ISSN NO. 2320-5407

*Journal homepage: http://www.journalijar.com*

*INTERNATIONAL JOURNAL OF ADVANCED RESEARCH*

**RESEARCH ARTICLE**

# National Symposium On Emerging Trends In Computing & Informatics, NSETCI 2016, 12th July 2016,Rajagiri School of Engineering & Technology, Cochin, India.

## Study of different IoT based cloud services using a raspberry pi application.

***Vivek Joy and Divya James.**

Department of Information Technology, Rajagiri School of Engineering & Technology, India

*Manuscript Info*

***Key words:***
Aws IoT,
IBM IoT,
Raspberry pi,
Mqtt,
Cloud suites
,

*Abstract*

It is believed that by the 2020 there will be about 20 billion connected devices which is 3 times more than human population in the world. These devices will be connected to each other using the internet protocols which are a set of standard protocols. This has led to the rise of the buzzword Internet of things or IoT. The growing no of devices means the data too grows rapidly. The collection, management and analysis of this data is the major challenge in IoT. The data is the most valuable asset which needs to be filtered and modified to get the correct results. An IoT setup may have a number of sensor networks and data is generated real time, as the number of sensors grow managing it individually becomes difficult also the data analysis and storage requires huge resources. This meant that the data needs to be analysed and processed in the clouds for managing costs. Having sighted this opportunity major cloud companies like Amazon, IBM and Microsoft have come up with their own IoT based suites to manage data generated by IoT devices. Amazon introduced AWS IoT and IBM introduced IBM Watson. This paper aims at studying these suites and comparing and contrasting the services offered by them using a raspberry pi based application called weather station. The application will use a temperature and humidity sensor to sense data and the management and processing of the data will be done at the cloud suites.



## Introduction:-

IoT is a major buzzword nowadays. There are many many companies catering to offer IoT based products and services. IoT is based on the protocols of the already existing internet. IPv4 or IPv6 is the networking protocol used in IoT. But the problem is all the other higher layer protocols are not standardized for IoT. Different companies and programmers use different protocols and techniques for making their IoT ecosystem. A programmer needs to study different technologies and languages if he/she has to develop an IoT application for various devices. Also the devices and sensors in IoT varies significantly in configuration and their operating systems. So there is a need for a gateway for identifying the kind of sensors used. The gateway will help to identify different devices and connect these devices and sensors to their destination devices or clouds. The gateway will provide some kind of standardized way of connecting between the devices and their preferred destination.

Over the course of last year there has been an increase in companies providing IoT related services which offers a standardized development platform for programmers. The three major cloud providers Amazon and IBM have come up with their own IoT platforms which enables users to connect their devices and sensors with the providers cloud

service and transfer telemetry data to the cloud and vice versa. Users can also make their own applications used to access the device and sensor data or control them using their mobiles or other such computing devices. These services can also learn from the physical devices and sensors and infuse intelligence into them.

Here the aim is to study major IoT cloud platforms- Amazon aws IoT, and IBM Watson IoT. The study will be done using a raspberry pi based application called weather station.
A microcontroller called Arduino will be used to consolidate data from different sensors and the output will be sent to the raspberry pi. This parts together forms the weather station. The telemetry data from the weather station will be transferred to the cloud using these services and apps will be developed for accessing their data and also controlling them. An example scenario the weather station will transfer data about temperature to the cloud services and an app will be made in the service that monitors this data and when the temperature causes 50 degree Celsius it will trigger an event of sending the data to users twitter account or an application developed by the user for their mobiles or tablets.

These services offer many kinds of different PaaS based IoT services like machine learning, neural network etc. which will not be covered here.

## Literature Survey:-

A smart home system in which the working of DHT11 sensor with raspberry pi was proposed in (D.P. Sharma, A.Baldeo, C. Philip, June 2015). (S. Nazeem Basha, Dr. S.A.K. Jilani, Mr.S. Arun, March 2016) proposed an intelligent door system using Raspberry pi and AWS IoT in which the connectivity to AWS IoT was clearly described using python language. The major IoT cloud platforms- Amazon aws IoT, and IBM Watson IoT – are taken and they have different approach/ concepts in connecting to IoT.

**AWS IoT:-**

AWS IoT is a platform that enables you to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline(Amazon Whitepaper, April 2016).

.

1) **AWS IoT Device SDK:-**
   AWS IoT provides an SDK to help you easily and quickly connect your hardware device or your mobile application. The AWS IoT Device SDK enables your devices to connect, authenticate, and exchange messages with AWS IoT using the MQTT, HTTP, or WebSockets protocols. The Device SDK supports C, JavaScript, and Arduino, and includes the client libraries, the developer guide, and the porting guide for manufacturers. You can also use an open source alternative or write your own SDK. Here we have gone for the eclipse paho mqtt SDK for communicating with the cloud. Eclipse paho-mqtt helps in writing programs in various languages like Java, Python etc. Here python was the main programming langiage used for collecting sensor data and communicating with AWS IoT service.

