# A developer's guide to the Internet of Things (IoT)- Coursera

DJ's notes

2019

# Contents


**1 Introduction** **3**

1.1 What is IoT? . . . 3

1.2 What is the value of IoT? . . . 3

1.3 Why is IoT so special ? . . . 3

1.4 How does IoT works . . . 3

**2 Rapid application development in the cloud** **4**

2.1 Cloud computing . . . 4

2.2 IBM computing . . . 4

2.3 Rapid application for IoT . . . 4

2.4 Introduction to NodeRED . . . 6

2.5 Additional NodeRED nodes . . . 10

**3 Rapid application development on a Raspberry Pi** **10**

3.1 Rasberry Pi and SenseHAT . . . 10

3.2 NodeRED on Raspberry Pi part 1 . . . 12

3.3 NodeRED on Raspberry Pi part 2 . . . 13

3.4 Introduction to the Watson IoT Platform . . . 14

3.5 Controlling a device . . . 18

**4 Lower level programming for the Internet of Things** **19**

4.1 Watson IoT APIs . . . 19

4.2 Message Queue Telemetry Transport (MQTT) . . . 19

4.3 Deploying Applications to the IBM Cloud . . . 20

4.4 Wrap-up . . . 20

**5 Error FAQ** **21**

# 1 Introduction

## 1.1 What is IoT?

- IoT is a network of interconnected things, objects, devices, and systems that connects and transmit data and exchange data among them.
- The data exchanged is collected, analyzed and acted on them.

## 1.2 What is the value of IoT?

- Collecting data from IoT based devices (such as wearables and smart appliances) enables business to learn more about their operating environment.
- This way, businesses can identify and act on the potential to create new value.
- Value by unlocking your revenue from existing products and services, value by inspiring new working practices and processes, value by changing or creating new business models or strategies.
- The potential of Internet of Things lies on the intelligence.

## 1.3 Why is IoT so special ?

- Standard connectivity which can send data to the cloud from anywhere.
- Discrete industries come together
- Provide a service

## 1.4 How does IoT works

- How does IoT work? `https://www.youtube.com/watch?v=QSIPNhOiMoE&feature=youtu.be`
- The Future of IoT at Work. `https://www.youtube.com/watch?v=4jjcznMXF8M&index=4&list=PLBFOHYVTEVoDzBoFYC9PJq-pUT_YkdeOg`

# 2 Rapid application development in the cloud

## 2.1 Cloud computing

1. Infrastructure-as-a-Service (IaaS)
2. Platform-as-a-Service (PaaS)
3. Software-as-a-Service (SaaS)
4. Fuction-as-a-Service (FaaS)

## 2.2 IBM computing

1. Get IBM cloud Lite account for 6months to get access to Bluemix
   `https://e5.onthehub.com/WebStore/Security/LtiSignIn.ashx`
2. Apply code
   `https://console.bluemix.net/account/billing?accountId=e6269f255a404d6cb1a5849dae2f9510`
3. IBM cloud Identity and access management (IAM)
4. Cloud Foundry Orgs, add a region. `https://console.bluemix.net/account/organizations?accountId=e6269f255a404d6cb1a5849dae2f9510`
5. Note: IBM bluemix is now IBM cloud. The core components are the same, but some settings and docs are different.

## 2.3 Rapid application for IoT

1. **Deploying Node-RED on the IBM Cloud**
   a Catalog `https://console.bluemix.net/catalog`
   b Node-RED editor guide `https://nodered.org/docs/user-guide/editor`
   c Go to Node Red (different from course video probably due to update) `https://console.bluemix.net/catalog/starters/node-red-starter`
   d Enable `https://console.bluemix.net/apps/23d84272-669c-44a2-91a3-f009a176740e?paneId=overview&env_id=ibm:yp:eu-gb`
   e Create Platform API key `https://console.bluemix.net/iam/#/apikeys`
   f Go to Git profile and create an access token (or ssh key) `https://git.eu-gb.bluemix.net/profile/personal_access_tokens`

