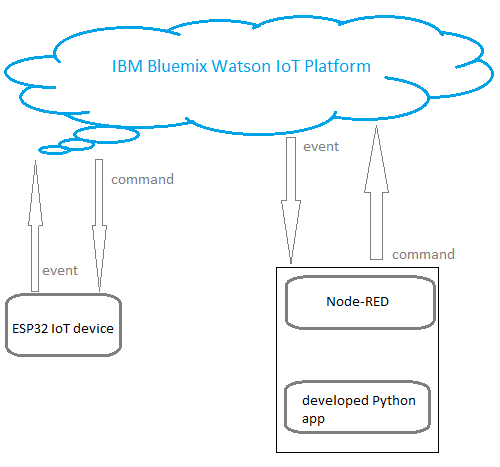
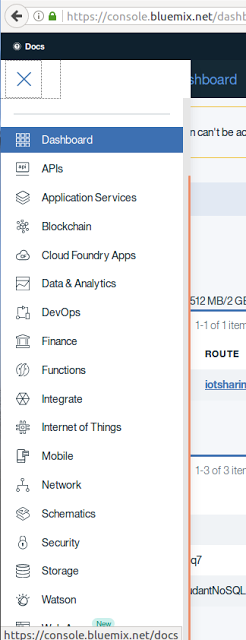
# **[Demo 33: Monitor and control ESP32 via IBM Bluemix Watson IoT Platform](http://www.iotsharing.com/2017/10/monitor-and-control-esp32-via-ibm-bluemix-watson-iot-platform.html)**

**1. Introduction**  
**2. Hardware**  
**3. Bluemix setup steps and ESP32 software**  
**4. How to monitor and control ESP32**  
**4.1 Node-Red**  
**4.2 Using own developed Python application**  
**5. Using graph to display data**  
**6. Using Alert Notification**  
**7. Message Hub**  
 **1. Introduction**  
In this demo, I will show you how to monitor and control ESP32 via IBM Bluemix Watson IoT Platform.  
IBM Bluemix is a cloud platform that supports many products and services such as: Compute Infrastructure, Compute Services, Storage, Mobile, Wason, Application services, Data and analytics, Internet of Things, ...  
In this demo, we just focus on Internet of Things service of Bluemix.  
In order to use Bluemix, you need to register an account. I registered a free account and it is available in 30 trial days.  
You can sign up [here](https://console.ng.bluemix.net/registration/). It is easy just follow the steps in registration form.  
Here is the model of our demo: ESP32 communicate with Node-RED and Python app via IBM Bluemix Watson IoT Platform.

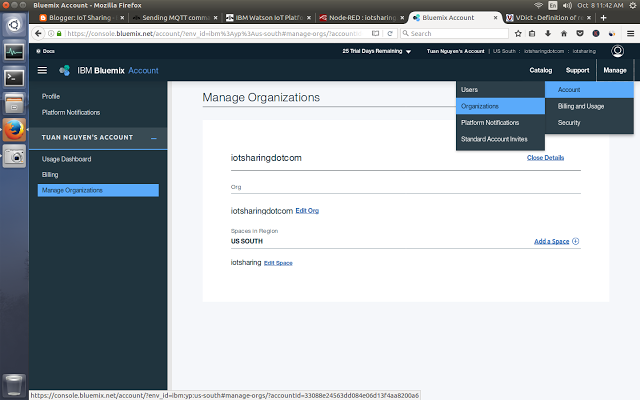
[](https://2.bp.blogspot.com/-XpNwIkSPNFQ/WdxCGJYhB6I/AAAAAAAAEZk/25IaPMupSd0qNyg1agmqAIqGXxXHjOdZgCLcBGAs/s1600/bluemix11.png)

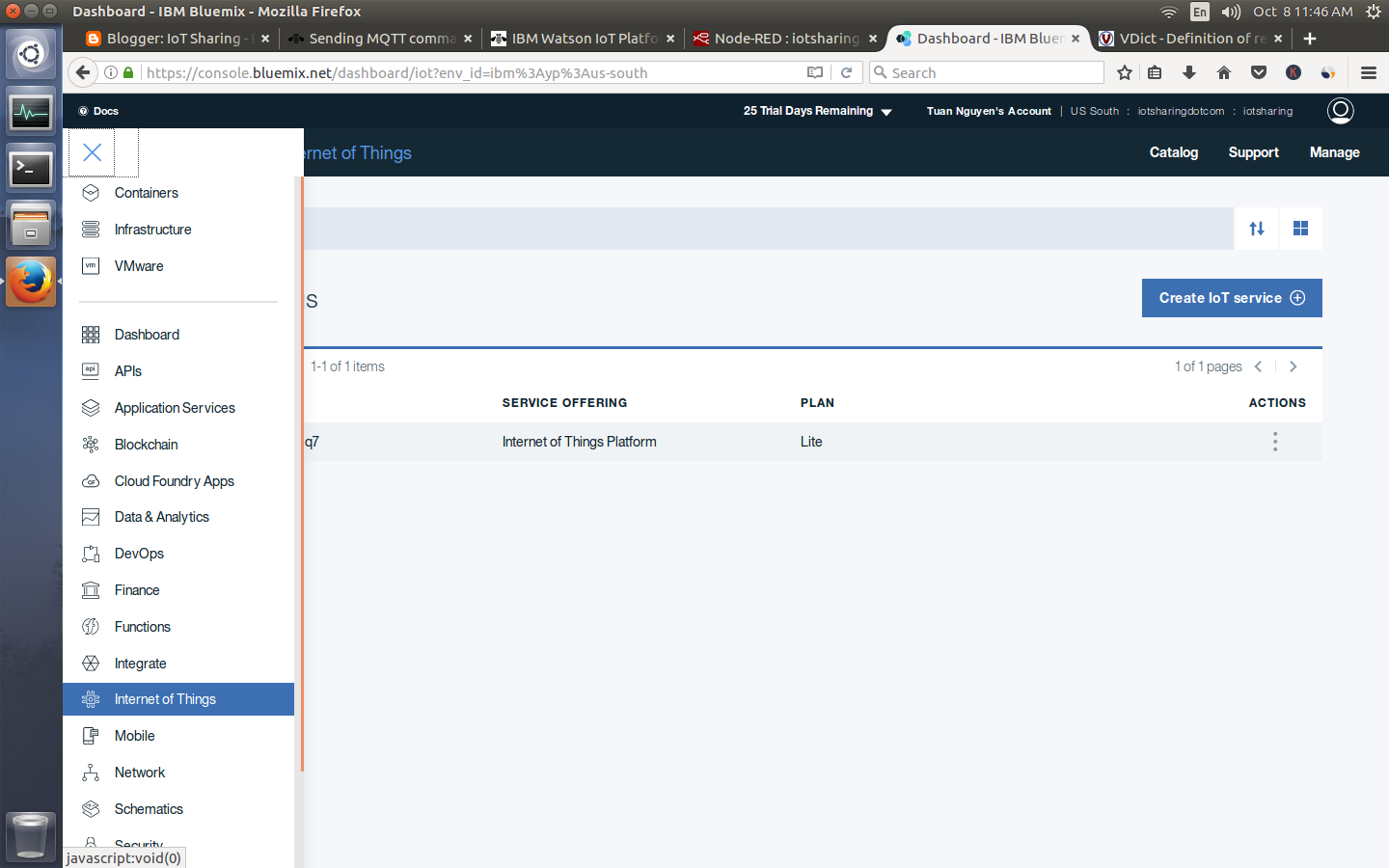
**Figure: ESP32 communicate with Node-RED, Python app via Bluemix**

