

PRÁCTICA 1 FUENTES DE DATOS Y APROVISIONAMIENTO: IOT

Lucian Iacob

A lo largo de esta práctica hemos utilizado diversas funcionalidades del catálogo de IBM Cloud para conectar un aparato Android (teléfono móvil en mi caso) y obtener datos del acelerómetro que lleva incorporado.

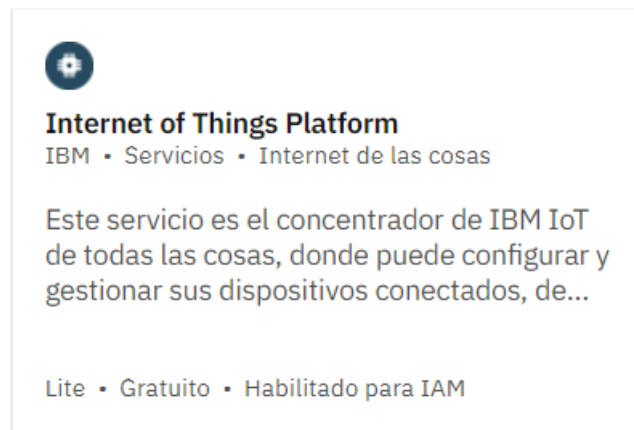
La práctica se divide en tres partes principales:

- Parte Básica
- Parte Intermedia
- Parte Avanzada

A continuación mostraremos capturas de pantalla del proceso seguido:

Parte Básica

Comenzamos creando una plataforma para IoT, esta será nuestro componente principal. Nos permitirá comunicarnos y consumir datos de los dispositivos conectados permitiendo supervisión y análisis en tiempo real mediante paneles de control.

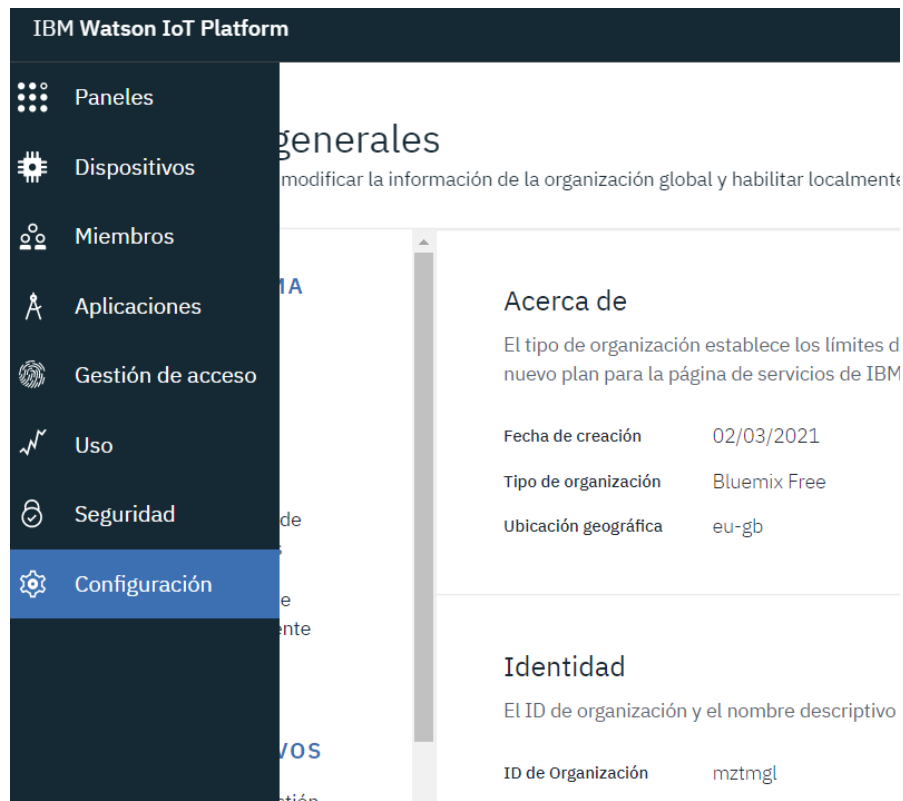


Tras crear la instancia siguiendo los pasos de la guía, la lanzamos:



Ahora accedemos a nuestra instancia, se nos creará una organización (muy importante prestar atención al nombre de esta pues será a lo que conectemos los dispositivos).

Podemos ver que tenemos acceso a paneles de monitorización, a los dispositivos que tenemos registrados, a los miembros que pueden acceder a la plataforma así como a elementos de configuración y seguridad.



A continuación, conectamos nuestro dispositivo Android a la plataforma. Para ello primero creamos un tipo de dispositivo (Android) que se vaya a conectar a la plataforma y segundo conectamos nuestro teléfono como tal. Se nos da la opción de añadir metadatos (número de serie, modelo, fabricante, etc.) pero para lo que vamos a hacer no nos es necesario.

Añadir dispositivo

Identidad

Información del dispositivo

Seguridad

Resumen

Seleccione un tipo de dispositivo para el dispositivo que está añadiendo y dé al dispositivo un ID exclusivo.

Tipo de dispositivo	Android
ID de dispositivo	112233445566

En la pestaña de Seguridad hemos de añadir una señal de autenticación para el dispositivo, siguiendo las instrucciones, la nuestra será Token.1234

Nos quedará algo similar a esto:

Añadir dispositivo

✓

Identidad

✓

Información del dispositivo

✓

Seguridad

●

Resumen

Ahora pasamos a las aplicaciones que debemos instalar en el teléfono para que pueda conectar a la plataforma, serán 2:

IoTool–Internet of Things (IoT) sensor platform

IoToolIBM Watson Cloud

La interfaz de la aplicación nos muestra 3 gráficas temporales para los 3 ejes en los que mide el acelerómetro. En la configuración de la app, rellenaremos los campos necesarios para conectarlo a la plataforma Cloud que hemos creado, recalcar que la señal de seguridad que indicamos previamente será la contraseña de esta configuración:



The screenshot shows the 'Cloud' settings screen in the IoTool app. At the top, there's a back arrow and the text 'Cloud'. Below that, there's a section titled 'Install' with the text 'Install a new cloud extension'. Underneath, there's a section titled 'Cloud settings' with the following options: 'Use cloud' (checked), 'Select cloud service' (Set to: IBM Watson IoT), 'IBM Watson IoT Settings', 'Send data' (checked), and 'Sync after session' (unchecked). The 'Send data' option has a subtext: 'Uncheck to view data from database on IoTool Cloud'. The 'Sync after session' option has a subtext: 'Start synchronization after session is stopped, to transfer any remaining data'.

Tras esto volvemos a nuestra plataforma Watson IoT. Si ahora vamos a ver la información sobre el dispositivo que acabamos de conectar podremos ver sus sucesos recientes. Si en el teléfono le damos al play para registrar movimientos del teléfono, en la plataforma veremos lo siguiente:

Sucesos recientes

Los sucesos recientes listados muestran la corriente activa de datos que entran y salen en este dispositivo.

Suceso	Valor	Formato	Último recibido
accel	{"d":{"AccelerometerX@StarterSensor":3.46939...	json	hace unos segundos
accel	{"d":{"AccelerometerX@StarterSensor":0.52190...	json	hace unos segundos
accel	{"d":{"AccelerometerX@StarterSensor":5.57151...	json	hace unos segundos
accel	{"d":{"AccelerometerX@StarterSensor":2.51909...	json	hace unos segundos
accel	{"d":{"AccelerometerX@StarterSensor":9.15397...	json	hace unos segundos

Vemos como tenemos sucesos provenientes del acelerómetro en formato json en tiempo real. Si hacemos click en uno de los sucesos podemos ver la información completa que trae:

Nombre de suceso accel

Hora de recepción 2 de mar. de 2021 20:30

```
1 {
2   "d": {
3     "AccelerometerX@StarterSensor": 3.4693970680236816,
4     "AccelerometerY@StarterSensor": 5.783944129943848,
5     "AccelerometerZ@StarterSensor": 7.491604328155518
6   }
7 }
```

Ya hemos comprobado que el teléfono se ha conectado correctamente. Ahora podemos proceder a crear paneles de visualización de estos datos:

Crear tarjeta

Tipo de tarjeta
Seleccionar tipo de tarjeta

Dispositivos

Visualización genérica

Gráfico de líneas

Gráfico de barras

Diagrama de anillo

Valor

Indicador

Semáforo

Propiedades de dispositi...