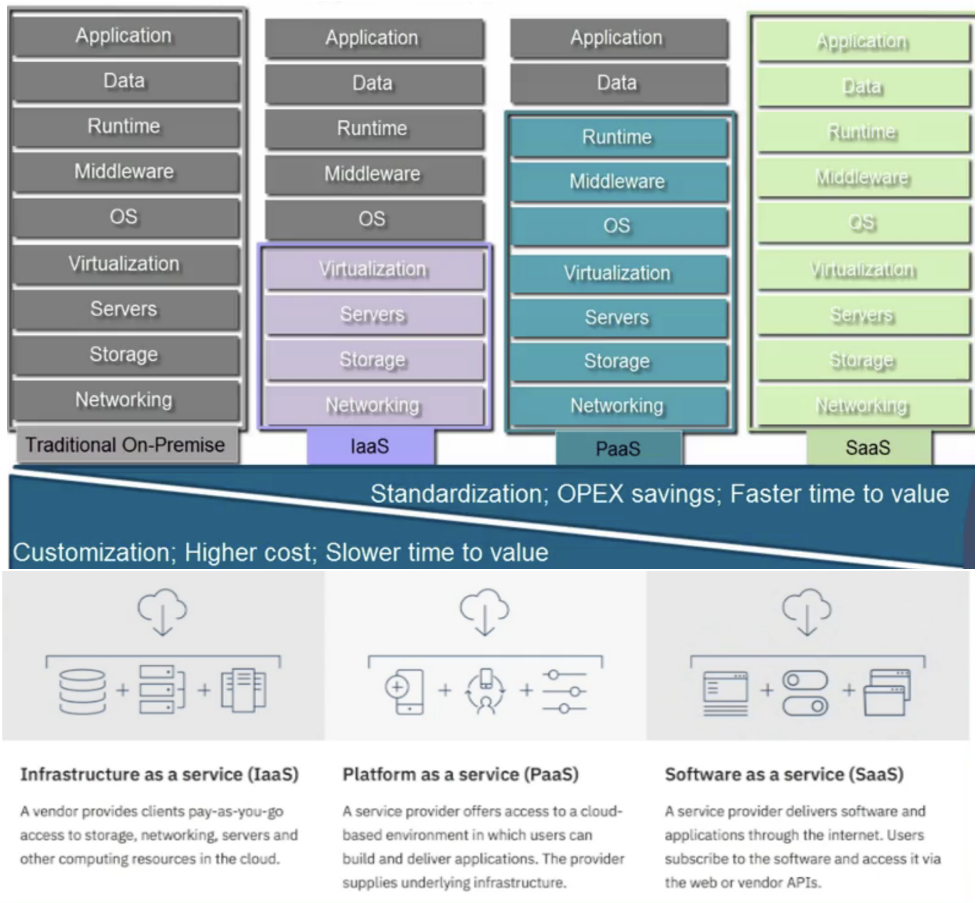



Figure 1: Cloud Computing

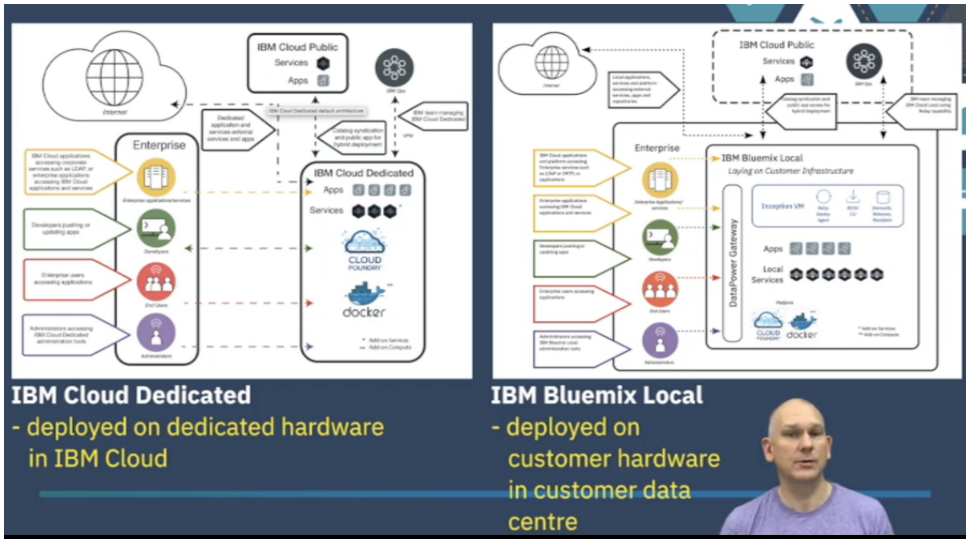



Figure 2: IBMcloud

g Copy and save the token

h By default, the project is created as a private repository.

```
// To access the repository on ibm git repository
```

```
git clone https://username@git.eu-gb.bluemix.net/username/jnode.git
// password is the token
```

i Open the cloned project in Atom, make changes in the editor, add commit messages and commit to the master branch. (gitplus package needed)