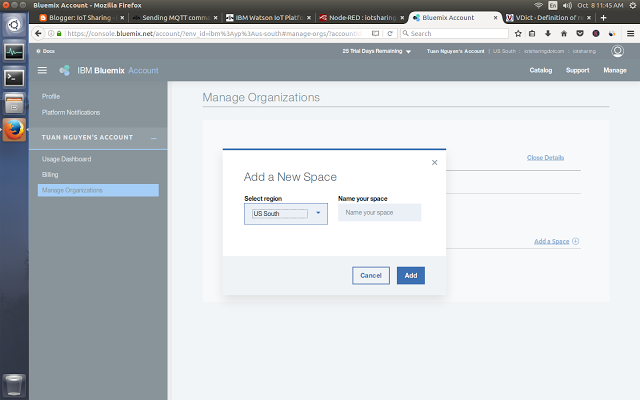
**2. Hardware**  
You need a LED or relay as in [Demo 1: Blinky](http://www.iotsharing.com/2017/05/blinky-hello-world-on-arduino-esp32.html)  
**3. Bluemix setup steps and ESP32 software**  
Here is the Menu of Bluemix services. It is in top-left corner.

[](https://2.bp.blogspot.com/-oEBvAZ1w9iE/WdmmsHo3hYI/AAAAAAAAEUI/qlKlngO6TCIrpXbBSw5cIkL7SDt6TIRUwCLcBGAs/s1600/bluemix8.png)

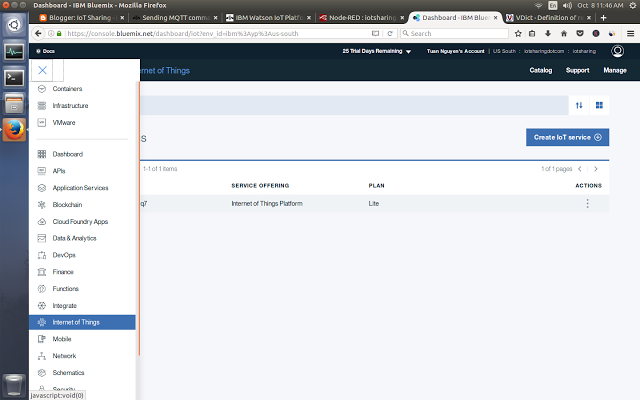
But before using IoT service we need to create a **Space** (where you deploy your IoT application) for it. It is under **Manage - Account - Organizations** tabs.

[](https://1.bp.blogspot.com/-beGfqWD1GEw/Wdms54YPWsI/AAAAAAAAEUY/2eVXjlYtkjAZyfoDE3MaJrWTRyKZ0BZewCLcBGAs/s1600/bluemix9.png)

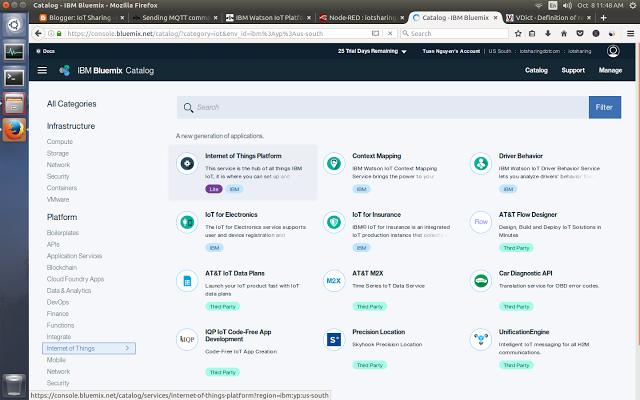
[](https://2.bp.blogspot.com/-KaSiBAUUh1w/Wdmt065J4YI/AAAAAAAAEUk/gQdgfTWb1Lk3ajOgIdQg8UO10KPUpD6PQCLcBGAs/s1600/bluemix11.png)

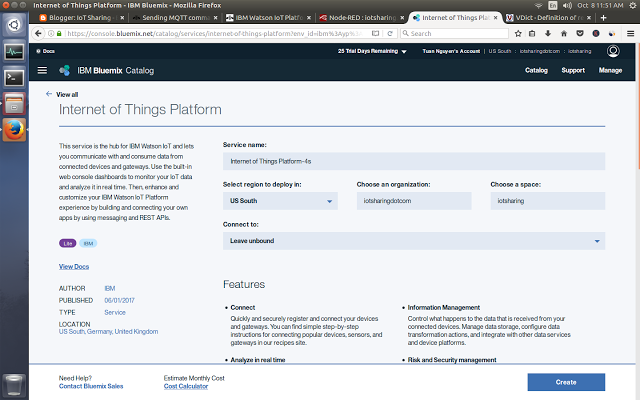
[](https://2.bp.blogspot.com/-tQ_qUSTfqEI/WdmtdIc7iTI/AAAAAAAAEUg/cdUaauiADrg_irxEvoNfa_0ZMOT-GQDcwCLcBGAs/s1600/bluemix10.png)