Todas las propiedades ...

Lista de dispositivos

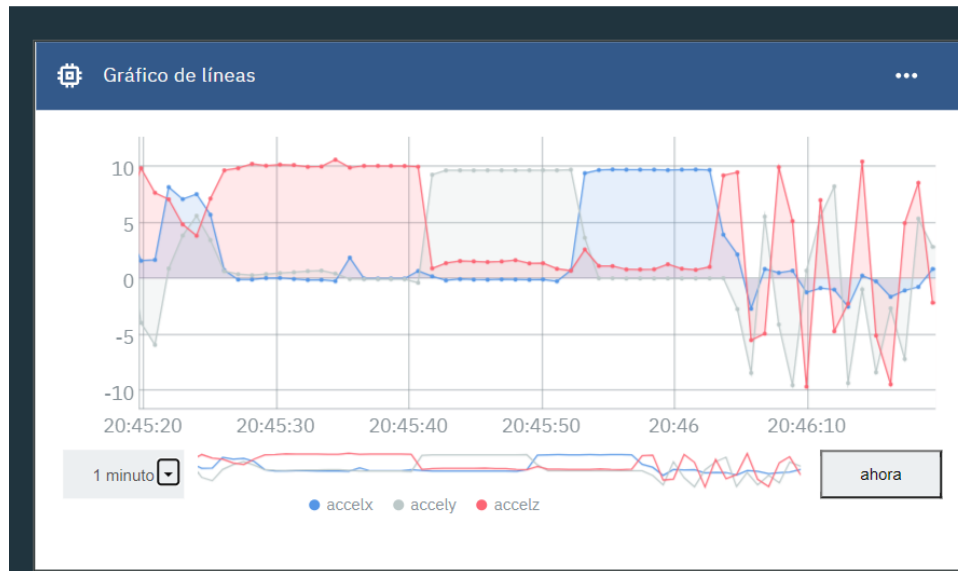
Información de dispositi...

Correlación de dispositi...

Uso

Como nosotros deseamos visualizar unos datos que varían en el tiempo, lo mejor que podemos hacer es elegir un gráfico de líneas. Rellenaremos la información necesaria para crear este panel siguiendo la guía. En esta parte lo que hacemos es crear un modelo de datos diciéndole qué tipos de datos va a recibir (Valores float para cada eje de movimiento) y de qué dispositivo. Tras esto y registrar datos durante cierto tiempo, lo que veremos es esto:

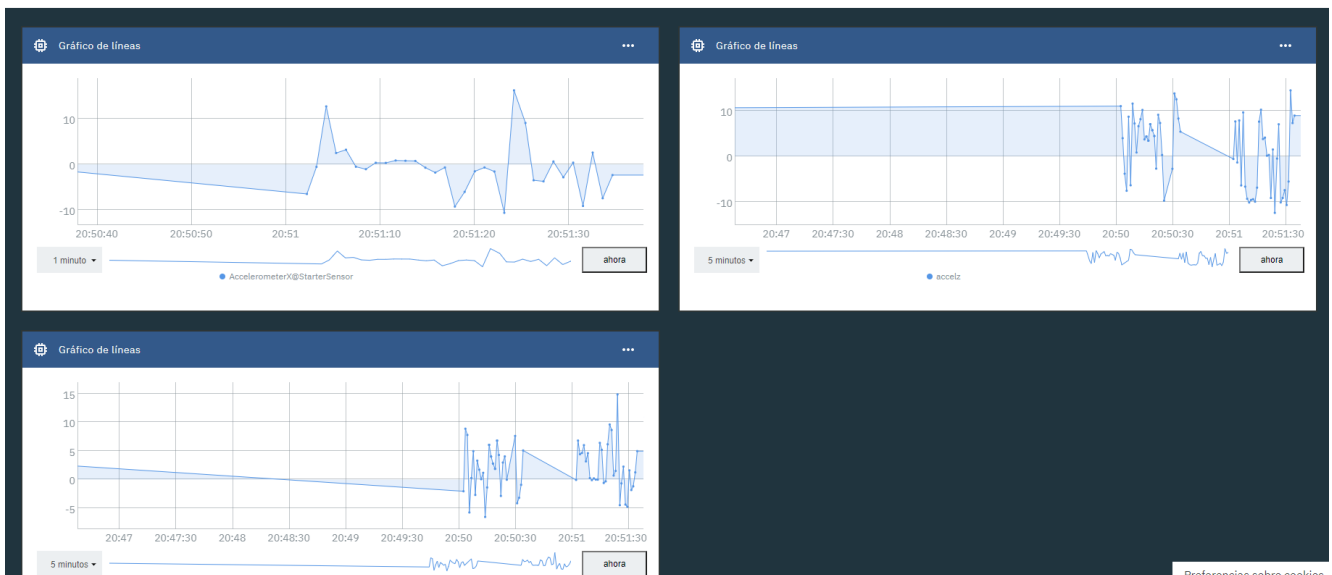
< board-iot-il



Vemos una línea temporal para los 3 ejes, si esto nos parece muy lioso, podríamos crear 3 gráficos de líneas una para cada eje:

< board-iot-il

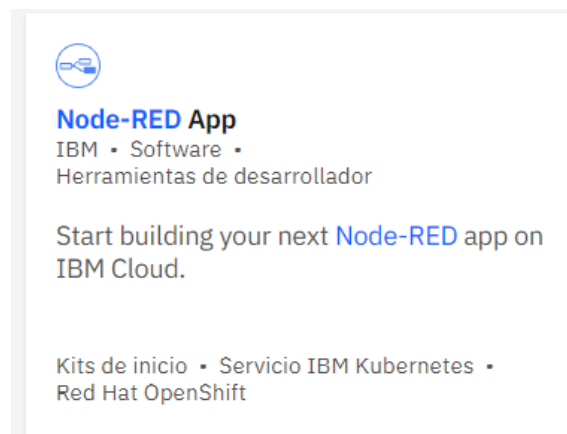
+ Añadir nueva tarjeta Configuración



Hasta aquí abarcaría la parte básica de nuestra práctica, pasemos a la intermedia:

Parte Intermedia

Ya tenemos el teléfono conectado a la plataforma, pero no tenemos ningún medio para controlar/manipular el workflow bajo el que se rigen los datos. Aquí entra en juego la app Node-RED. Una herramienta de programación para enlazar hardware, APIs y servicios online de múltiples maneras con una interfaz muy sencilla de point and click prácticamente. En el catálogo de IBM Cloud aparecerá como:



Creamos la instancia siguiendo los pasos de la guía y procedemos a conectarla con la plataforma IoT que hemos creado previamente. Lo veremos más adelante, pero podemos ir adelantando que la instancia creada viene relacionada con la base de datos no relacional en la nube Cloudant.

Una vez lanzada la instancia Node-RED, toca relacionarla con la plataforma IoT ("Connect existing Service"). Con ello procederemos al lanzamiento de la app:

Detalles

URL de aplicación	Primero debe desplegar su aplicación
Fuente	Descargar código
Grupo de recursos	Default
Destino de despliegue	Primero debe desplegar su aplicación
Creado	2/3/2021

Servicios

Cloudant

[Abrir panel de control](#) [Documentación](#) [Referencia de API](#)

Credenciales ▾

Internet of Things Platform

[Abrir panel de control](#) [Documentación](#)

Credenciales ▾

[Conectarse a servicios existentes](#) + [Crear servicio](#) +

Automatización de despliegue

Configurar Entrega continua

La Entrega continua no está habilitada para esta aplicación. Habilite la Entrega continua para automatizar compilaciones, pruebas y despliegues con Delivery Pipeline, GitLab, etc.