## 2.4 Introduction to NodeRED

1. **Introduction to NodeRED 1**

   a Access NodeRED `https://jnode.eu-gb.mybluemix.net`

   b Provide a username and password

   c Follow the guide and enter `https://jnode.eu-gb.mybluemix.net/red/#flow/e7bfdaf.a4b2b28`

2. **Introduction to NodeRED 2**

   a Nodes works by sending JavaScript messages.

      i. Messages flow along the connectors between nodes

      ii. When javascript arrives, it is made available as the msg variable.

      iii. The most important content in a message is found in the payload property

   b To change settings of a node, double click the node.

   c Where are the nodes stored?

      i. Cloud: NoSQL

      ii. Raspberrypi: Local

   d How to store the nodes on the NodeRED flow editor user interface?

      i. Up right: Menu/Export and save as a library

      ii. Local board: click on sheet then dragging over required nodes and save as text files

   e How to import the nodes on the NodeRED user interface?

      i. Up Right: Menu/Import text files

      ii. **Passwords** are not saved when the import and export flows. Need to reinput any passwords stored in the flow.

   f How to install nodes?

      i. Menu/Manage palette

ii. npm install `https://github.com/node-red/node-red`

## 3. Introduction to NodeRED 3

### a Import a function node example

```
// 0. Copy code in references/week2_NodeRED function node_p1.pptx and
    paste on the node editor.
// A. You must return a javascript object.
    return {'hello j'}; // return string in javascript object. Correct!
    return 'hello j', // return string object. An error message will show
        up on the Sidebar/Debug ("Function tried to send a message of type
        string")
// B. How does the functional node handle a Javascript array object?
      var msg1 = { payload:"first message" };
      var msg2 = { payload:"second message" };
      var msg3 = { payload:"third message" };
      var msg4 = { payload:"fourth message" };
      return [msg1, msg2, msg3, msg4] // An array of four objects.
       // If one output is set, only the first message is sent. See
           "Array".
       // If three outputs are set, three messages. See "Multiple returns".
     return [msg1, [msg2, msg3], msg4] // Combine an array within an array
       // Set three outputs, msg1 to output1, msg2+3 to output2, msg4 to
           output3
       // Why is this functionality useful? It allows splitting a large
           set of data into separate messages.
    return [[msg1, msg2, msg3, msg4]] // Enclose an array within an array,
        the function node will separate the items in the inner array, and
        send all of them as separate four messages.
// C. How does the JSON object send data down to node?
      //C1. Create a json object
        msg.payload ={"a": {"b": "test"}};
        return msg;
      //C2. Create a json string
        msg.payload ='{"a": {"b": "test"}}';
         return msg;
      //C3. Create a JSON.parse method on the JSON.string
        msg.payload = msg.payload.a.b;
          return msg; //When you interact with third party sites using
```

```
        APIs, libraries, and packages, it's quite common to get the
        JSON data back as a string.
//C4. Status and logging functions
  //Many APIs in JavaScript use function callbacks. Therefore, we
      can all an asynchronous function, and when that function
      completes, it will then carry on the flow.
  node.status({text:"started"});
  setTimeout(function() {
      node.status({fill:"red",shape:"ring",text:"stage 1"});
      setTimeout(function() {
          node.status({fill:"green",shape:"dot",text:"stage 2"});
          node.error("This is an error", {'payload' : 'Caught
              error'}); //catchnode: Catch for a specific or set of
              nodes on this sheet.
          setTimeout(function() {
              node.error("Second error"); //fire an error message,
                  go to debugger and system console.
              node.status({});
              node.send({'payload' : 'Completed'});
              }, 1000); //Set a timer (1000=1s)
      }, 1000);
  }, 1000);

  node.warn("Exiting node");
  return;
```

b **Store and share nodes**

i. Guide

`https://nodered.org/docs/writing-functions#storing-data`

ii. Store data using context object

A. Context guide `https://nodered.org/docs/user-guide/context`

B. Data is saved in memory and will not survive across restarts of the system

C. Context scopes

- context.: between connected nodes
- flow.: on the sheet
- global.: all the nodes and sheets

D. Method

- .set: set a value
- .get: get a value

c **Run new packages**

i. Node.js: I can easily bring a new package and use it in my application by using the require keyword.

ii. In NodeRED: your function code runs in a sandbox which is not allowed to use the required keyword. So how to add a package your function node wants to use to the global system?

