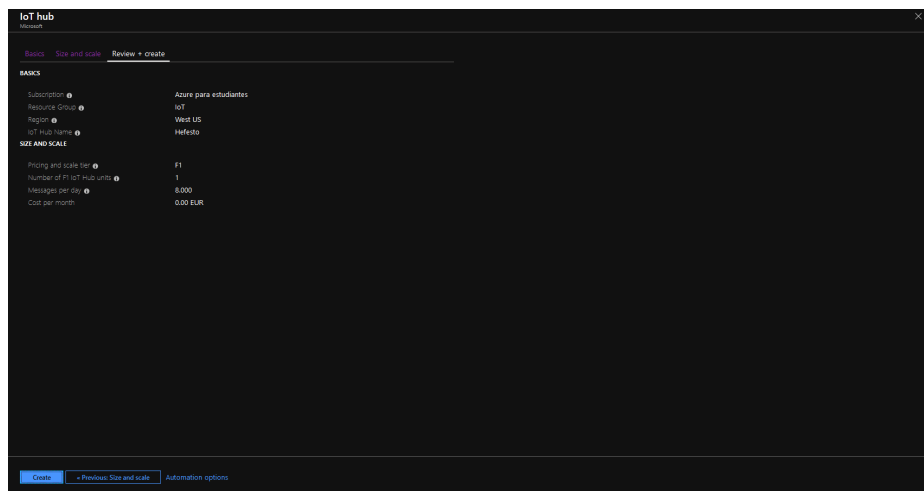


Figure 4: Revisión y Creación.

*aziotHubdevice--identitycreate--hub--nameYourIoTHubName--device--idMyNodeDevice*  
(3)

Dentro de 3 se debe sustituir YourIoTHubName por el nombre del Hub, que en este caso será Hefesto; y MyNodeDevice por el nombre del dispositivo, se ha dejado tal cual por mera comodidad, al tener esta práctica como objetivo aprender

a usar las nociones básicas de las plataformas Cloud.

Al ejecutar queda la siguiente traza 5:

```
acereda@Azure:~$ az iot hub device-identity create --hub-name Hefesto --device-id MyNodeDevice
{
  "authentication": {
    "symmetricKey": {
      "primaryKey": "8yj2ayIR4pYyuWGT0+q2ciU1HfI5B08Zk6CdLiakoIM=",
      "secondaryKey": "5s1CippVQkXxLmh+S11cFRluecppawzZXlb0p0da2ns="
    },
    "type": "sas",
    "x509Thumbprint": {
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    }
  },
  "capabilities": {
    "iotEdge": false
  },
  "cloudToDeviceMessageCount": 0,
  "connectionState": "Disconnected",
  "connectionStateUpdatedTime": "0001-01-01T00:00:00",
  "deviceId": "MyNodeDevice",
  "deviceScope": null,
  "etag": "NjU3MTMyOTc4",
  "generationId": "636944257001021266",
  "lastActivityTime": "0001-01-01T00:00:00",
  "status": "enabled",
  "statusReason": null,
  "statusUpdatedTime": "0001-01-01T00:00:00"
}
```

Figure 5: Registro del dispositivo.

Acto seguido, se debe conseguir la cadena de conexión perteneciente a al dispositivo. Una vez más, se debe ejecutar en la consola un comando:

```
az iot hub device-identity show-connection-string --hub-name YourIoTHubName --device-id MyNodeDevice
(4)
```

Una vez más, se sustituye YourIoTHubName por Hefesto, que es al que se quiere hacer referencia. La cadena resultante de la ejecución del comando debe guardarse 6 para, posteriormente, poder realizar la conexión con el dispositivo.

Figure 6: Conexión con el dispositivo.

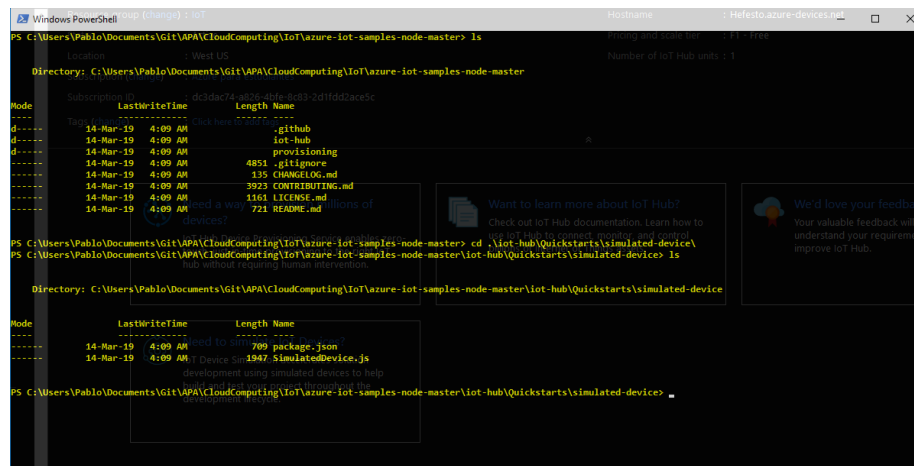
Finalmente, se debe hacer de manera similar, pero en este caso con la cadena de conexión de servicio. Este último permite conectarse al IoT Hub y recuperar los mensajes enviados por el dispositivo.

```
az iot hub show-connection-string --name YourIoTHubName --outputtable
(5)
```