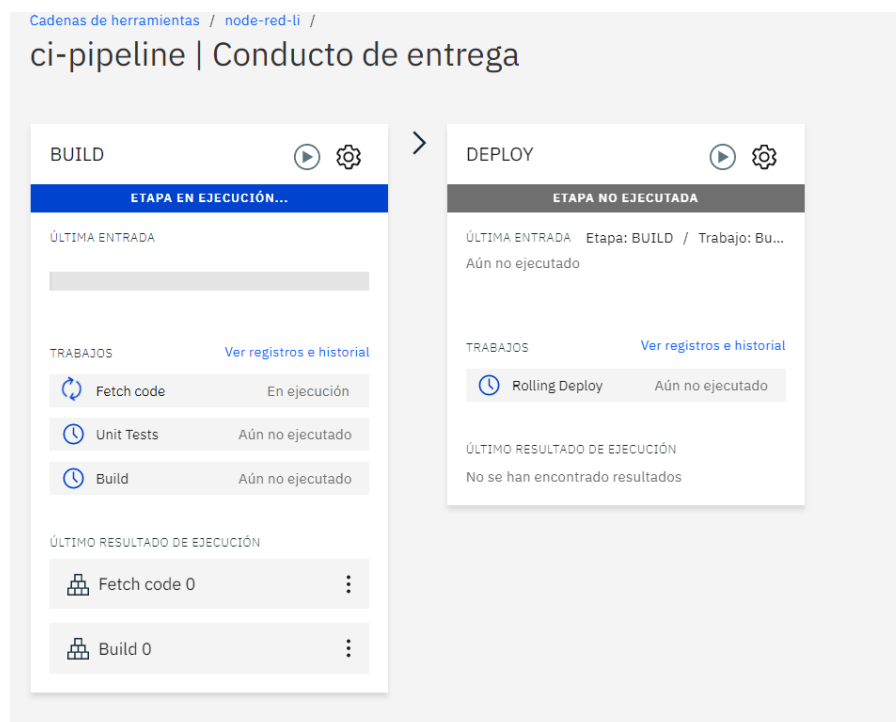
[Desplegar su aplicación](#)

El despliegue lo realizaremos mediante la plataforma como servicio (PaaS) de código abierto Cloud Foundry, que es la que soporta el servicio IBM Bluemix, la cual engloba varios lenguajes de programación y servicios así como la metodología de desarrollo DevOps de forma integrada para crear, ejecutar, desplegar y gestionar aplicaciones en la nube. Esto nos permitirá habilitar lo que se llama “continuous delivery”.

Para este despliegue necesitaremos crear una llave API.

The screenshot shows the 'Destino de despliegue' (Deployment Destination) section of the IBM Cloud console. It features four cards for different deployment targets: 'Servicio Kubernetes', 'Red Hat OpenShift', 'Cloud Foundry' (which is selected), and 'Code Engine'. Below these cards, there is a field for the 'Clave de API de IBM Cloud' (IBM Cloud API Key) with a 'Nuevo' (New) button. Further down, the 'Número de instancias' (Number of instances) is set to 1. A memory allocation slider is shown, ranging from 64 MB to 2000 MB, with a current value of 256 MB. At the bottom, there are dropdown menus for 'Región' (Region) set to 'Londres', 'Organización' (Organization) set to 'lucian.iacob@estudiante.uam.es', and 'Espacio' (Space) set to 'dev'.

Tras lanzarla, comenzará un proceso que tardará unos minutos para la creación de esta y realizar sus correspondientes test. Lo que veremos si miramos la pipeline de ejecución será esto:



Tras completarse la build, tendremos una URL disponible para Node-RED. Si accedemos a él se nos redirigirá ahí. Primero se nos pedirá crear una cuenta:

Secure your Node-RED editor

☒ Secure your editor so only authorised users can access it

Username

lucianiacob

Password

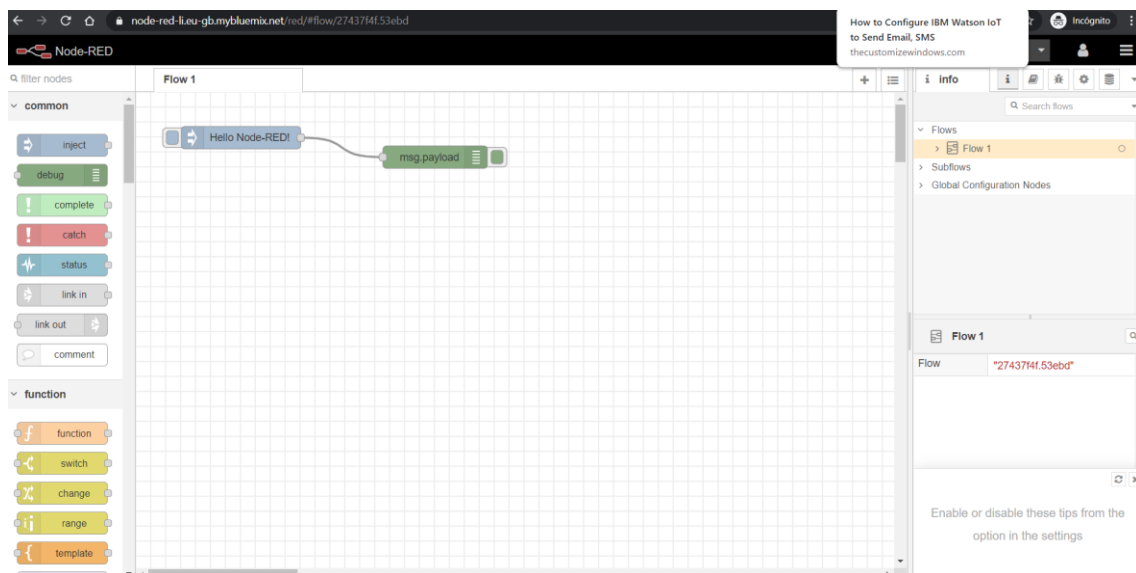
.....

good

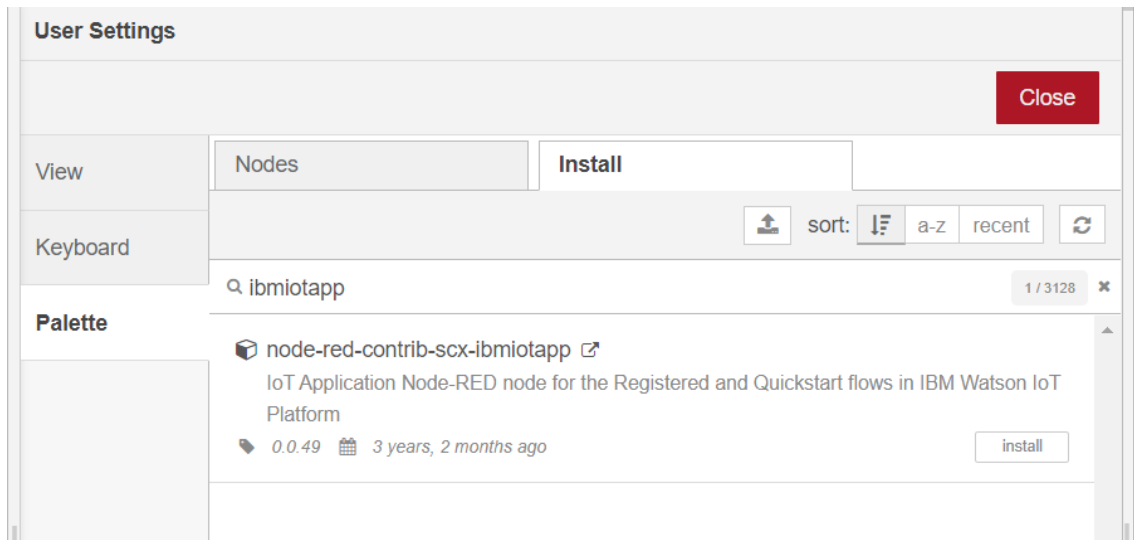
☐ Allow anyone to view the editor, but not make any changes

☐ *Not recommended:* Allow anyone to access the editor and make changes

Con esta cuenta accederemos al editor de workflows de Node-RED:



Aquí es donde crearemos el workflow deseado con los distintos elementos que queramos. Primero deberemos instalar la extensión de IBM-IoT dedicada a Node-RED:



Con esta extensión, tras configurarla, podremos controlar el flujo de datos proveniente de nuestro móvil a través de la plataforma creada al inicio.