- In Atom, check jnode/bluemix-settings.js
- In Atom, check jnode/packages.js
- Push to git
- On the Note-RED sheet, add a function, edit function node

---

```
//Add an inject (timestamp)
// Add a function node
  var crc = global.get('crc'); // package crc
  msg.payload = crc.crc32('hello').toString(16);
  return msg;
// Connect to "entire message"
// Get access to the package
```

---

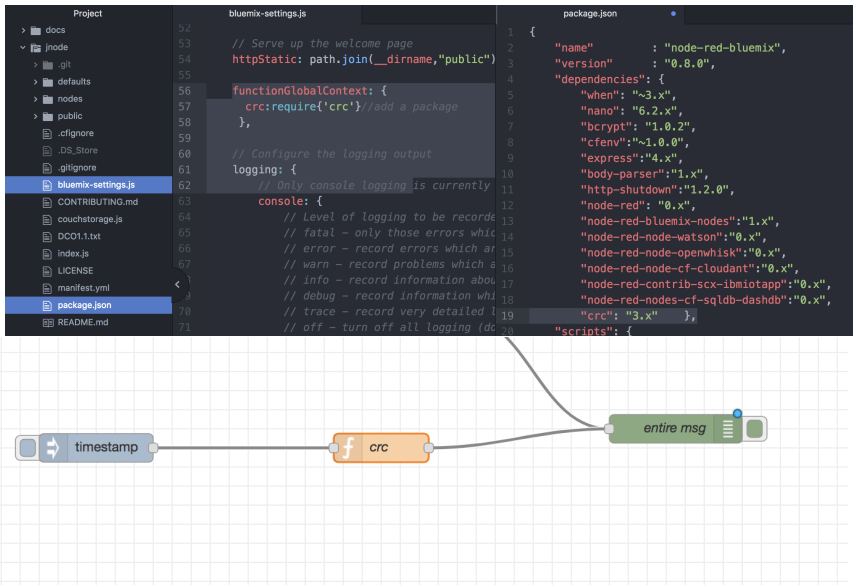



Figure 3: Add a package

### 2.5 Additional NodeRED nodes

# 3 Rapid application development on a Raspberry Pi

## 3.1 Rasberry Pi and SenseHAT

1. A quick look at devices and sensor options

   a GPIO: general purpose input output

   b Grovekit

   c Message Queue Telemetry Transport (MQTT)

   d Raspberry Pi `https://www.raspberrypi.org`

   e Sense HAT: the board has sensors for gyroscope, accelerometer, magnetometer, temperature, barometric pressure, and humidity.

   f What do you need?

   - power supply
   - HDMI cable + monitor
   - keyboard
   - mouse
   - SD card

2. Setting up a Raspberry Pi

   a Overwrite format microSD card using SD Card formatter (Quick format in the future)

   b Format to MS_FAT (Mac Disk Utility)

   c Download Rasbian Stretch 4.14 (with desk top and recommended software)

   d Install Rasbian .zip file using Etcher `https://www.raspberrypi.org/documentation/installation/installing-images/README.md`

```
// Enter RPI shell, CTRL+ALT
// Get IP address
    ifconfig //or
    hostname -I
// Enable connectivity
    sudo raspi-config // Select "5 Internet Options"
    Enable ssh, VNC (allow connect to the RPi UI remotely). SPI (sensors),
        GPIO (40 pins), 1-Wire interface (low speed data) and camera
```

```
// Connect to ssh
    ssh pi@ ip address
// Select"Host name" to change a name
// Reboot RPI
    sudo reboot -n
    sudo rpi-update
    sudo reboot -n
// Update Raspbian OS
    sudo apt-get update //download
    sudo apt-get -y upgrade //install
    sudo apt-get -y autoremove //remove installation packages
    // Update Node.js and nodered
    update-nodejs-and-nodered
    // Allow node applications to run bluetooth scans
    sudo setcap cap_net_raw+eip $(eval readlink -f `which node`) //Use back
        quotes here, not single quotes
    // Install a text editor
    sudo apt-get install -y geany
    //Reboot
    sudo reboot -n
```