From Menu choose IoT service

[](https://2.bp.blogspot.com/-KaSiBAUUh1w/Wdmt065J4YI/AAAAAAAAEUk/Ruwm7YIw_yoO-qXpEzAW85EKka08J39CwCEwYBhgL/s1600/bluemix11.png)

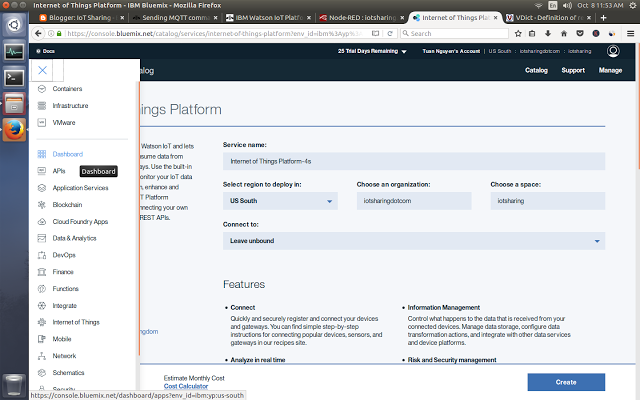
Choose **Create IoT Service**and**Internet of Thins Platform**

[](https://1.bp.blogspot.com/-lgLPlfyvZ34/WdmunBXasAI/AAAAAAAAEUs/eqgSLOCWIJ0lU3NGKKrZpva2JDkEZsmvQCLcBGAs/s1600/bluemix12.png)

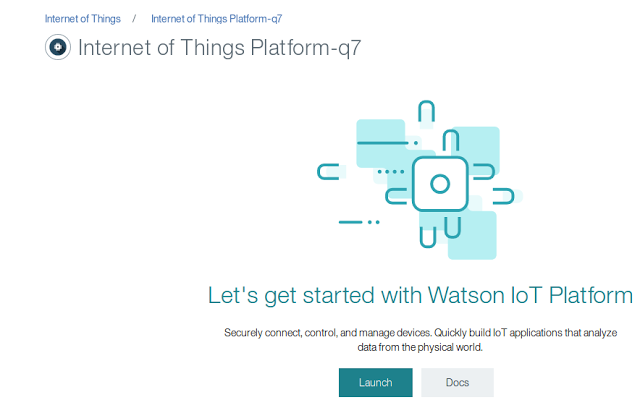
[](https://1.bp.blogspot.com/-afkeDbZGhQ4/Wdmu7WPvEXI/AAAAAAAAEUw/XcW4BQixChQtKCAS76FNABUvVRNrBjbhwCLcBGAs/s1600/bluemix13.png)

Edit fields as your expectation and choose **Create** ... Done

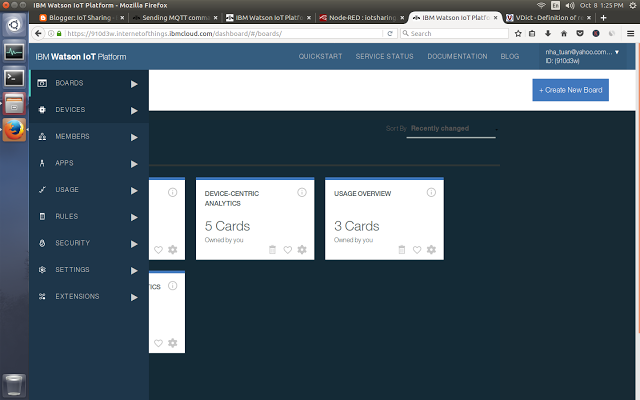
From Menu choose **Dashboard**

[](https://2.bp.blogspot.com/-NYsgU6dM0W0/WdmvWbSQxdI/AAAAAAAAEU0/yl25gxIxq1M1aGQrFv3tuRoYlIKnAgtUQCLcBGAs/s1600/bluemix14.png)

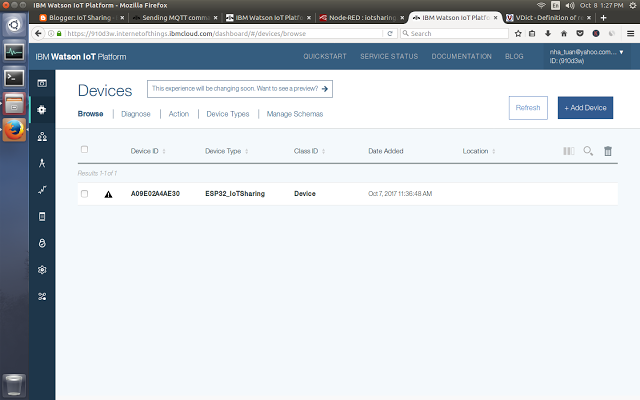
ChooseIoT service which you created and choose **Launch**

**[](https://4.bp.blogspot.com/-WxwJmBI2OoQ/WdmwfqAhGJI/AAAAAAAAEVA/5OWQ162ovfAikwKtRQuhvnmvtpLi8Jd4wCLcBGAs/s1600/bluemix15.png)**

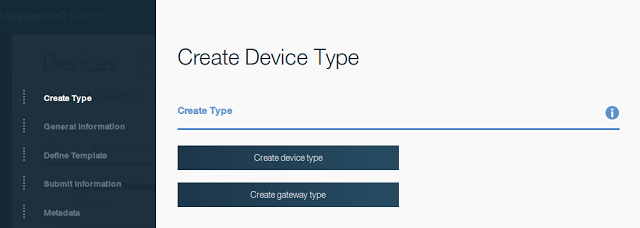
Now you see IBM Watson IoT Platform dashboard with Menu