La configuración de esta extensión tiene que coincidir con la de los elementos anteriores:

Edit ibmiot in node

Delete Cancel Done

Properties

Authentication: Bluemix Service

Input Type: Device Event

Device Type: All or Android

Device Id: All or 112233445566

Event: All or accel

Format: All or json

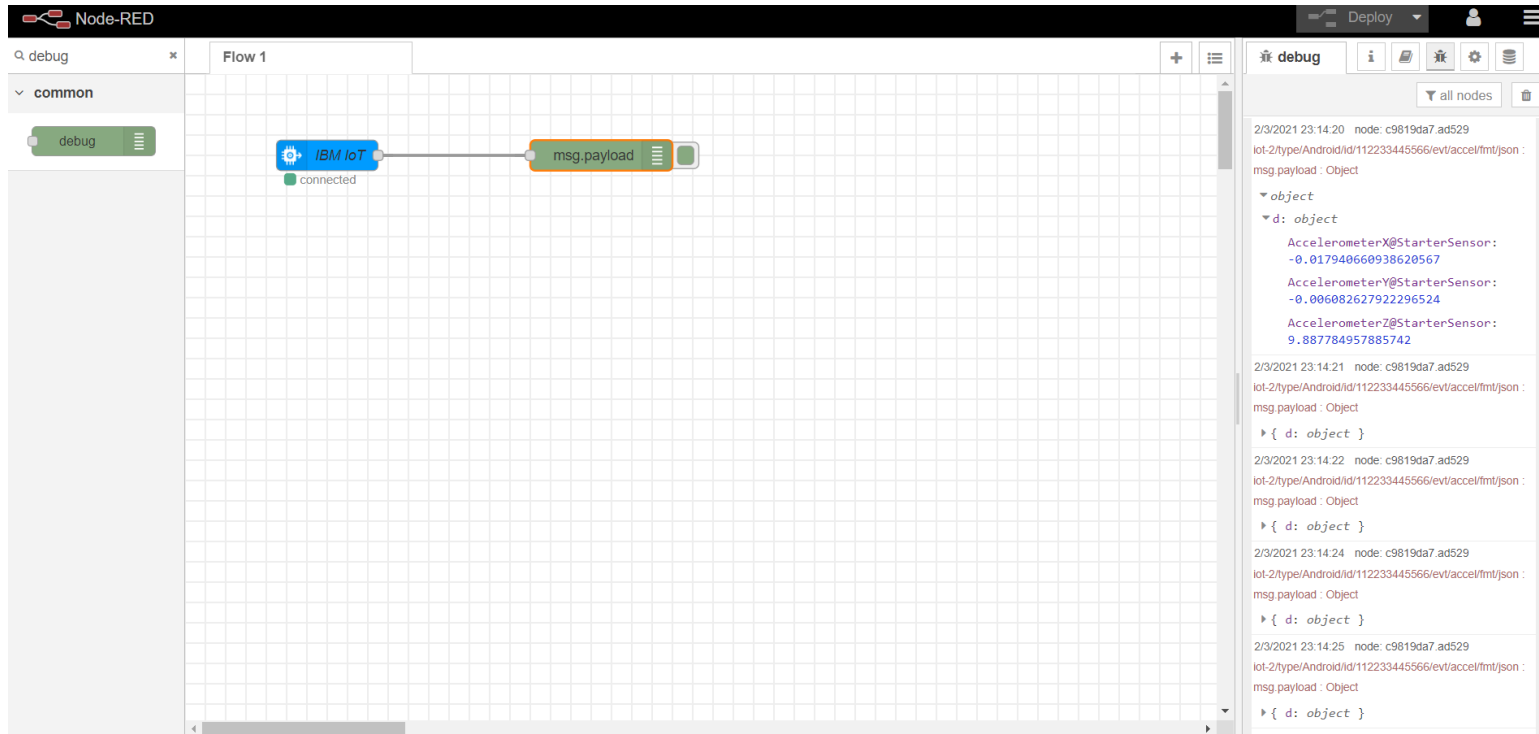
QoS: 0

Name: IBM IoT

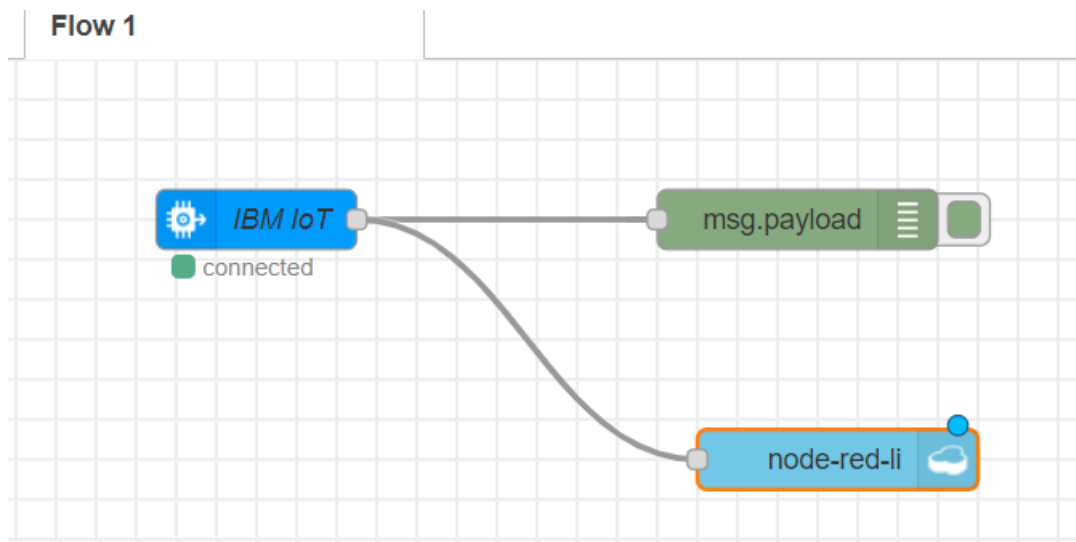
Service: registered

Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to IoT Applications
Check the info tab, to get more information about each of the fields

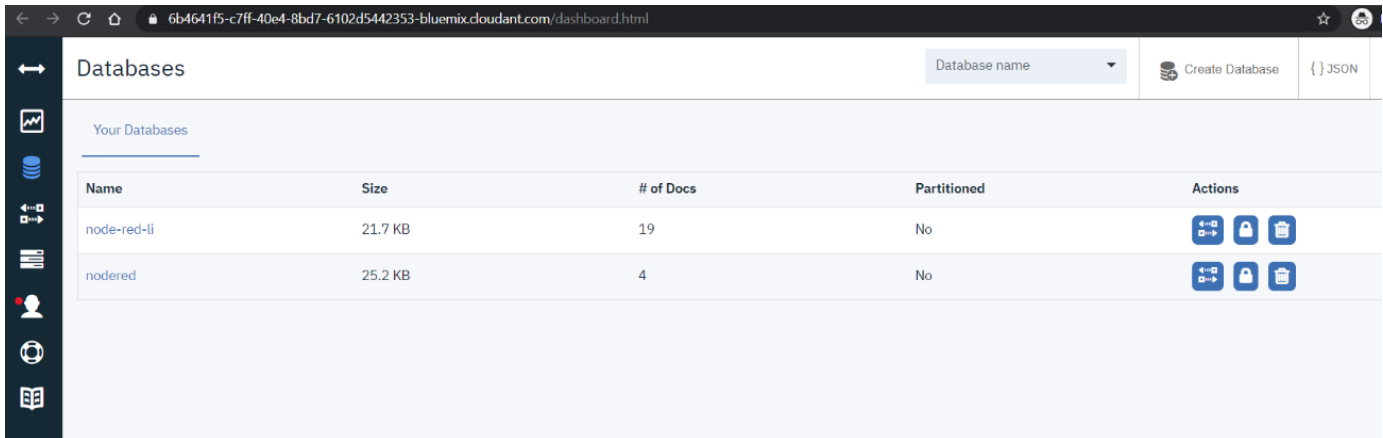
Nuestro workflow inicial será la conjunción de esta extensión con un módulo debug que nos muestra por pantalla mensajes tal y como se generan con los 3 valores X, Y, Z:



Ya tenemos una fuente de datos conectada de manera que podemos recibirlos y visualizarlos, el paso siguiente es persistirlo. Para ello utilizaremos la base de datos previamente mencionada Cloudant. Gracias a Node-RED, este proceso es muy sencillo pues simplemente tenemos que añadir la extensión del mismo nombre al gráfico quedándonos tal que así:



Nos basta con indicar un nombre a la base de datos ("node-red-li" en mi caso). Si ahora accedemos a la consola de Cloudant desde el catálogo de servicios que estamos utilizando podremos ver la base de datos generada en la que iremos archivando los datos generados por el teléfono:



The screenshot shows the IBM Cloud Databases dashboard. At the top, there's a search bar for "Database name" and a "Create Database" button. Below this, a table lists the databases:

Name	Size	# of Docs	Partitioned	Actions
node-red-li	21.7 KB	19	No	[Icons for edit, lock, delete]
nodered	25.2 KB	4	No	[Icons for edit, lock, delete]

Haciendo click en “node-red-li” tendremos el listado de todas las entradas, si hacemos click en una veremos lo siguiente:



The screenshot shows the document view for the database 'node-red-li'. At the top, there's a breadcrumb navigation: 'node-red-li > 179f6149cb81eb78e9622b1084b95326'. Below this, there are buttons for 'Save Changes' and 'Cancel'. The main area displays a JSON document with the following structure:

```

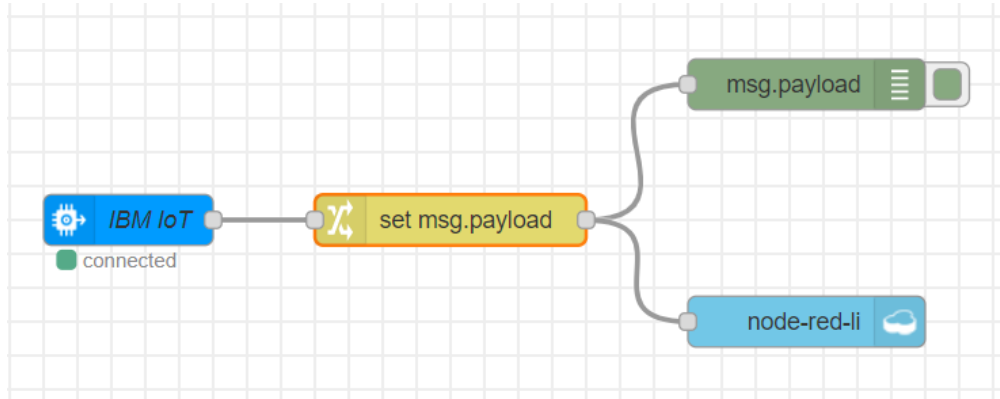
1 {
2   "_id": "179f6149cb81eb78e9622b1084b95326",
3   "_rev": "1-4bf391aa70d7fa8f2c68d659ab268c51",
4   "topic": "iot-2/type/Android/id/112233445566/evt/accel/fmt/json",
5   "payload": {
6     "d": {
7       "AccelerometerX@StarterSensor": -1.6521673202514648,
8       "AccelerometerY@StarterSensor": 6.164851665496826,
9       "AccelerometerZ@StarterSensor": 5.87041711807251
10    }
11  },
12  "deviceId": "112233445566",
13  "deviceType": "Android",
14  "eventType": "accel",
15  "format": "json"
16 }
```

Tenemos el id, el topic al que pertenece, los datos como tal en los 3 ejes y sus metadatos como el ID del dispositivo, su tipo, y el tipo de evento que estamos recibiendo.

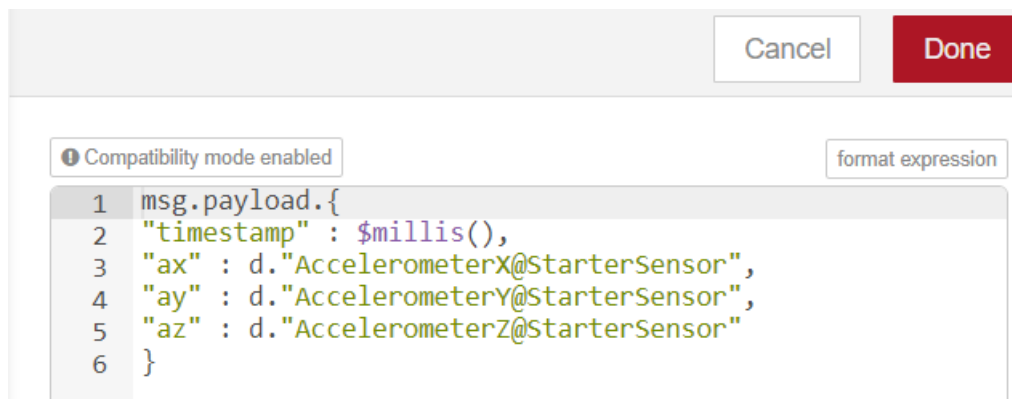
Ya tenemos un lugar donde guardar los datos que nos llegan continuamente del teléfono. Con todo esto ya establecido, pasamos a la parte avanzada de la práctica.

Parte Avanzada

Vamos a añadirle complejidad a nuestro flujo de datos añadiendo una extensión que nos cambie el formato del mensaje que manda el teléfono (set.msg.payload)



Usamos este plugin intermedio para que la salida se nos vea de la siguiente manera:



Si miramos la salida del módulo comprobamos que ha tenido efecto el cambio:

```
3/3/2021 19:43:03 node: c9819da7.ad529
iot-2/type/Android/id/112233445566/evt/accel/fmt/json :
msg.payload : Object
  { timestamp: 1614796982673, ax:
    -1.6665318012237549, ay:
    6.308497905731201, az:
    4.022172927856445 }

3/3/2021 19:43:04 node: c9819da7.ad529
iot-2/type/Android/id/112233445566/evt/accel/fmt/json :
msg.payload : Object
  { timestamp: 1614796983736, ax:
    -5.518636703491211, ay:
    -1.7165210247039795, az:
    10.708891868591309 }
```

Ya tenemos la persistencia del dato y el dato en un formato deseado, para que todo esto no sea en balde vamos a conectarlo a algún proyecto donde podamos explotarlo. Para esto usaremos Watson Studio, un entorno de colaboración para programación, especialmente orientado a la democratización del machine learning para infundir la IA entre los proyectos en los que estén los usuarios.

Para ello primero debemos crear una comunicación segura entre nuestra base de datos Cloudant y Watson Studio, esto lo lograremos mediante una API-key. Esto lo logramos creándola en los ajustes de seguridad del servicio Cloudant:

node-red-li-cloudant-1614718606905 Activo Añadir etiquetas Detalles Acciones...

Gestionar

Credenciales de servicio

Plan

Conexiones

Credenciales de servicio

Puede generar un nuevo conjunto de credenciales para casos en los que quiera conectarse manualmente una app o un consumidor externo a un servicio de IBM Cloud. [Más información](#)

Buscar credenciales...