### 3.0.4 Envío de datos de telemetría simulados

Todo lo hecho anteriormente está muy bien y tal, pero no es una “aplicación real”. El dispositivo debería poder mandar datos al fin y al cabo. Para ello se debe abrir una terminal local, e ir al directorio donde se ha descargado, en este caso, el ejemplo del repositorio Git proporcionado anteriormente en este documento. Se debe entrar al directorio *iot-hub*

*Quickstarts*  
*simulated-device*.



The screenshot shows a Windows PowerShell terminal window. The user is in the directory `C:\Users\Pablo\Documents\Git\APPA\CloudComputing\IoT\azure-iot-samples-node-master`. They run `ls` and see a list of files including `.github`, `iot-hub`, `provisioning`, `.gitignore`, `CHANGELOG.md`, `CONTRIBUTING.md`, `LICENSE.md`, and `README.md`. Then they run `cd .\iot-hub\Quickstarts\simulated-device\` and run `ls` again, showing files like `package.json` and `SimulatedDevice.js`. On the right side of the terminal, there is a sidebar with information about IoT Hub, including a link to documentation and a feedback form.

Figure 7: Conexión con el dispositivo.

Debe sustituirse el contenido de *connectionString* 7 a la salida obtenida en la ejecución de 4 en el archivo *SimulatedDevice.js*.

Tras guardar el archivo, ya puede ejecutarse el código:

*npm install* (6)

*node SimulatedDevice.js* (7)

El resultado de la ejecución da el siguiente resultado 10, que como se puede observar hace referencia a los mensajes enviados por el dispositivo.

### 3.0.5 Lectura de los datos de telemetría procedentes de su instancia de IoT Hub

Pero no solo hay que mandar datos, también hay que ser capaz de leerlos. En este apartado se explica como crear un proceso que tome los datos del dispositivo, desde IoT Hub, y los lea. Para ello hace falta otra instancia de la terminal local. En

```

10 // Copyright (c) Microsoft. All rights reserved.
11 // Licensed under the MIT license. See LICENSE file in the project root for full license information.
12
13 'use strict';
14
15 // The device connection string to authenticate the device with per: 'iot-hub;DeviceId=MyNodeDevice;SharedAccessKey=My2ay3R4pYyaGT0qZc1UHT588B24GdLlakoU=
16
17 // WORK
18 // For simplicity, this sample sets the connection string to a constant. In a production environment, the recommended approach is to use
19 // an environment variable to make it available to your application
20 // For more on this, see: https://aka.ms/iot-sdk-node
21 // https://docs.microsoft.com/azure/iot-hub/iot-hub-quickstart-security#shared-access-key-authentication
22
23 // Using the Azure CLI
24 // 1. Generate a connection string: az iot-hub device-id generate --device-id MyNodeDevice --output table
25 // 2. Using the Node.js Device SDK for IoT Hub:
26 // https://github.com/Azure/azure-iot-sdk-node
27 // The sample connects to a device specific MQTT endpoint on your IoT Hub.
28 var mqtt = require('azure-iot-device-mqtt').Mqtt;
29 var DeviceClient = require('azure-iot-device').Client;
30 var Message = require('azure-iot-device').Message;
31
32 var client = DeviceClient.fromConnectionString(connectionString, mqtt);
33
34 // create a message and send it to the IoT hub every second
35 setInterval(function() {
36   // Simulate telemetry
37   var temperature = 30 + (Math.random() * 15);
38   var message = new Message(3500, stringify({
39     temperature: temperature,
40     humidity: 60 + (Math.random() * 20)
41   }));
42
43   // Add a custom application property to the message.
44   // The IoT Hub will filter on these properties without access to the message body.
45   message.properties.add('temperatureAlert', (temperature > 30) ? 'true' : 'false');
46   console.log('Sending message: ' + message.getData());
47
48   // Send the message
49   client.sendEvent(message, function (err) {
50     if (err) {
51       console.error('send error: ' + err.toString());
52     } else {
53       console.log('message sent');
54     }
55   }, 1000);
56 }, 1000);

```

Figure 8: Cadena de conexión del dispositivo.

```

PS C:\Users\Pablo\Documents\Git\APACloudComputing\IoT\azure-iot-samples-node-master\iot-hub\Quickstarts\simulated-device> npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
added 156 packages from 156 contributors and audited 537 packages in 7.426s
found 0 vulnerabilities

```

Figure 9: Instalación de paquetes necesarios para la ejecución.

este caso, en lugar de la ruta *simulated-device*, se accede a la ruta *iot-hub Quickstartd2c-messages* 11.

