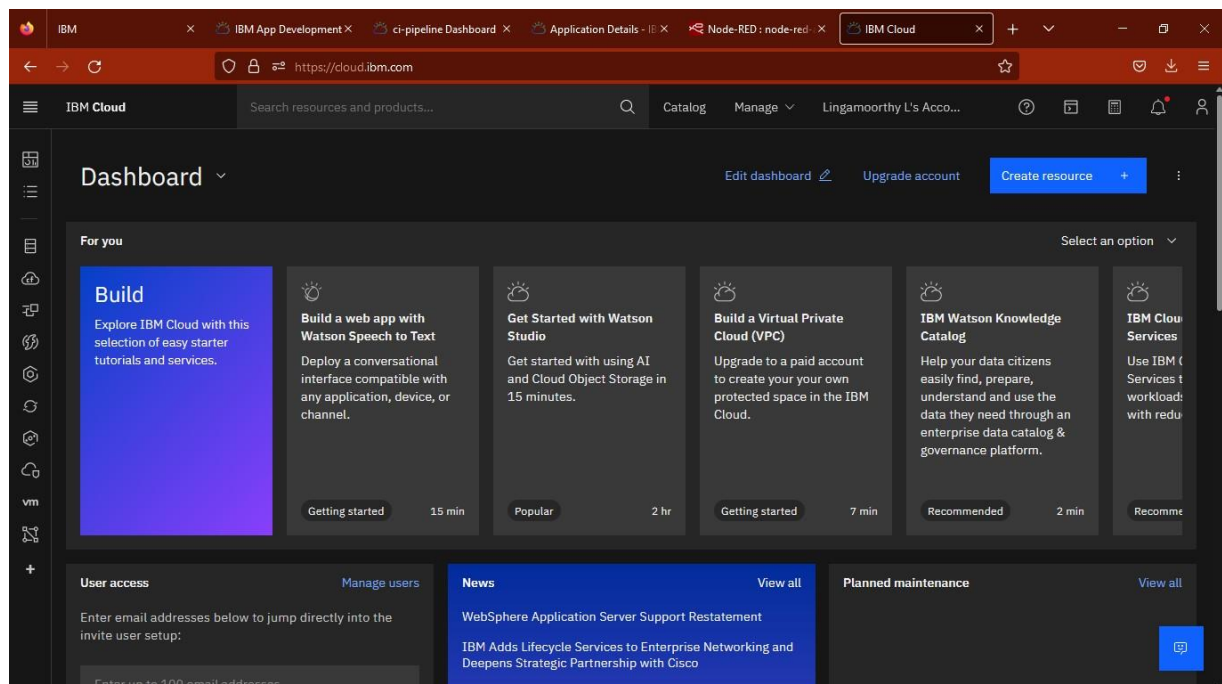


Project Development Phase Delivery of Sprint-1


Date	16 November 2022
Team ID	PNT22022TMID14391
Project Name	INDUSTRY - SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

BUILDING PHASE:

TO Create the Watson Assistant Service and IOT Platform:



Choosing the location:



Catalog /

Internet of Things Platform

This service is the hub of all things IBM IoT, it is where you can set up and manage your connected devices so that your apps can access their live and historical data.

Create About

Type
Service

Provider
IBM

Last updated
08/15/2022

Category
Internet of Things

Compliance
IAM-enabled

Location
Frankfurt
London
Dallas
Washington DC

Related links
Docs
Terms

Select a location

London (eu-gb) ▾

Select a pricing plan

Displayed prices do not include tax. Monthly prices shown are for country or location: [United States](#)

Plan	Features	Pricing
Lite	<p>Includes up to 500 registered devices, and a maximum of 200 MB of each data metric</p> <ul style="list-style-type: none">Maximum of 500 registered devicesMaximum of 500 application bindingsMaximum of 200 MB of each of data exchanged, data analyzed and edge data analyzed <p>The Lite service plan for Internet of Things Platform includes up to 500 registered devices, and a maximum of 200 MB each of data exchanged, data analyzed, and edge data analyzed per month.</p> <p>Lite plan services are deleted after 30 days of inactivity.</p>	Free

Summary

Internet of Things Platform **Free**

Location: London

Plan: Lite

Service name: Internet of Things Platform-mw

Resource group: Default

☒ I have read and agree to the following license agreements:
[Terms](#)

Create

Add to estimate

Resource list /

Internet of Things Platform-ip

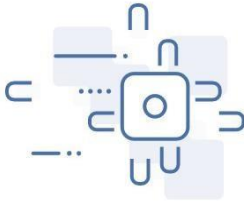
Active Add tags

Details Actions...

Manage

Plan

Connections



Let's get started with IBM Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

Launch Docs

Ready for the next level?