Nueva credencial +

Nombre de clave	Fecha de creación
<input checked="" type="checkbox"/> 108334b9-3412-4c24-82e4-c30557af4328	2021-03-02 9:57 PM

```
{
  "apikey": "fkXt34TPsJNxW5k51WyQyZ7EM5_450sX4XK1-iEOTXms",
  "host": "6b4641f5-c7ff-40e4-8bd7-6102d5442353-bluemix.cloudantnosqldb.appdomain.cloud",
  "iam_apikey_description": "Auto-generated for key 745800ce-4a46-4c71-bbb5-95d746cfa3c9",
  "iam_apikey_name": "108334b9-3412-4c24-82e4-c30557af4328",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/a488e41b8aa54889a5800f205d8973d2::serviceid:ServiceId-ef3e491b-d29-47a2-9540-3c9cc37940a0",
  "password": "b6c2dc759b40c274d728f59033b280ec",
  "port": 443,
  "url": "https://apikey-v2-5jkg9ds98tmztaic3yec2l16ysp2d11tb1uajnu8co7:b6c2dc759b40c274d728f59033b280ec@6b4641f5-c7ff-40e4-8bd7-6102d5442353-bluemix.cloudantnosqldb.appdomain.cloud",
  "username": "apikey-v2-5jkg9ds98tmztaic3yec2l16ysp2d11tb1uajnu8co7"
}
```

Vemos aquí el valor de la llave creada automáticamente por el servicio. Hecho esto, pasemos a crear la instancia de Watson Studio, esta es su interfaz principal:

Bienvenido, Lucian!

Watson Studio • Pruebe otras aplicaciones IBM Cloud Pak for Data

Aprender mediante ejemplo

Recorra los pasos de resolución de un problema empresarial específico en un proyecto de ejemplo.

Realizar una guía de aprendizaje guiada

Trabajar con datos

Cree un proyecto para que su equipo prepare datos, busque conocimientos o cree modelos.

Crear un proyecto

Amplíe sus prestaciones

Añada herramientas, bases de datos u otras características creando instancias de servicios.

Crear un servicio

Navegación rápida

Proyectos

Despliegues

Soporte

Documentación

Preguntas más frecuentes

Novedades

Proporcionar comentarios

Visión general

Proyectos recientes

il-ws-project-2021-02-27

27 feb 2021 12:58

Sus servicios

watson-li

Watson Studio

Hoy a las 20:10

Notificaciones

Sin notificaciones

Verá sus notificaciones más recientes aquí.

Espacios de despliegue

No hay espacios de despliegue

Después de crear los espacios, podrá verlos aquí.

Nuevo espacio de despliegue +

Optión

Aquí podemos ver los proyectos que tenemos en marcha, iniciaremos uno desde cero ahora para esta práctica. Al crearlo nos tocará añadir un objeto de almacenamiento en la nube. Nos saldrá una ventana emergente con la instancia que debemos crear:

Cloud Object Storage

Autor: IBM • Fecha de última actualización: Dec 11, 2020 • [Documentos](#) • [Documentos de API](#)

Crear | Acerca de

Plan de precios
Los precios mostrados no incluyen impuestos. Los precios mensuales que se muestran son para el país o región: United States

Plan	Características	Precios
Lite	1 Instancia de servicio COS Almacenamiento de hasta 25 GB/mes. Hasta 20.000 solicitudes GET/mes Hasta 2.000 solicitudes PUT/mes Hasta 10 GB de recuperación de datos/mes Hasta 5GB de salida pública Se aplica al total agregado en todas las clases de depósito de almacenamiento El plan de servicio Lite para el Almacenamiento de objetos en la nube incluye capacidad de recuperación Regional e Interregional, clases de datos flexibles y seguridad incorporada. Los servicios del plan Lite se suprimen tras 30 días de inactividad.	Gratuito
Standard	No hay ninguna tarifa mínima, así que solo paga por lo que usa.	Consulte los detalles sobre los precios

Resumen

Cloud Object Storage
Región: Global
Plan: Lite
Nombre de servicio: Cloud Object Storage-ly
Grupo de recursos: Default

Crear

Tras crear la instancia de Storage, volviendo a la ventana de creación de proyecto Watson, podremos finalizar el proceso.

Con todo creado llegamos al menú de inicio del proyecto de Watson Studio. Para conectar este con Cloudbant vamos a añadirle un notebook:

Proyectos / project-li-3-3-2021

0 Bytes utilizados
Cloud Object Storage

Colaboradores
Lucian Iacob
Administrador
[Ver todo \(1\)](#)

Seleccionar tipo de activo

Tipos de activos disponibles

- Datos
- Conexión
- Datos conectados
- Experimento de AutoAI
- Notebook
- Panel de control
- Modelo de reconocim...
- Modelo de Natural La...
- Modelo desde archivo
- Flujo del modelador
- Flujo de Data Refinery
- Experimento de Deci...

NUEVO

Nuevo notebook

BlancoDesde el archivoDesde URL

Nombre

notebook-li

Descripción (opcional)

Escriba aquí la descripción

Seleccionar entorno de ejecución

Default Spark 2.4 & Python 3.6 (Driver: 1 vCPU 4 GB RAM, 2 Executors: 1 vCPU 4 GB RAM)

El entorno de ejecución seleccionado utiliza 1 controlador con 1 vCPU y 4 GB RAM, y 2 ejecutores cada uno con 1 vCPU y 4 GB RAM. Consume 1,5 unidades de capacidad por hora. [Más información](#) acerca de las horas de unidad de capacidad y los planes de precios de Watson Studio.

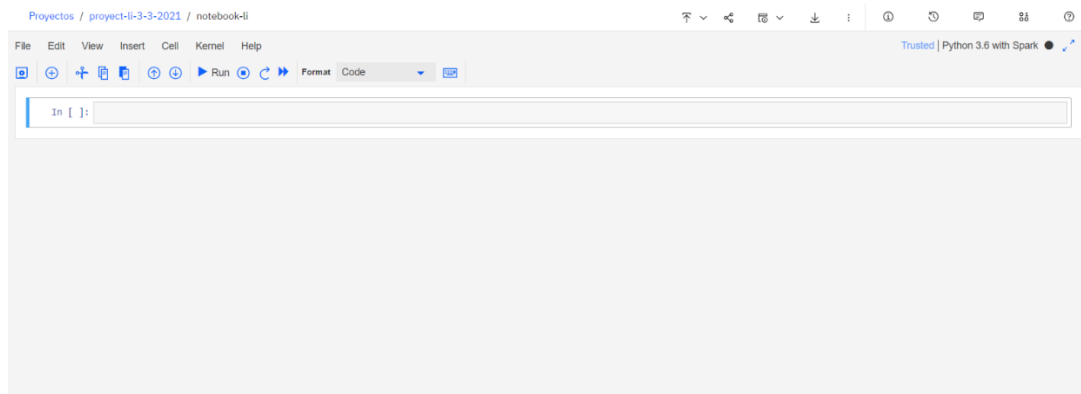
Idioma

☒ Python 3.6

Versión de Spark

☒ 2.4

Seleccionamos un servicio de Python 3.6 con Spark 2.4 para nuestro notebook. Tras crearlo, se nos iniciará una interfaz similar a la de los Jupyter Notebooks a los que estamos ya más que acostumbrados.



Lo primero que importamos es la librería pixiedust. Si vamos a su documentación, se describe como:

“[PixieDust](#) is an open source Python helper library that works as an add-on to Jupyter notebooks to improve the user experience of working with data. It also fills a gap for users who have no access to configuration files when a notebook is hosted on the cloud.”