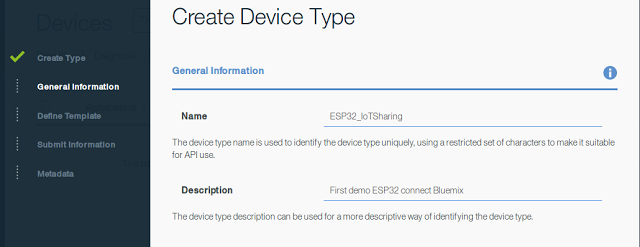
[](https://4.bp.blogspot.com/-e1d1oy-Z2gY/WdnFFqmnXEI/AAAAAAAAEVQ/3XhVMpzn4hAmCUSmY7hdFDFlSAbcBqKxwCLcBGAs/s1600/bluemix16.png)

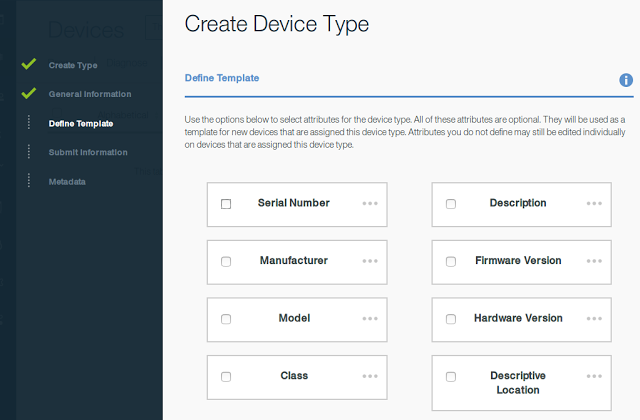
Choose **Devices**and**Add Device**

[](https://3.bp.blogspot.com/-5ZXBFW3xuPk/WdnFeFPaofI/AAAAAAAAEVU/qqsavP4rnakvmp0fm2jDHnZYvMC6cq6mwCLcBGAs/s1600/bluemix17.png)

Before adding new device, It will recommend you to add Device Type and related device information first.

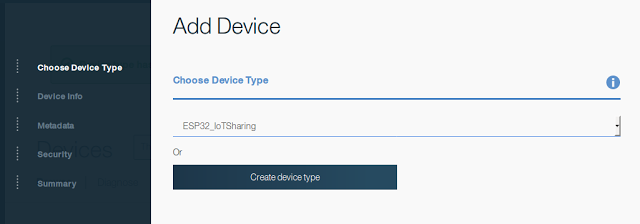
[](https://1.bp.blogspot.com/-l-wi0fSVANU/WdnGTBkw0eI/AAAAAAAAEVc/2UV_J6L3Wl8CEbG3CJWUF3PdVdLejbDhwCLcBGAs/s1600/bluemix3.png)

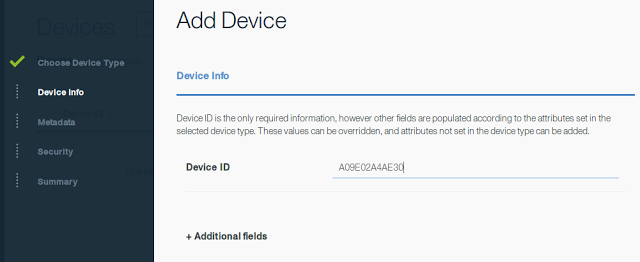
[](https://3.bp.blogspot.com/-u-ydVDMKd34/WdnGeErDnCI/AAAAAAAAEVk/HP8MTRaB6G04kHuu2V37THQfViel26dtgCLcBGAs/s1600/bluemex3.png)

[](https://1.bp.blogspot.com/-2S34TXbLh8w/WdnGYmZ-fgI/AAAAAAAAEVg/8ufgGruHmWQBEwXlSCZEtLy6dh1egKsfACEwYBhgL/s1600/bluemix4.png)

Here you can fill it with expected information or leave them empty (modify them later).

After creating Device Type, you can add new device.

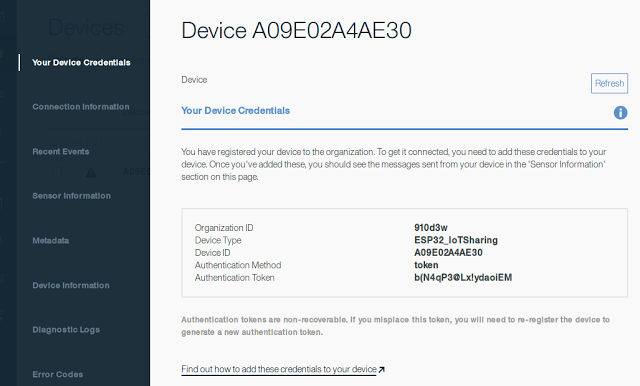
[](https://4.bp.blogspot.com/-xqblLdeNxN0/WdnJlVY3zdI/AAAAAAAAEVw/YBxHrfCqr0k9pex9RdBymovvtf-IO_2rQCLcBGAs/s1600/bluemix5.png)

[](https://1.bp.blogspot.com/-qheEg30Vww4/WdnJzP1RSVI/AAAAAAAAEV4/8K6Jbc-A2EsXWW3xNU8N1wxtKUeX7cYxQCLcBGAs/s1600/bluemix6.png)

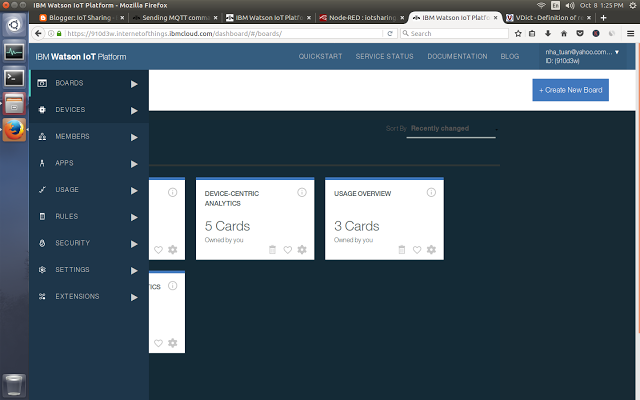
Here, Device ID is MAC address of ESP32. You can use the code below to get MAC address:

|  |
| --- |
| uint64\_t chipid;  void setup() {  Serial.begin(115200);  }  void loop() {  chipid=ESP.getEfuseMac();//The chip ID is essentially its MAC address(length: 6 bytes).  Serial.printf("ESP32 Chip ID = %04X",(uint16\_t)(chipid>>32));//print High 2 bytes  Serial.printf("%08X\n",(uint32\_t)chipid);//print Low 4bytes.  delay(3000);  } |