IBM Watson IoT Platform Journey

☒ **Lite**


The Lite service plan provides a lightweight

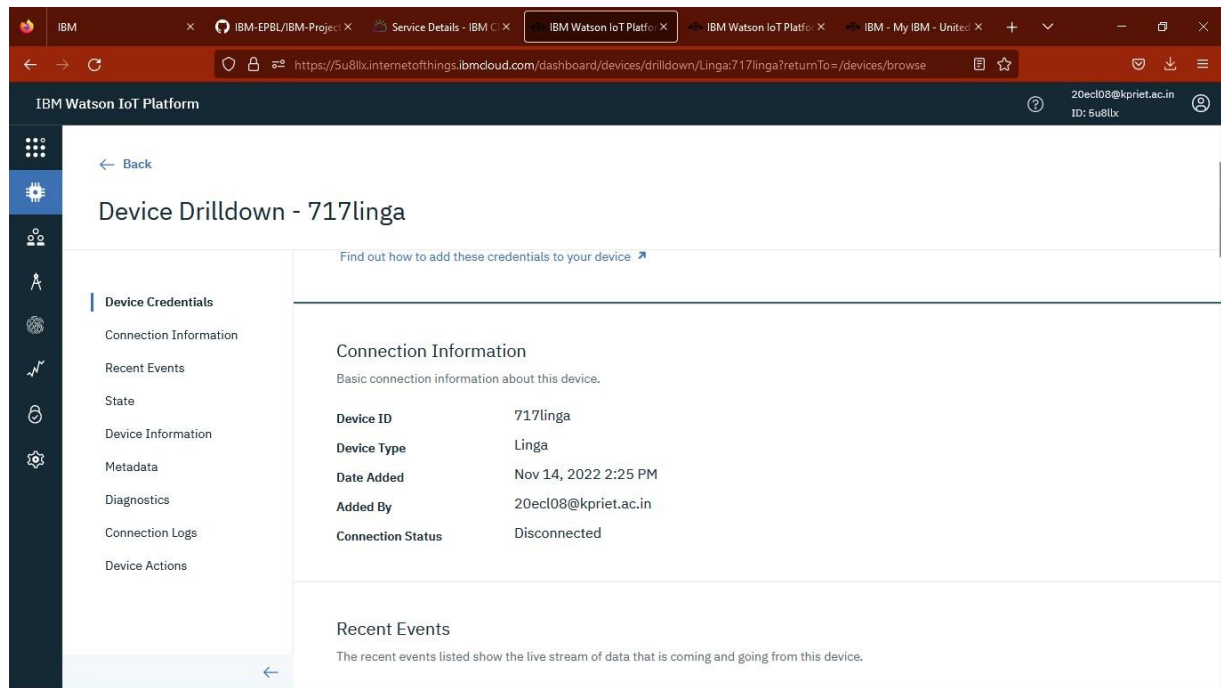
☐ **Non-Production**

The Non-Production service plan is a full-

☐ **Production**

The Production service is a fully managed SaaS





Program:

```
#include <WiFi.h>
#include <PubSubClient.h>

#include "DHT.h"

#define DHTPIN 15

#define DHTTYPE DHT22

#define LED 2

DHT dht (DHTPIN, DHTTYPE);

void callback(char* subscribetopic, byte* payload, unsigned int  payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "zbgr67"//IBM ORGANITION ID

#define DEVICE_TYPE "fershidevicetype"
```

```

#define DEVICE_ID "fershideviceid"           //Device ID mentioned in ibm watson IOT
Platform

#define TOKEN "fershiageona"   //Token   String

data3; float t;

char publishTopic[] = "iot-2/evt/Data/fmt/json";    // topic name and type of event perform and
                                                    format in which data to be send

char subscribetopic[] = "iot-2/cmd/command/fmt/String";    // cmd REPRESENT
command type AND COMMAND IS TEST OF FORMAT STRING char
authMethod[] = "usetoken-auth";    // authentication method char token[] = TOKEN; char
clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;    //client id


PubSubClient client(server, 1883, callback ,wifiClient);

void setup()// configureing the ESP32
{
    Serial.begin(115200); dht.begin(); pinMode(LED,OUTPUT); delay(10);
Serial.println(); wificonnect(); mqttconnect();

} void loop()// Recursive
Function    {
    t = dht.readTemperature();

    Serial.print("temperature:");

    Serial.println(t);

    PublishData(t); delay(1000); if
(!client.loop()) {    mqttconnect();
    }
}

```

```

/*.....retrieving to Cloud.....*/

void PublishData(float temp) { mqttconnect();//function call for connecting to ibm

/* creating the String in in form JSON to update the data to ibm cloud */

String payload = "{\"temperature\": "; payload
+= temp; payload += " }";

Serial.print("Sending payload: ");

Serial.println(payload); if
(client.publish(publishTopic, (char*) payload.c_str()))
{

Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or else
it will print publish failed

} else {

Serial.println("Publish failed");

}

} void mqttconnect() { if
(!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server); while
(!!client.connect(clientId, authMethod, token)) {

Serial.print("."); delay(500);

} initManagedDevice();

Serial.println();

```

```

} } void wificonnect() //function defination for wificonnect
{

    Serial.println();

    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection while
(WiFi.status() != WL_CONNECTED) {
delay(500);

    Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());
} void initManagedDevice() { if
(client.subscribe(subscribetopic)) {    Serial.println((subscribetopic));

    Serial.println("subscribe to cmd OK");

} else {

    Serial.println("subscribe to cmd FAILED");

}

}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

Serial.print("callback invoked for topic: ");

Serial.println(subscribetopic); for (int i = 0; i < payloadLength;

```

```
i++) {  
  
  //Serial.print((char)payload[i]);   data3 +=  
  
  (char)payload[i];  
  
  }  
  
  
  Serial.println("data: "+ data3);  if(data3=="lighton")  
  
  {  
  Serial.println(data3); digitalWrite(LED,HIGH);  
  
  
  }  else  
  
  {  
  Serial.println(data3); digitalWrite(LED,LOW);  
  
  
  } data3="";  
  
  }
```