2) **Device Gateway:-**
   The AWS IoT Device Gateway enables devices to securely and efficiently communicate with AWS IoT. The Device Gateway can exchange messages using a publication/subscription model, which enables one-to-one and one-to-many communications. With this one-to-many communication pattern AWS IoT makes it possible for a connected device to broadcast data to multiple subscribers for a given topic. The Device Gateway supports MQTT, WebSockets, and HTTP 1.1 protocols and you can easily implement support for proprietary or legacy protocols. The Device Gateway scales automatically to support over a billion devices without provisioning infrastructure.

3) **Authentication and Authorization:-**
   AWS IoT provides mutual authentication and encryption at all points of connection, so that data is never exchanged between devices and AWS IoT without proven identity. AWS IoT supports the AWS method of authentication (called 'SigV4') as well as X.509 certificate based authentication. Connections using HTTP can use either of these methods, while connections using MQTT use certificate based authentication, and connections using WebSockets can use SigV4. With AWS IoT you can use AWS IoT generated certificates, as well as those signed by your preferred Certificate Authority (CA). You can map your choice of role and/or

policies to each certificate, so that you can authorize devices or applications to have access, or change your mind and revoke access altogether without ever touching the device.

**4) Registry:-**

The Registry establishes an identity for devices and tracks metadata such as the devices' attributes and capabilities. The Registry assigns a unique identity to each device that is consistently formatted regardless of the type of device or how it connects. It also supports metadata that describes the capabilities of a device, for example whether a sensor reports temperature, and if the data are Fahrenheit or Celsius.

**5) Device Shadows:-**

With AWS IoT you can create a persistent, virtual version, or "shadow," of each device that includes the device's latest state so that applications or other devices can read messages and interact with the device. The Device Shadows persist the last reported state and desired future state of each device even when the device is offline. You can retrieve the last reported state of a device or set a desired future state through the API or using the rules engine.

Device Shadows make it easier to build applications that interact with your devices by providing always available REST APIs. In addition, applications can set the desired future state of a device without accounting for the devices current state. AWS IoT will compare the difference between the desired and last reported state, and command the device to make up the difference.

**6) Rules Engine:-**

The Rules Engine makes it possible to build IoT applications that gather, process, analyze and act on data generated by connected devices at global scale without having to manage any infrastructure. The Rules Engine evaluates inbound messages published into AWS IoT and transforms and delivers them to another device or a cloud service, based on business rules you define. A rule can apply to data from one or many devices, and it can take one or many actions in parallel.

The Rules Engine can also route messages to AWS endpoints including AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, and Amazon Elasticsearch Service with built-in Kibana integration. External endpoints can be reached using AWS Lambda, Amazon Kinesis, and Amazon Simple Notification Service (SNS).

You can author rules within the management console or write rules using a SQL-like syntax. Rules can be authored to behave differently depending upon the content of the message. For example, if a temperature reading exceeds a certain threshold it could trigger a rule to transmit data to AWS Lambda. Rules can also be authored to take into account other data in the cloud, such as data from other devices. For example you could say take an action if this temperature is more than 15\% higher than the average of 5 other devices.

**IBM IoT:-**

IBM Internet of Things Foundation is a fully managed, cloud-hosted service that makes it simple to derive value from Internet of Things (IoT) devices(IBM IoT Documentation,2015). Using their recipes you can get your raspberry pi connected and start sending data securely up to the cloud using the open, lightweight MQTT messaging protocol.

From there, you can setup and manage the device using an online dashboard or using API's provided by them so that you can make apps that can access live and historical data pretty quickly.

Now we can take a look at IBM IoT Concepts:

**1) Organizations:-**

When you register with the Watson IoT platform you are given an organization ID, this is a unique 6 character identifier for your account. **Organizations** ensure your data is only accessible from your devices and

applications. Once registered, devices and API keys are bound to a single organization. When an application connects to the service using an API key it registers with the organization that "owns" the API key.

For security it is impossible for cross-organization communication within the Watson IoT platform eco-system, intentional or otherwise. The only way to transmit data between two organizations it to explicitly create two applications, one within each organization, that communicate with each other to act as a relay between the organizations.

**2) Devices:-**

1. A device can be anything that has a connection to the internet and has data it wants to get into the cloud.
2. A device is not able to directly interact with other devices.
3. Devices are able to accept commands from applications.
4. Devices uniquely identify themselves to the Watson IoT platform with an authentication token that will only be accepted for that device.
5. Devices must be registered before they can connect to the Watson IoT platform.