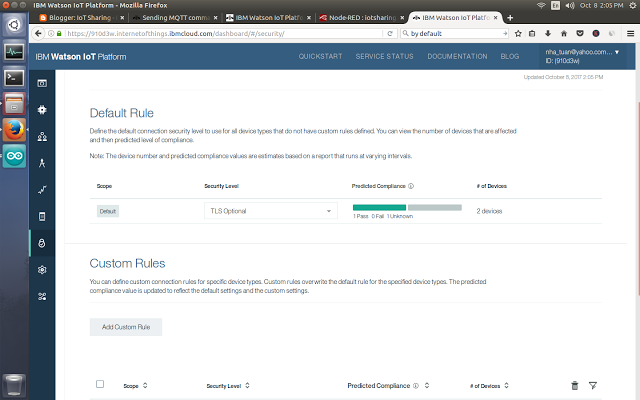
And let authentication token is Auto-generated. Finally, you have:

[](https://1.bp.blogspot.com/-Sqnf6ZdUSKQ/WdnJta6L1bI/AAAAAAAAEV8/0IBuYRPOQ1oI-c2Ry44uXrvJdb-JGDD2wCEwYBhgL/s1600/bluemix7.png)

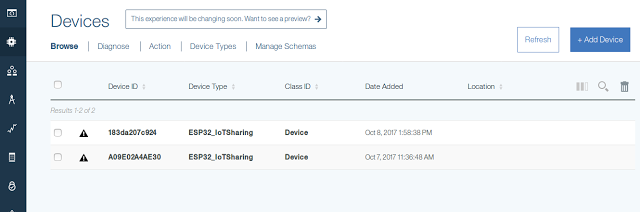
Remember to save this information for future use.  
**Note:** **TLS with Token Authentication** is enabled by default. So if your device use TLS you can ignore it. In my demo using ESP32, I do not use TLS so I have to disable it to avoid Authentication error. In order to disable it just chose **Security** from Menu.

[](https://4.bp.blogspot.com/-e1d1oy-Z2gY/WdnFFqmnXEI/AAAAAAAAEVY/5wtTMAr_cYwSsgeBLjUbaPYefCbPEkpbACEwYBhgL/s1600/bluemix16.png)

Choose **Connection Security**and then**TLS Optional**.

[](https://2.bp.blogspot.com/-e8yXyMMu5vE/WdnOd8ULXVI/AAAAAAAAEWE/sVcTA9HmAVMe3S4qzA4htde8SUBaxWAqACLcBGAs/s1600/bluemix18.png)

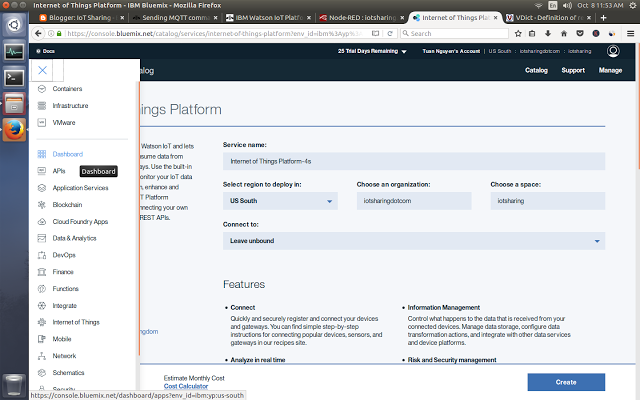
Now choose Devices from Menu, you will see a list of devices with Disconnected sign. It is time to connect our devices first time.

[](https://2.bp.blogspot.com/-LUQvvcadxXY/WdnPFJul0BI/AAAAAAAAEWM/dfWUo7bLQ_o_rsWGhGP1uWsUeZd2sKowACLcBGAs/s1600/bluemix19.png)

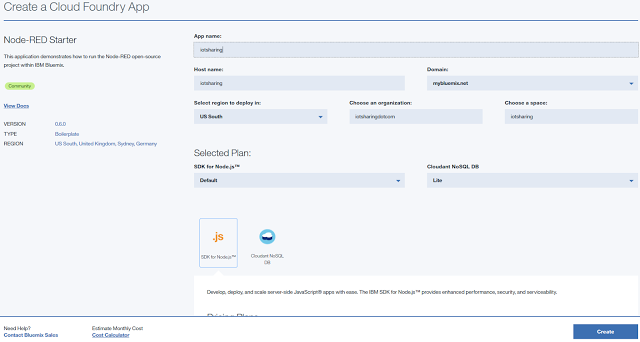
In order to connect to IBM Watson™ IoT Platform, we will use MQTT protocol ([Demo 14](http://www.iotsharing.com/2017/05/how-to-use-mqtt-to-build-smart-home-arduino-esp32.html)). You can refer [document](https://console.bluemix.net/docs/services/IoT/iotplatform_task.html" \l "iotplatform_task) to understand the procedure to connect to it. The Arduino ESP32 code will do the jobs:  
- Connect to cloud.  
- Publish a temperature message (generated by using random function). If you have DHT11/DHT22 you can use [Demo 3](http://www.iotsharing.com/2017/05/how-to-arduino-esp32-dht11-dht22-temperature-humidity-sensor.html).  
- Subscribe the ON/OFF message to control LED.