Esta librería nos permitirá añadir un paquete que nos conectará con Cloudant:

```
In [1]: import pixiedust
pixiedust.installPackage("org.apache.bahir:spark-sql-cloudant_2.11:0")

Waiting for a Spark session to start...
Spark Initialization Done! ApplicationId = app-20210303194256-0000
KERNEL_ID = e5edc0a2-66d9-4da3-963c-d8be5bc84b17
Pixiedust database opened successfully
Table VERSION_TRACKER created successfully
Table METRICS_TRACKER created successfully

Share anonymous install statistics? (opt-out instructions)

PixieDust will record metadata on its environment the next time the package is installed or updated. The data is anonymized and aggregated to help plan for future releases, and records the following values:

{
  "data_sent": currentDate,
  "runtime": "python",
  "application_version": currentPixiedustVersion,
  "space_id": nonIdentifyingUniqueId,
  "config": {
    "repository_id": "https://github.com/ibm-watson-data-lab/pixiedust",
    "target_runtimes": ["Data Science Experience"],
    "event_id": "web",
    "event_organizer": "dev-journeys"
  }
}

You can opt out by calling pixiedust.optOut() in a new cell.

Pixiedust version 1.1.18
```

Ahora importamos SparkSession para crear nuestra sesión de Spark que nos permita trabajar con DataFrames

```
In [2]: from pyspark.sql import SparkSession
        spark = SparkSession.builder.getOrCreate()

In [ ]: def readDataFrameFromCloudant(database):

        cloudantdata = spark.read.format("org.apache.bahir.cloudant")\
        .option("cloudant.host", 'CLOUDANT_HOST')\
        .option("cloudant.username", 'CLOUDANT_USERNAME')\
        .option("cloudant.password", 'CLOUDANT_PASSWORD')\
        .load(database)

        return cloudantdata
```

Creamos una función “readDataFrameFromCloudant” que nos lea los datos de la base de datos de Cloudant. En la foto anterior debemos rellenar los datos de HOST, USERNAME, PASSWORD con la información de la API-key que generamos antes:

```
{
  "apikey": "fkXt34TPsJNxW5k51WyQyZ7EM5_450sX4XK1-iE0TXms",
  "host": "6b4641f5-c7ff-40e4-8bd7-6102d5442353-bluemix.cloudantnosqldb.appdomain.cloud",
  "iam_apikey_description": "Auto-generated for key 745800ce-4a46-4c71-bbb5-95d746cfa3c9",
  "iam_apikey_name": "108334b9-3412-4c24-82e4-c30557af4328",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/a488e41b8aa54889a5800f205d8973d2::serviceid:ServiceId-ef3e491b-d299-47a2-9540-3c9cc37940a0",
  "password": "b6c2dc759b40c274d728f59033b280ec",
  "port": 443,
  "url": "https://apikey-v2-5jpk9ds98tmztaic3yec2116ysp2d11tb1uajnu8co7:b6c2dc759b40c274d728f59033b280ec@6b4641f5-c7ff-40e4-8bd7-6102d5442353-bluemix.cloudantnosqldb.appdomain.cloud",
  "username": "apikey-v2-5jpk9ds98tmztaic3yec2116ysp2d11tb1uajnu8co7"
```

Definida ya la función para leer los datos, la ejecutamos y mostramos por pantalla los datos

```
In [ ]: df = readDataFrameFromCloudant('node-red-li')
        df.createOrReplaceTempView('df')

        display(df)
```


Lo que se nos muestra por pantalla es lo siguiente:

SPARK JOB PROGRESS

Hide All

JOB	PROGRESS	DURATION	STATUS
0	1 stage	6.11 sec	

OptionsShare

Schema

Table

Search table

Showing 100 of 150 rows

_id	_rev	deviceId	deviceType	eventType	format	topic
03dc027d9164ef7e5fb32dc30f4a0271	1-c3dd173ee453218093fd9777ab578f99	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
03dc027d9164ef7e5fb32dc30f4a1b76	1-9221a34cc1a817c35bf75ce0bbf2e043	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
03dc027d9164ef7e5fb32dc30f4a448f	1-467471983b0e81dd77a7c511cde2cb4c	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084b95a08	1-96a030b49ae1936edc59f610ef7bdcea	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bfb4f	1-f4fc0a61b34b61bb7af1fb2199674535	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bcd260	1-1928a0f2f08a2c6fa9ecfbcc1de8e7a8	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bdb25	1-1f56279f3d561d803961d765d312cbf6	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bdb0b1	1-f3a5ff06fdefec25119a40230c7312a1	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bdb934	1-e0d6ae62277fe9aa9a4ca477f4d8d7b	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bdc5d1	1-307a8cad750f38ee71ac49a085c30d2	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bdd842	1-b0eb5d6a3b37effeafb63143d3e7473a	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json
179f6149cb81eb78e9622b1084bde22f	1-738a77de261bd54dc7147d6e85f2f9fc	112233445566	Android	accel	json	iot-2/type/Android/id/112233445566/evt/accel/fmt/json

Vemos cómo tenemos cada uno de los registros que se han guardado en la base de datos. Con esto ya tendríamos finalizada la práctica dedicada a IoT mediante el uso de la nube de IBM.

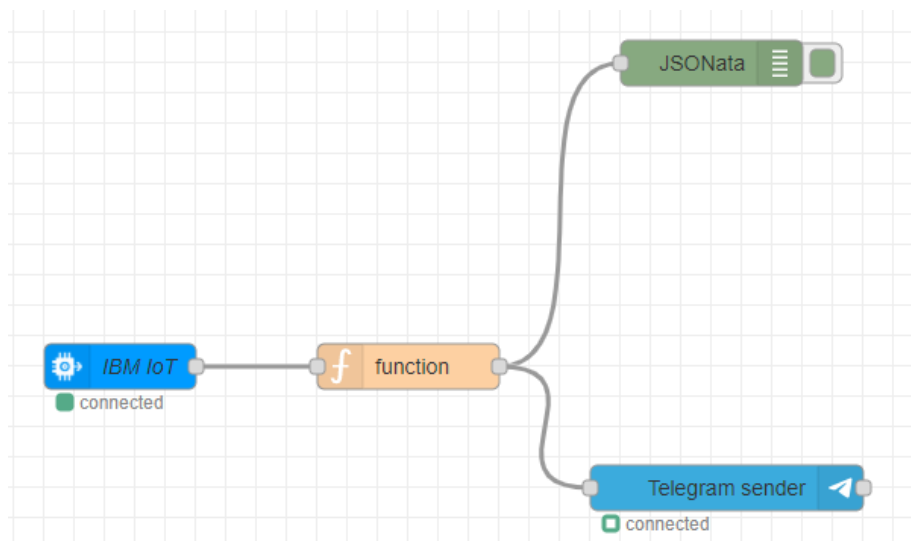
Para recapitular un poco, hemos visto uno de los infinitos ejemplos de andar por casa que podemos crear para tener un primer contacto con el internet de las cosas. En nuestro caso hemos utilizado un dispositivo al que tenemos acceso todos hoy en día como un teléfono y lo hemos conectado a una plataforma IoT de manera que hemos definido nuestra fuente de datos, luego hemos utilizado Node-RED como herramienta para controlar el flujo de trabajo, indicando el formato deseado de los datos y la conexión a una base de datos Cloudant para persistirlo. Por último lo hemos conectado a un proyecto de tipo notebook para poder trabajar con el dato ya persistido con el que podríamos hacer la explotación deseada para nuestro estudio.

Pero las opciones son incontables, tenemos servicios de gran cantidad de empresas (AWS, Azure, etc) y por supuesto infinidad de dispositivos que nos puedan servir como fuentes de datos, desde la lavadora de nuestra casa que podemos monitorizar para saber si está en marcha hasta la detección de llegada de camiones a un silo mediante mini detectores que conectemos en la entrada.

Ampliación

Por ampliar un poco esta práctica, por jugar un poco con ello, he decidido cambiar el workflow en node red para conectar la salida con un bot que he creado en la app de Telegram, de manera que en la conversación con ese bot nos lleguen los valores de salida del acelerómetro.

Nuestro workflow ahora se verá de la siguiente forma:



La fuente de datos sigue siendo el plugin de IBM IoT, a continuación necesitamos una función para transformar los datos al formato que necesita la extensión de Telegram para poder mandar mensajes:

```
1 msg.payload.chatId = 932398392;
2 msg.payload.type = "message";
3 msg.payload.content = JSON.stringify(msg.payload.d);
4 var d = new Date();
5 msg.payload.date = d;
6 delete msg.payload.d;
7 return msg;
```

Necesitamos pasarle el ID del chat que tengamos con el bot creado para que sepa a dónde mandar el mensaje, le indicamos que el tipo de mensaje y además el contenido y fecha.