**Managed devices** are defined as devices which contain a management agent. A management agent is a set of logic which allows the device to interact with the Watson IoT platform Device Management service via the Device Management protocol. Managed devices can perform device management operations including location updates, firmware download and updates, and reboot and factory reset.

An **unmanaged device** is a device without a management agent. An unmanaged device can still connect to the Watson IoT platform and send and receive events and commands. However, it cannot send any device management requests, or perform any of the device management operations.

**3) Applications**

1) An application is anything that has a connection to the internet and wants to interact with data from devices and/or control the behaviour of those devices in some manner.
2) Applications identify themselves to the Watson IoT platform with an API key and a unique application ID.
3) Applications do not need to be registered before they can connect to the Watson IoT platform, however they must present a valid API key that has previously been registered.

**4) Gateway Devices**

Gateways are a specialized class of Device. They have the combined capabilities of an Application and a Device allowing them to serve as access points providing connectivity to the service for other devices without the ability to directly connect. Gateway devices can register new devices and can send and receive data on behalf of devices connected to them. Gateway Devices must be registered before they can connect to the service

**5) Events**

Events are the mechanism by which **devices** publish data to the Watson IoT platform. The device controls the content of the event and assigns a name for each event it sends.

When an event is received by the Watson IoT platform from a device the credentials of the connection on which the event was received are used to determine from which device the event was sent. With this architecture it is impossible for a device to impersonate another device.

- **Devices** are unable to receive events, regardless of whether they are its own events or those of another device.
- **Applications** are able to process events from devices in real time. When an application receives an event it has visibility of the source of the event and the data contained in that event. Applications can be configured to subscribe to all events from all devices, a subset of events, a subset of devices or a combination of these events.

**6) Commands:-**

**Commands** are the mechanism by which **applications** can communicate with **devices**. Only applications can send commands, which must be issued to specific devices.

The device must determine which action to take on receipt of any given command, and even controls which commands it will subscribe to in the first place. It is possible to design your device to listen for any command, or to simply subscribe to a set of specific commands.

The data in the clouds can be processed and send to various output locations. It can be send to applications like twitter and Gmail or can be send as an SMS to alert the user when the data crosses some checkpoints. The data can be send for data analysis to HDFS. The data can be plotted as a graph using different tools and can be used for data visualization using Microsoft Excel and other similar Softwares.

**Proposed System:-**
The figure below (FIG 1) is the abstract architecture of the weather station:

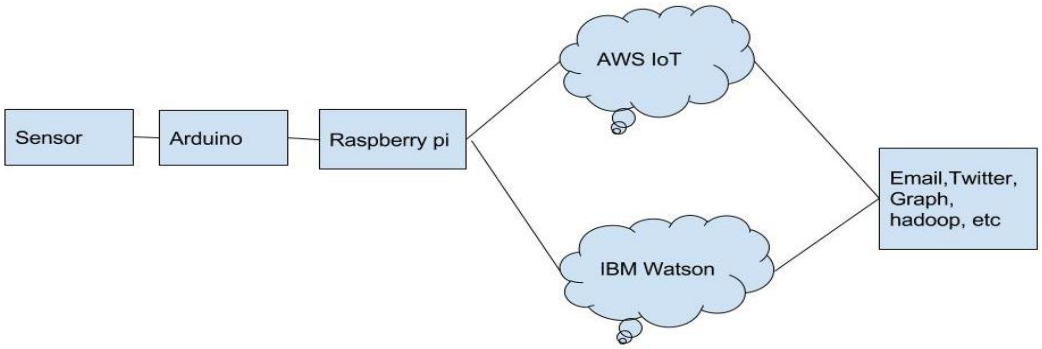


**FIG 1:** Architecture of the weather station application.

**Sensor**
The sensor used for sensing temperature and humidity is called Dht11 sensor. The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds (Dennis, A. K., 2013).

**Arduino**
Arduino Boards are microcontrollers used for connecting the sensors and taing their readings. There are many different types of Arduino boards available, here Arduino Uno board is used. The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller (Dennis, A. K., 2013).

**Raspberry Pi**
The Raspberry Pi is a series of credit card–sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation with the intent to promote the teaching of basic computer science in schools and developing countries. The original Raspberry Pi and Raspberry Pi 2 are manufactured in several board configurations through licensed manufacturing agreements with Newark element14 (Premier Farnell), RS Components and Egoman. The hardware is the same across all manufacturers. There are different generations and versions of raspberry pi, here Raspberry Pi Model 2B+ is used. It features a Broadcom system on a chip (SOC) which include an ARM compatible CPU and an on chip graphics processing unit GPU (a VideoCore IV). CPU speed 700 MHz and on board memory is 1 GB RAM. Secure Digital SD cards are used to store the operating system and program memory in either the SDHC or MicroSDHC sizes. It has four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support

common protocols like I2C. It has an RJ45 Ethernet port for connecting to internet (FOUNDATION, RASPBERRY PI. 2014), ( Upton, E. Halfacree, G. 2014).