|  |
| --- |
| #include <WiFi.h>  #include <WiFiClient.h>  #include <PubSubClient.h>  //-------- your wifi -----------  const char\* ssid = "dd-wrt";  const char\* password = "0000000000";  #define ORG "910d3w"  #define DEVICE\_TYPE "ESP32\_IoTSharing"  #define DEVICE\_ID "A09E02A4AE30"  #define TOKEN "b(N4qP3@Lx!ydaoiEM"  //-------- Bluemix information to build up MQTT message -------  char server[] = ORG ".messaging.internetofthings.ibmcloud.com";  char pubTopic[] = "iot-2/evt/status/fmt/json";  char subTopic[] = "iot-2/cmd/test/fmt/String";  char authMethod[] = "use-token-auth";  char token[] = TOKEN;  char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;  WiFiClient wifiClient;  PubSubClient client(server, 1883, NULL, wifiClient);  const char led = 4;  void receivedCallback(char\* pubTopic, byte\* payload, unsigned int length) {  Serial.print("Message received: ");  Serial.println(pubTopic);  Serial.print("payload: ");  for (int i = 0; i < length; i++) {  Serial.print((char)payload[i]);  }  Serial.println();  /\* we got '1' -> on \*/  if ((char)payload[0] == '1') {  digitalWrite(led, HIGH);  } else {  /\* we got '0' -> on \*/  digitalWrite(led, LOW);  }  }  void setup() {  Serial.begin(115200);  Serial.println();  pinMode(led, OUTPUT);  Serial.print("Connecting to ");  Serial.print(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");    Serial.print("WiFi connected, IP address: ");  Serial.println(WiFi.localIP());  if (!client.connected()) {  Serial.print("Reconnecting client to ");  Serial.println(server);  while (!client.connect(clientId, authMethod, token)) {  Serial.print(".");  delay(500);  }  client.setCallback(receivedCallback);  if (client.subscribe(subTopic)) {  Serial.println("subscribe to cmd OK");  } else {  Serial.println("subscribe to cmd FAILED");  }  Serial.println("Bluemix connected");  }  }  long lastMsg = 0;  long temperature = 0;  void loop() {  client.loop();  long now = millis();  if (now - lastMsg > 3000) {  lastMsg = now;  temperature = random(0, 40);  String payload = "{\"d\":{\"Name\":\"" DEVICE\_ID "\"";  payload += ",\"temperature\":";  payload += temperature;  payload += "}}";  Serial.print("Sending payload: ");  Serial.println(payload);  if (client.publish(pubTopic, (char\*) payload.c\_str())) {  Serial.println("Publish ok");  } else {  Serial.println("Publish failed");  }  }  } |

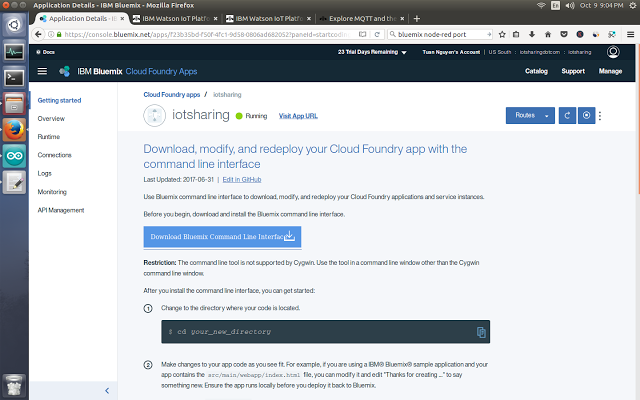
**4. How to monitor and control ESP32**  
We went through steps to setup Bluemix IoT Platform and ESP32. Now we will learn how to develop or setup applications to monitor and control ESP32 Iot device. There are some ways to do that. In this demo, I will use 2 ways: Node-Red (you can refer [Demo 8](http://www.iotsharing.com/2017/05/tcp-udp-ip-with-esp32.html) to know more about Node-RED) and own developed Python application.  
**4.1 Node-Red**  
We will use Bluemix Node-RED Starter.  
From Menu choose **Clound Foundry Apps**and then**Create Clound Foundry apps**. From **Filter** box search**"Node-RED Starter"**.Choose **"Node-RED Starter"** and fill expactation information.

**[](https://1.bp.blogspot.com/-NYsgU6dM0W0/WdmvWbSQxdI/AAAAAAAAEU4/d3LLbKNsE7gY1siSwic3apm_LX7x1ilSwCPcBGAYYCw/s1600/bluemix14.png)**

Click **Create**

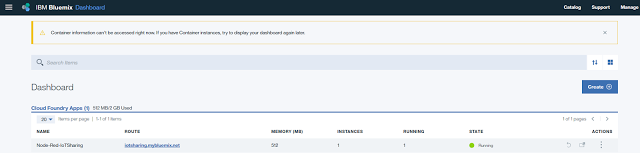
[](https://4.bp.blogspot.com/-kA6QHRV-jL4/WdrttZ3JpHI/AAAAAAAAEWg/F5vSGfHiS7osWFlouSeSnyaM22f9OLGOQCEwYBhgL/s1600/bluemix.png)

Waiting until service was deployed ...

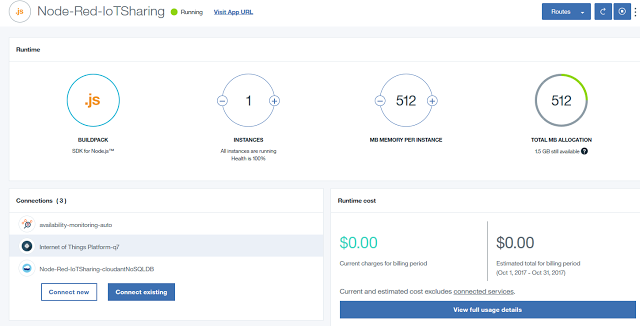
[](https://2.bp.blogspot.com/-loDRDHOntSw/WduDccXXU3I/AAAAAAAAEYA/_iO1NHMKqW4fsz6kILB62_I2VYlUtS2_wCLcBGAs/s1600/bluemix20.png)

**Figure: Node-RED service was deployed successfully**

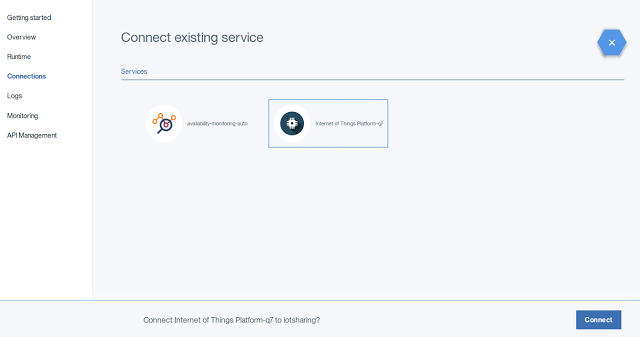
Next, from Menu choose **Dashboard**and you can see "Node-RED" is running. And you can access Node-RED flow editor online by going to the link in **Route**column (it is "[iotsharing.mybluemix.net](https://iotsharing.mybluemix.net/)")

[](https://4.bp.blogspot.com/-5fLFYisQ-D4/WdrvTP2eprI/AAAAAAAAEWo/xQ11acBqzckntSsWpJC7S8xP_nNerBjtACLcBGAs/s1600/bluemix1.png)

Now, we need to bind our Node-RED starter with our created Bluemix IoT Platform service. From dashboard and Cloud Foundry Apps table, click the Node-RED row (picture above).