Al igual que en el caso de la escritura de datos, en la lectura de datos es necesario especificar en el fichero *ReadDeviceToCloudMessages.js*. En este caso el contenido de *connectionString* debe sustituirse 12 por lo obtenido en la cadena de conexión de servicio 5.

Una vez más, debe ejecutarse 6, seguido de 8.

$$nodeReadDeviceToCloudMessages.js \quad (8)$$

El resultado de la ejecución del comando anterior (8) puede observarse en la captura



```
PS C:\Users\Pablo\Documents\Git\APA\CloudComputing\IoT\azure-iot-samples-node-master\iot-hub\Quickstarts\simulated-device> node SimulatedDevice.js
Sending message: {"temperature":51.27881151051054,"humidity":67.51744722957876}
Sending message: {"temperature":54.271148375843864,"humidity":74.8323120242681}
message sent
Sending message: {"temperature":23.8895404809372,"humidity":64.91808133280941}
message sent
Sending message: {"temperature":30.07347526515331,"humidity":73.56948434215266}
message sent
Sending message: {"temperature":20.6303656744054,"humidity":71.21886179078297}
message sent
Sending message: {"temperature":30.27066592337431,"humidity":60.07739696216528}
message sent
Sending message: {"temperature":29.45726461255355,"humidity":63.96204984272126}
message sent
Sending message: {"temperature":31.859264127301046,"humidity":72.46268021890698}
message sent
Sending message: {"temperature":28.895931932716465,"humidity":69.1056764244689}
message sent
Sending message: {"temperature":27.60104528049302,"humidity":70.98179933676631}
message sent
Sending message: {"temperature":29.49388547514831,"humidity":68.25875620154417}
message sent
Sending message: {"temperature":27.075766281379934,"humidity":75.21417484087345}
message sent
Sending message: {"temperature":23.88796216670538,"humidity":77.13934215165939}
message sent
Sending message: {"temperature":23.009380692251703,"humidity":73.21025657359193}
message sent
Sending message: {"temperature":24.49733784178272,"humidity":65.8295694997492}
message sent
Sending message: {"temperature":20.27543253706848,"humidity":79.54467474421627}
message sent
Sending message: {"temperature":25.71134662532206,"humidity":73.62023086257943}
message sent
Sending message: {"temperature":31.045686617664185,"humidity":65.14874066327624}
message sent
Sending message: {"temperature":29.020821369465466,"humidity":74.04387315578938}
message sent
Sending message: {"temperature":29.20751480551222,"humidity":71.62139808785884}
message sent
Sending message: {"temperature":27.30881659300963,"humidity":77.53572694555112}
message sent
Sending message: {"temperature":27.121631352928446,"humidity":60.69755929520702}
message sent
Sending message: {"temperature":34.523847238040514,"humidity":61.33894184576938}
```

Figure 10: Conexión con el dispositivo.

```
PS C:\Users\Pablo\Documents\Git\APA\CloudComputing\IoT\azure-iot-samples-node-master\iot-hub\Quickstarts\read-d2c-messages> ls
Sending message: {"temperature":27.3554562,"humidity":68.54995119}
message sent
Directory: C:\Users\Pablo\Documents\Git\APA\CloudComputing\IoT\azure-iot-samples-node-master\iot-hub\Quickstarts\read-d2c-messages
message sent
Sending message: {"temperature":27.4509485,"humidity":69.6854995119}
Mode LastWriteTime Length Name
-----
14-Mar-19 4:09 AM 1035 ReadDeviceToCloudMessages.js
14-Mar-19 4:09 AM 674 package.json
14-Mar-19 4:09 AM 1035 readme.md
Sending message: {"temperature":23.8851211,"humidity":61.33894184576938}
SERVICE CONNECTION STRING
```

Figure 11: Lectura de datos del dispositivo.

```
PS C:\Users\Pablo\Documents\Git\APA\CloudComputing\IoT\azure-iot-samples-node-master\iot-hub\Quickstarts\read-d2c-messages> ls
Sending message: {"temperature":27.3554562,"humidity":68.54995119}
message sent
Directory: C:\Users\Pablo\Documents\Git\APA\CloudComputing\IoT\azure-iot-samples-node-master\iot-hub\Quickstarts\read-d2c-messages
message sent
Sending message: {"temperature":27.4509485,"humidity":69.6854995119}
Mode LastWriteTime Length Name
-----
14-Mar-19 4:09 AM 1035 ReadDeviceToCloudMessages.js
14-Mar-19 4:09 AM 674 package.json
14-Mar-19 4:09 AM 1035 readme.md
Sending message: {"temperature":23.8851211,"humidity":61.33894184576938}
SERVICE CONNECTION STRING
```

Figure 12: Cadena de conexión a servicio.

## Amazon Web Service

## 4 Seguridad

## 5 Web & Logic Apps