Now we can take a look at the working of the system:

**1) Collecting Sensor Data:-**

The temperature and humidity sensor is a digital sensor. It can be connected to an Arduino board by connecting to a digital pin and connecting wires to Vcc and GND. The board can be then programmed using the Arduino IDE using C language. First we have to create a header file for the sensor called DHT.h and the output from the pins are coded in a C++ source file DHT.cpp. Then we will write the arduino code for the program to get our desired output we will save it in the .ino format. Then we will upload the code to Arduino board. Then we can check the output on serial monitor on Arduino IDE. The Arduino board is then connected via an USB to raspberry pi. The raspberry pi acts as a gateway between sensors and the cloud services. It is what is detected as an IoT device in the cloud. The raspberry pi can be coded using python (Dennis, A. K., 2013).

**2) IBM IoT implementation:-**

IBM Watson IoT offers 30 day free trial with all its features. Once signed in it takes you to a dashboard from which you select the topic IBM IoT foundation(FIG 2). Then you have to download the ibm-iot-sdk for raspberry pi from github. Then a code is run to get device id of the raspberry pi, which will give a unique device id. Then the device is registered by giving it a name and device type in the IBM IoTF dashboard(FIG 3). We will insert the device ID we got from raspberry pi.

The raspberry pi and remote application can be coded using different languages like nodejs, python etc. In this application python is used as the language of implementation. MQTT protocol requires an mqtt client, for python there is a python module called eclipse paho-mqtt which can be used for mqtt communication. For secure communication IBM provides authentication tokens which should be used for communicating with the cloud. The cloud acts as a broker for the mqtt protocol(A. Stanford-Clark and H. L. Truong,October,2007). The raspberry pi acts as a gateway device for connecting sensors and actuators.

The sensor data is collected from DHT11 sensor connected on a raspberry pi GPIO pin and sent to the broker by publishing to a topic “raspberry/temp” at the cloud. The data is sent as a message payload. The remote application requires the authentication tokens plus an authentication key to communicate with IBM cloud. This can be generated from the devices dashboard in cloud. The remote application then subscribes to a topic “raspberry/temp” at the cloud from where it receives the message payload. When it receives the data of the sensor it publishes a command to the topic “raspberry/led. Which is subscribed by the raspberry application from “raspberry/led”. The command alternates between “On” and “Off” which results in turning on and off of the LED.