La extensión de debug es simplemente para comprobar que todo ocurre de acuerdo a lo establecido, vamos que el mensaje sale bien:

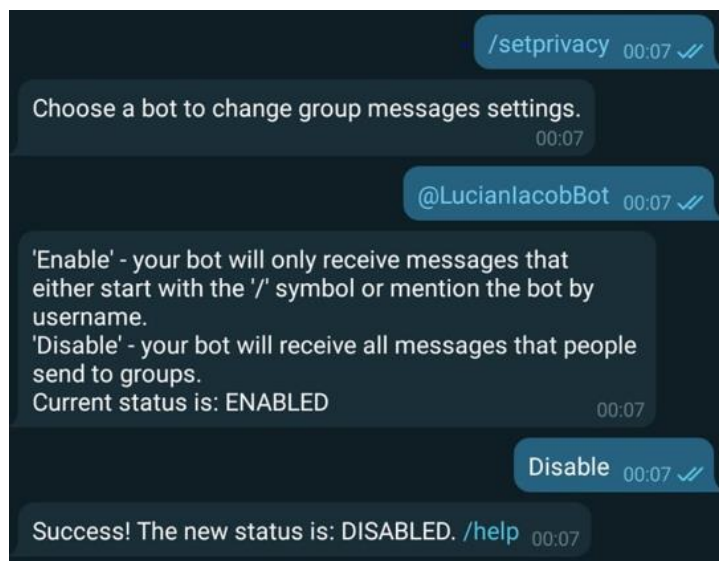
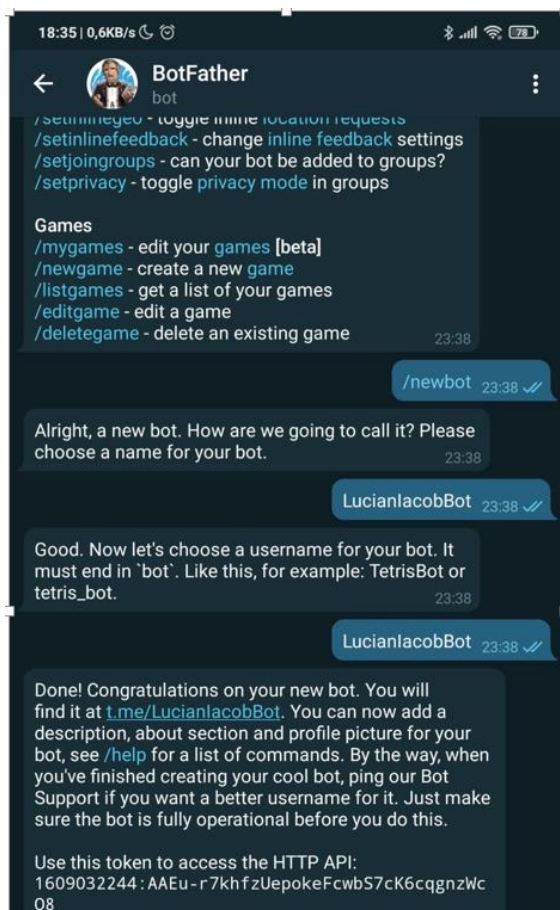


Por último tenemos el plugin de Telegram sender, este se encarga de enviar el mensaje deseado al chat que elijamos a través del bot que le configuremos:

A screenshot of the 'Edit telegram bot node' configuration window in Node-RED. The window has a title bar 'Edit sender node > Edit telegram bot node' and buttons for 'Delete', 'Cancel', and 'Update'. Below the title bar is a 'Properties' section with a settings icon and a document icon. The configuration fields are: 'Bot-Name' (LucianIacobBot), 'Token' (1609032244:AAEu-r7khfzUepokeFcwbS7cK6cqgnzWcO8), a tip box stating 'Tip: If you don't have a token yet, you can create a new one here: @BotFather', 'Users' (Lucius1998), 'ChatIds' (932398392), 'Server URL' (Optional URL for proxying and testing e.g.: https://api.telegram.org), and 'Update Mode' (Polling). The 'Update Mode' is a dropdown menu.

Creamos un bot llamado LucianIacobBot, al crearlo nos dan un token que debemos introducir para temas de autenticación. Indicamos los users de Telegram que pueden recibir comunicaciones de nuestro dispositivo IoT a través de ese bot y el ID del chat creado entre nosotros y el bot.

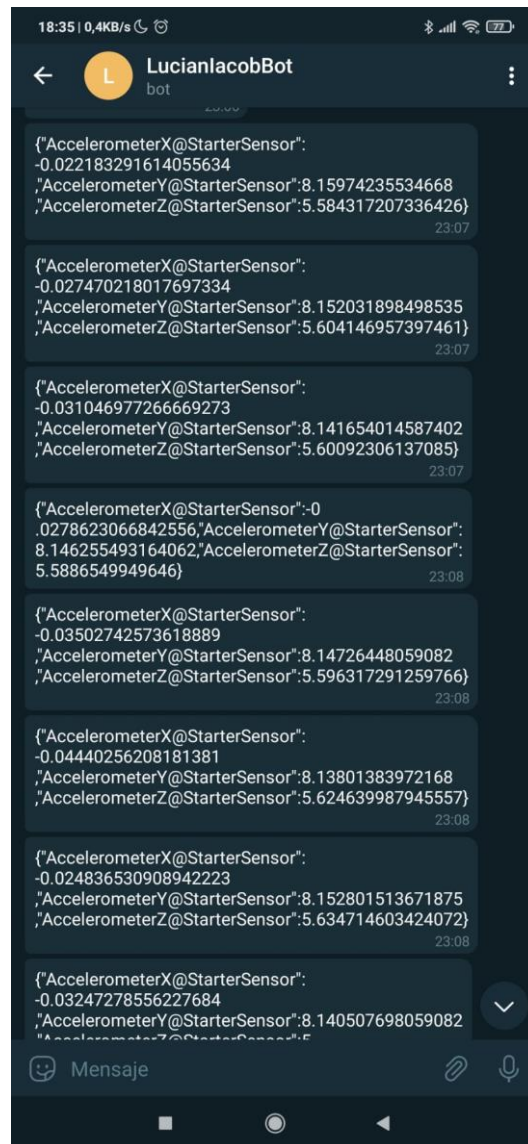
Para crear el bot hemos seguido los siguientes pasos:



Hemos utilizado el “BotFather” que nos permite crear bots mediante el comando /newbot. Lo llamamos LucianIacobBot cómo habíamos visto antes. Vemos que al final nos dan el token que utilizamos en Node-RED

Un punto importante es cambiar la privacidad del bot para que pueda recibir mensajes por si queremos comunicarnos con él.

Con todo esto montado, si lanzamos la build y activamos la medición de datos desde el teléfono vemos cómo nos empiezan a llegar mensajes en la conversación:



No se ha añadido ningún tipo de filtro ni comando por lo que los mensajes se reciben cada segundo provocando una masiva llegada de mensajes al teléfono, pero hemos conseguido establecer la comunicación que es lo principal.

Es cierto que este caso no tiene sentido porque nuestro origen de los datos es a su vez el destino porque usamos Telegram desde el mismo móvil del medimos el acelerómetro. Pero bien podríamos utilizar Telegram desde el ordenador para vigilar si nuestro teléfono se está moviendo tras añadir un filtro que nos mande mensajes cuando el teléfono se esté moviendo habiéndolo dejado en algún lugar donde nadie debería tocarlo.

Esta es solo una opción más de las infinitas que nos da el internet de las cosas. Por último me gustaría añadir una imagen de la propia documentación de IBM Watson IoT Platform que es un resumen de gráfico del esquema seguido en esta práctica y creo que lo aclara todo un poco más.