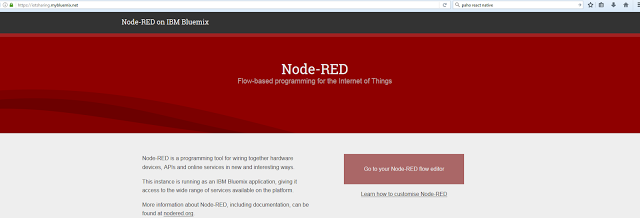
[](https://2.bp.blogspot.com/-suZui3ZKb38/Wdrx1-JpJJI/AAAAAAAAEW4/6cBmfp7WBuAuLs6OSszgN1Fw1IIWWYkuQCLcBGAs/s1600/bluemix2.png)

Choose **Connect existing**and choosethe Bluemix IoT Platform service which we created in **Section 3 Bluemix setup steps and ESP32 software**(the result is similar to picture above).

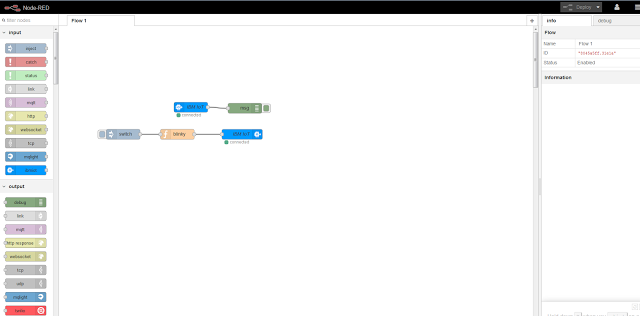
[](https://3.bp.blogspot.com/-TmpcLDWaEe0/WduDIxQG2WI/AAAAAAAAEX8/W4dBHhja0Twk_JwopL--ng2G7VRV0iocwCLcBGAs/s1600/bluemix21.png)

**Figure: Bind Node-RED with Bluemix IoT Platform**

**4.1.1 Let 's play with Node-RED**  
We go to the link in **Route**column ("[iotsharing.mybluemix.net](https://iotsharing.mybluemix.net/)").

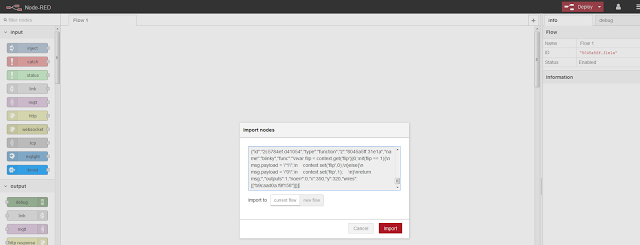
[](https://3.bp.blogspot.com/-p30BvHOfxQE/WdrzsisQhYI/AAAAAAAAEXE/BNyrgQOi3OoVLtwNV3oBHOGWHb3ZXgKYgCLcBGAs/s1600/bluemix3.png)

And choose **Go to your Node-RED flow editor**(you may need to wait a little time to initialize Node-RED flow editor if you see 404 error).  
Finally, it is done.

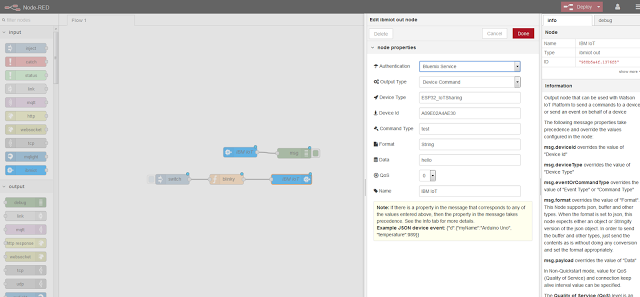
[](https://3.bp.blogspot.com/-zOMKmV-bfek/Wdr0Ukq4EQI/AAAAAAAAEXM/XSldkINbp-s0mg2n30vd2ZpLSixjUp85gCLcBGAs/s1600/bluemix4.png)

Node-RED Starter supports 2 nodes named IBM IoT in/out to connect to Bluemix IoT Platform easily. I created the flow as picture above and Export that flow to JSON format. So you just Import the JSON content into your Flow and change Device Information accordingly to the device which you created in **Section 3 Bluemix setup steps and ESP32 software**. In my Flow there are 2 blocks: 1 subscribe block (upper block; using IBM IoT in) to subscribe the event from ESP32 IoT device and bring to Debug block through Debug tab (Top-Right corner); 1 publish block (lower block; using IBM IoT out) to publish command ON/OFF LED to ESP32 IoT device.

[](https://1.bp.blogspot.com/-VqZf9pOsm0U/Wdr5qFEFaPI/AAAAAAAAEXc/1Cftxo-6Jj0Mby40QlK-bUiYyVx2szMhgCLcBGAs/s1600/bluemix5.png)

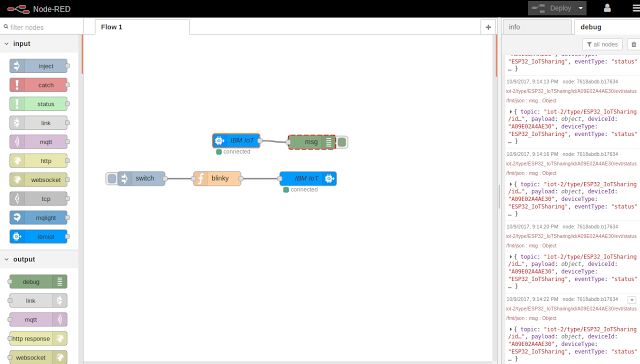
[](https://1.bp.blogspot.com/-aGsXTcbc67k/Wdr54Hq32RI/AAAAAAAAEXg/napSzmMmbAwyRfbhKmti6Rvrl63PmA0BgCLcBGAs/s1600/bluemix6.png)