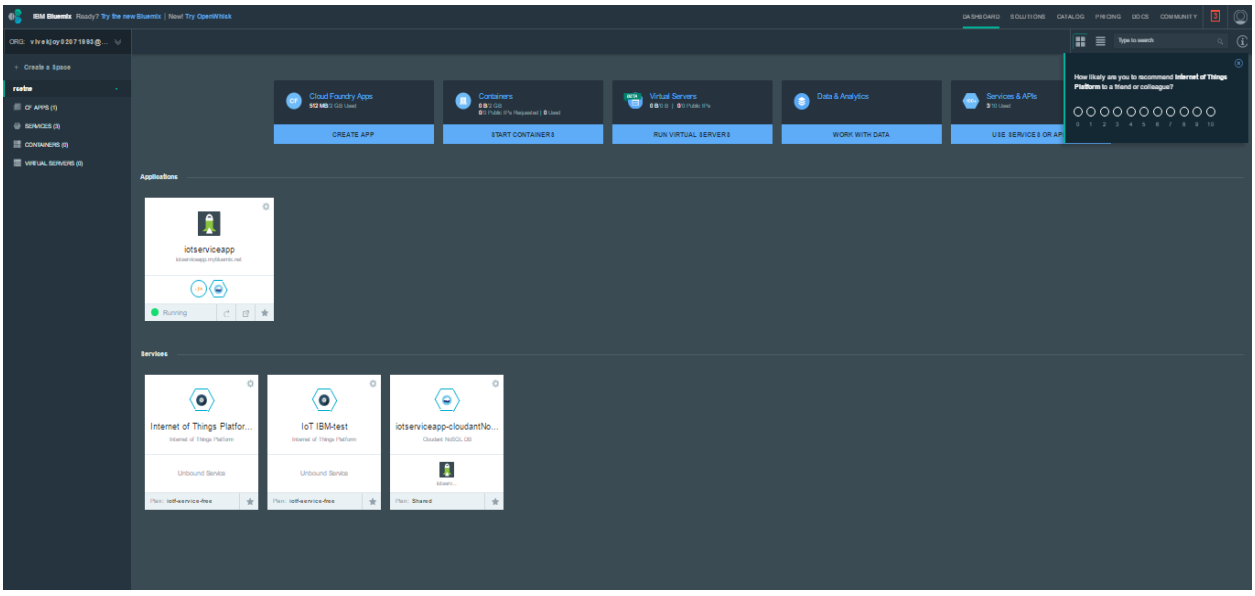

**FIG 2:** IBM IoT Web Interface

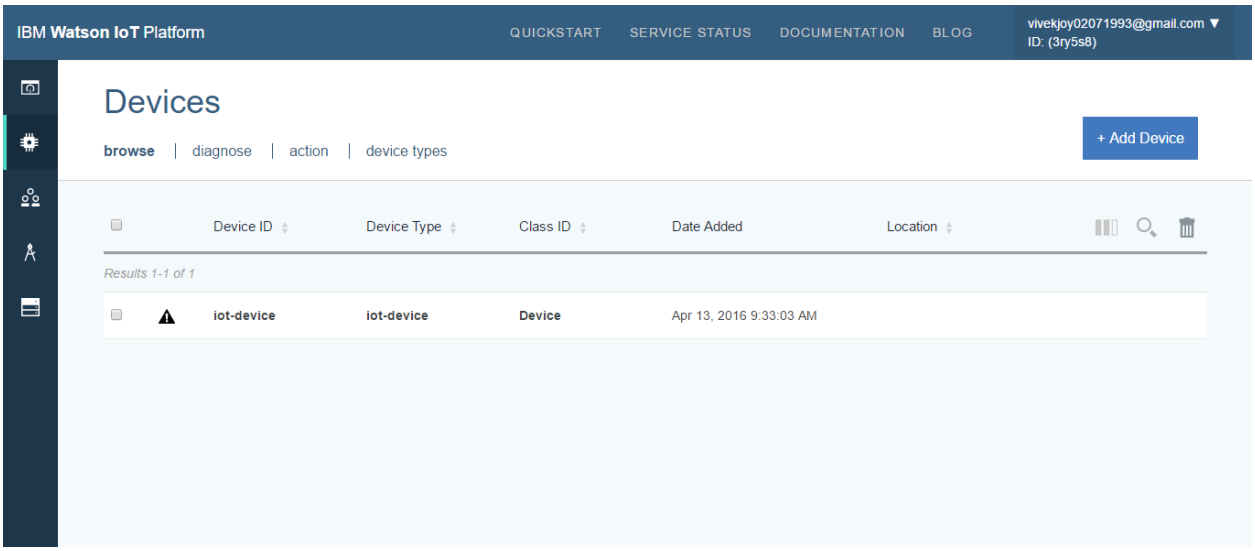


**FIG 3:** IBM IoT Devices Dashboard

### 3) AWS IoT Implementation:-

AWS IoT provides 12 months free tier services. After signing in it goes into a web interface(Fig 4). Devices are called things in AWS IoT and different services and actions are classified under resources in AWS IoT.
Resources has a button to create a resource. Resources consists of mainly five actions:

1. Create a thing - Here we create a thing named raspberry pi.
2. Create a certificate - generates digital certificates which is used for communication between the cloud and applications and devices.
3. Use my certificate and - AWS IoT gives a provision for the user to use their own certificates which they may have generated using their own algorithm.
4. Create a rule - A rule is used to connect to other services in the Amazon cloud. We can create a rule like if temperature is greater than a value then we can send a mail to an emailed using AWS SNS service or can store in their database called dynamoDB.
5. Create a policy - A policy determines how much access the thing has to the cloud.

We attach the policy and certificates and rules to the thing, which enables it to do the specified functions and certificates ensure authentication. Then we have to download AWS-iot-sdk from github. The coding of the applications can be done using nodejs, C or python. Here python is used is used. The digital certificates are downloaded and stored in a directory called certs. The certificates are a private key a public key and a root-CA certificate. Both the raspberry pi and remote application requires these certificates.
MQTT protocol requires an mqtt client, for python there is a library called eclipse paho mqtt which helps in mqtt communication(A. Stanford-Clark and H. L. Truong,October,2007). The certs folder directory should be provided in both the raspberry pi and remote applications. This helps in authentication. AWS IoT acts as a broker for mqtt pub/sub. There are two functions called **on_connect** and **on_message**. This functions are used to send and receive messages.

The publish function is written in **on_connect** function to send the sensor data to topic **"raspberry/temp"** The **on_message** function in the remote application reads the sensor data from **"raspberry/temp"** and it publishes an **"ON"** or **"OFF"** messages to the topic **"raspberry/led".** The raspberry pi application **on_connect** message receives the **"ON"** or **"OFF"** messages and turn on or off the LED respectively.
At the same time the published data from raspberry pi follows the rules and send messages to user email and also stores the data in dynamoDB. There is also a concept called thing shadows which stores the current state of the

thing, when the thing gets disconnected if an application subscribes to the thing it gets this current state from thing shadows. For using thing shadow the applications should publish to the topic AWS/thing/thingshadows/update/delta.

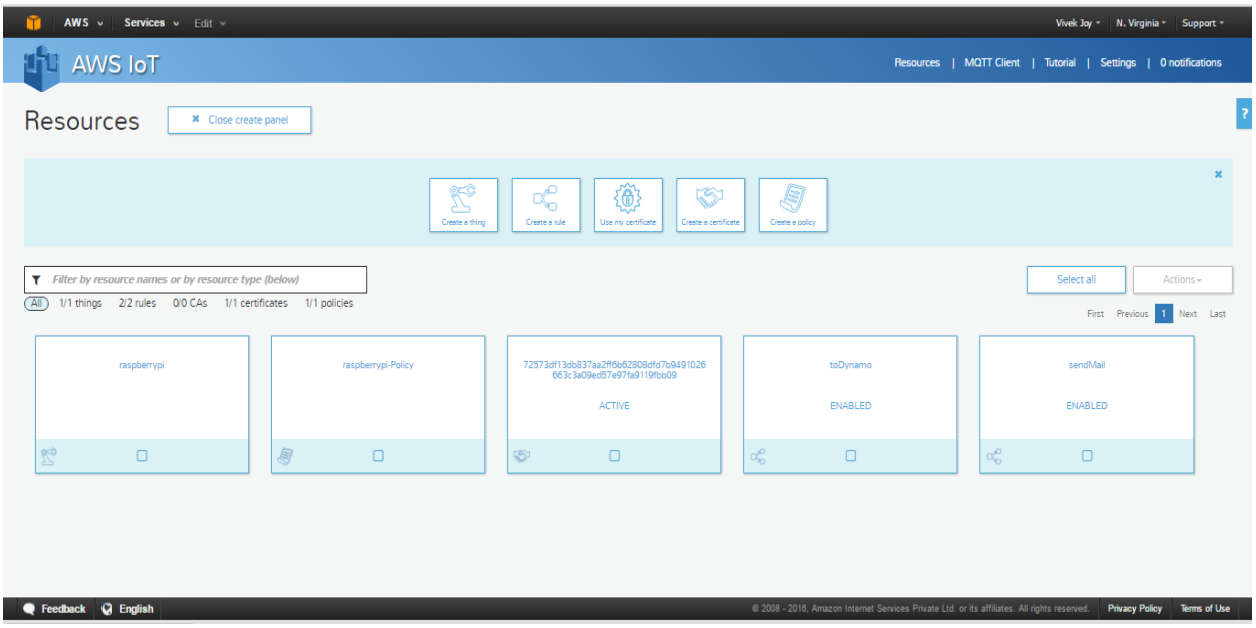


**FIG 4:** AWS IoT Web Interface

## Result and Discussion:-

The two cloud services are similar in their approaches. Both use same type of concepts like devices, gateways etc, The recommended communication protocol for both the services is MQTT. Both the services have native nodejs support for communicating with the cloud using their SDKs. Other programming languages are also supported using external libraries and SDKs. Here python was used with the support of eclipse paho mqtt library. Both the cloud services allow connecting to their different services like databases, Machine Learning services, MapReduce services and messaging services which can be used to send messages to applications like Email, Twitter etc.

Here in IBM IoT we have send the telemetry data received to twitter during a 5 minute interval. The data will be updated as a status in twitter(FIG 7). Also data can be send to gmail and other mails with IBM's native support for Node-Red which is a wiring tool for IoT devices like raspberry pi.

In AWS IoT The received data can be sent to various services like AWS Elastic MapReduce, Their databases like dynamoDB, we can send Email SMS etc or can even send to other topics for subscribing. Here we have stored the data in DynamoDB(FIG 5) using create a rule property of AWS IoT which has made storing the data to a database easier. Then we have also setup an AWS SNS service(FIG 6) for sending the data as an alert email to the user when the temperature croses a threshold.