**Figure: Import JSON content to Flow**

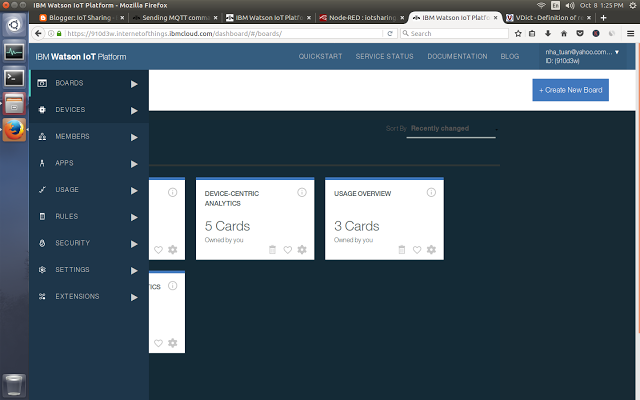
[](https://4.bp.blogspot.com/-SJIsyTQ6nUc/Wdr66PFCN0I/AAAAAAAAEXs/r9GpyAmJhvYLyUkF3DoAkKNOGLqXMK2cQCLcBGAs/s1600/bluemix7.png)

**Figure: Change Device Type and Device Id to yours**

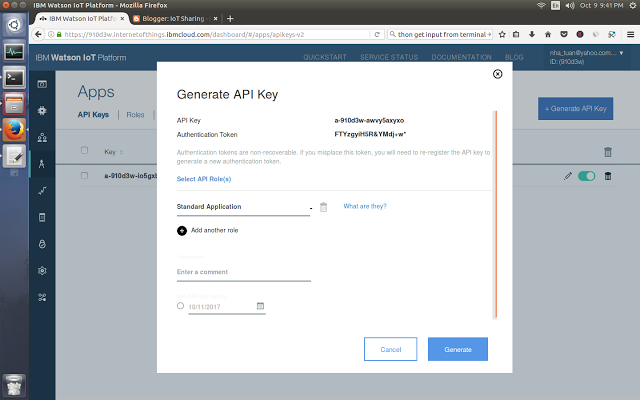
**4.1.2 Result**

**[](https://3.bp.blogspot.com/-dU10yaturHc/WduE7eAy1rI/AAAAAAAAEYM/Z3hSMv4pk2UGl__2AcnVsrrGCfZNxArvgCLcBGAs/s1600/bluemix22.png) Figure: Node-RED receive event from ESP32 IoT in debug tab**

**4.2 Using own developed Python application**  
**4.2.1 Setup**  
We will use PAHO MQTT Python (refer [Demo 14](http://www.iotsharing.com/2017/05/how-to-use-mqtt-to-build-smart-home-arduino-esp32.html)) for our application.  
First, we need to generate API Key for our application.  
From IBM Watson IoT Platform dashboard chose APPS:

[](https://3.bp.blogspot.com/-e1d1oy-Z2gY/WdnFFqmnXEI/AAAAAAAAEVY/lUm2n_G5RZU_ulgP1tlffo1ThO6nCWguwCPcBGAYYCw/s1600/bluemix16.png)

Then choose **Generate API Key**

[](https://2.bp.blogspot.com/-aUBOGmQ3Cpc/WduKu41TdjI/AAAAAAAAEYg/7u5EmYTBiiYms8fAhdKrClHo8gRKKU9ngCLcBGAs/s1600/bluemix25.png)

Remember to save this information for future use.

Here is the full Python code using Paho MQTT

|  |
| --- |
| from random import randint  import thread  import sys  try:  import paho.mqtt.client as mqtt  except ImportError:  import os  import inspect  cmd\_subfolder = os.path.realpath(os.path.abspath(os.path.join(os.path.split(inspect.getfile( inspect.currentframe() ))[0],"../src")))  if cmd\_subfolder not in sys.path:  sys.path.insert(0, cmd\_subfolder)  import paho.mqtt.client as mqtt  ORG = "910d3w"  DEVICE\_TYPE = "ESP32\_IoTSharing"  APP\_ID = "iotsharingdotcom" #choose any string  TOKEN = "86a2mx(YZaYo0\*&)B6"  DEVICE\_ID = "A09E02A4AE30"  server = ORG + ".messaging.internetofthings.ibmcloud.com";  pubTopic = "iot-2/type/" + DEVICE\_TYPE + "/id/" + DEVICE\_ID + "/cmd/test/fmt/String";  subTopic = "iot-2/type/+/id/+/evt/+/fmt/+";  authMethod = "a-910d3w-io5gxbfau6";  token = TOKEN;  clientId = "a:" + ORG + ":" + APP\_ID;  def on\_connect(mqttc, obj, flags, rc):  print("rc: "+str(rc))  def on\_message(mqttc, obj, msg):  print(msg.topic+" "+str(msg.qos)+" "+str(msg.payload))  def on\_publish(mqttc, obj, mid):  print("mid: "+str(mid))  def on\_subscribe(mqttc, obj, mid, granted\_qos):  print("Subscribed: "+str(mid)+" "+str(granted\_qos))  def on\_log(mqttc, obj, level, string):  print(string)  mqttc = mqtt.Client(client\_id=clientId)  mqttc.on\_message = on\_message  mqttc.on\_connect = on\_connect  mqttc.on\_publish = on\_publish  mqttc.on\_subscribe = on\_subscribe  mqttc.username\_pw\_set(authMethod, token)  mqttc.connect(server, 1883, 60)  mqttc.subscribe(subTopic, 0)  def ledControl( threadName, delay):  while True:  val = raw\_input('Enter on or off ')  if(val == "on"):  mqttc.publish(pubTopic, "1")  else:  mqttc.publish(pubTopic, "0")  try:  thread.start\_new\_thread( ledControl, ("ledControl", 0, ) )  except:  print "Error: unable to start thread"  mqttc.loop\_forever() |