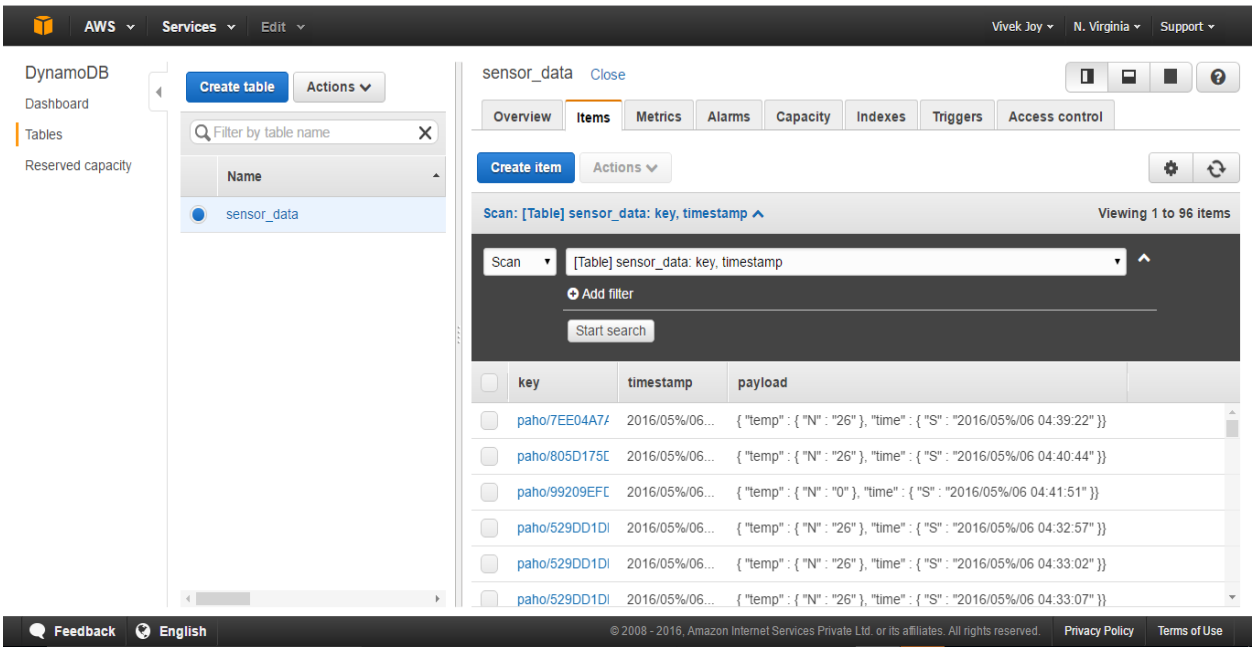


**FIG 5**: Storing data in DynamoDB using AWS IoT

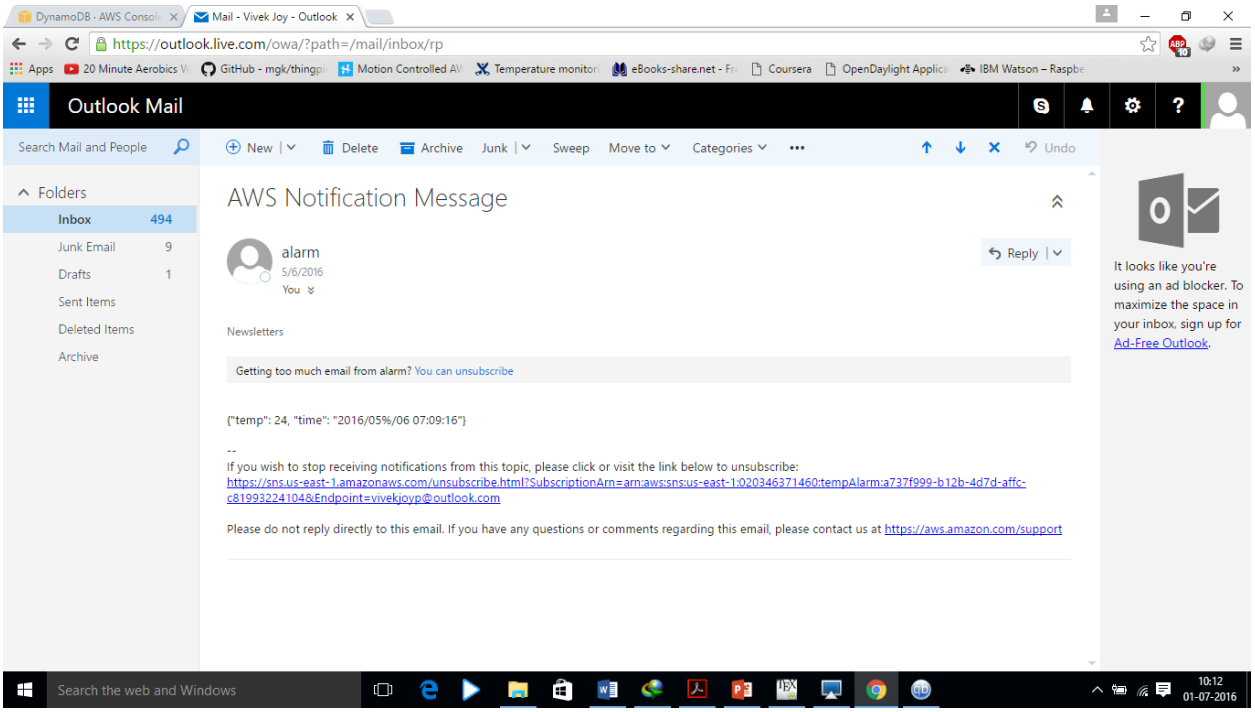


**FIG 6**: Sending Email alarm using AWS SNS service

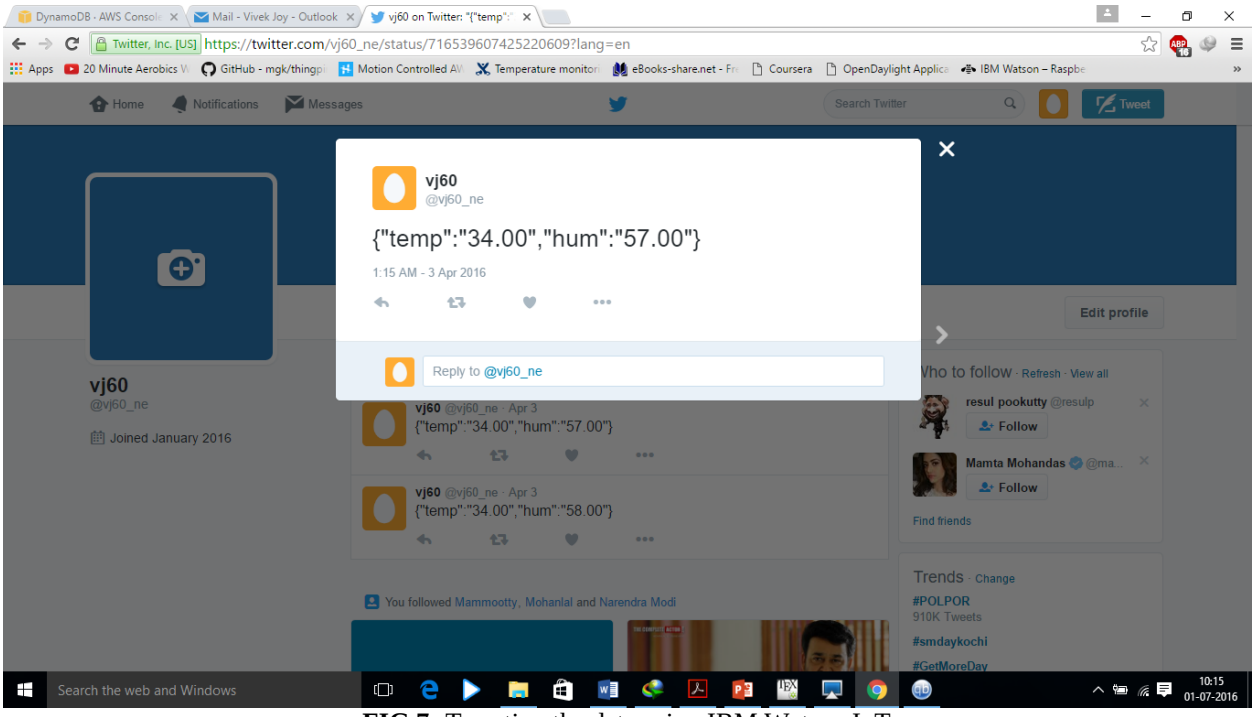


**FIG 7:** Tweeting the data using IBM Watson IoT

## References:-


1. **Amazon Whitepaper April 2016**, Core Tenets of IoT

2. **IBM IoT Documentation, (2015)** - https://iotf.readthedocs.io/en/latest/

3. **Dennis, A. K. (2013).** "Raspberry Pi Home Automation with Arduino". Packt Publishing.

4. **Upton, E. Halfacree, G. (2014** ),"Raspberry Pi User Guide", 3rd. Ed. Wiley.

5. **Bradbury, A. Everard, B. (2014)**, "Learning Python with Raspberry Pi". Wiley.

6. **FOUNDATION, RASPBERRY PI. (2014)**. NOOBS SETUP. Accessed 11, http://www.raspberrypi.org/help/noobs-setup/.

7. **X.Y. Chen & Z. G. Jin (2012)**, "Research on key technlogy and applications for Internet of Things," Physics Procedia, vol. 33,pp. 561-566.

8. **Stanford-Clark and H. L. Truong**, (**Oct. 2007** ),"MQTT for sensor networks (MQTTs)", http://www.mqtt.org/MQTTs_Specification_V1.0.pdf,.

9. **D.P. Sharma, A.Baldeo, C. Philip**, (**June 2015.)** "Raspberry Pi based Smart Home for Deployment in the Smart Grid", *International Journal of Computer Applications (0975 – 8887) Volume 119 – No.4.*

10. **. Nazeem Basha, Dr. S.A.K. Jilani, Mr.S. Arun,( March 2016)** An Intelligent Door System using Raspberry Pi and Amazon Web Services IoT, Vol.23 N0.2 IJETT.