Check Error FAQ

e Remote access to RPI from mac (make sure RPI ssh and VNC are enabled and RPI is rebooted ) `https://www.raspberrypi.org/documentation/remote-access`

```
// In Mac
   // Download and install VNC viewer and log in to my account
   // Download and Install VNC server and set a password

// In Pi
    // Install the VNC server
        sudo apt-get install tightvncserver //tightserver //password length
            of 8 ideally, otherwise truncated, but typing the same
    // Start the VNC server from the terminal
        vncserver :1 -geometry 1920x1080 -depth 24 // full HD resolution
    // Restart a server at next boot, and every subsequent boot
        sudo systemctl enable vncserver-x11-serviced.service
    // Stop VNC Server
        sudo systemctl stop vncserver-x11-serviced.service
```

```
// Download VNC viewer log in to my account
// Start vnc server
    vncserver  //New desktop is raspberrypi:1 (ipaddress:1)
// Enter password

// Remote connect to Mac VNC server
```

## 3.2 NodeRED on Raspberry Pi part 1

```
// Open NodeRED
    node-red-start
// Web brower
127.0.0.1:1880
// Open terminal
    cd /home/j/.node-red
    vim settings.js //install vim first
// Add crc package
    functionGlobalContext: {
        crc:require("crc")}
// Install package
    sudo npm install crc
// For changes to take effect
    node-red-stop
    node-red-start
// Add a string CRC calculation function in Node-RED sheet
    var crc = global.get('crc');
    msg.payload = crc.crc32('good day').toString(16);
    return msg;
    // Add an inject input node, payload ==string
    // Add debug output node
 // Close node-red window
     CTRL + C //does not stop Node-RED running
    node-red-stop // stop Node-RED running
```

Check Error FAQ

```
// In sending binary data across the net, it's often converted to a text format
    cd /home/j/.node-red
```

```
    npm install node-red-node-base64 cd /home/j/.node-red
    node-red-stop
    node-red-start
 //auto start Node-Red at every boot
    sudo systemctl enable nodered.service
```

## 3.3 NodeRED on Raspberry Pi part 2

```
// Hot to run a system command in the node?
        Add a exec functional node
           netstat // Command (for network statistics)
                  -a //Append, here you can add another command or a parameter "-a"
                     //Use sdout to link to debug node
```

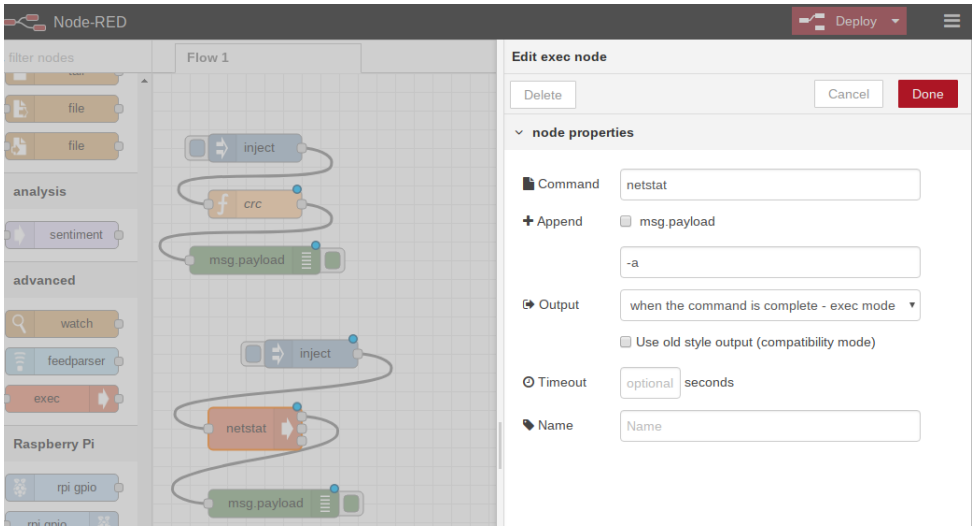



Figure 4: Execute a system command using a node

### IBM IoT quick start service

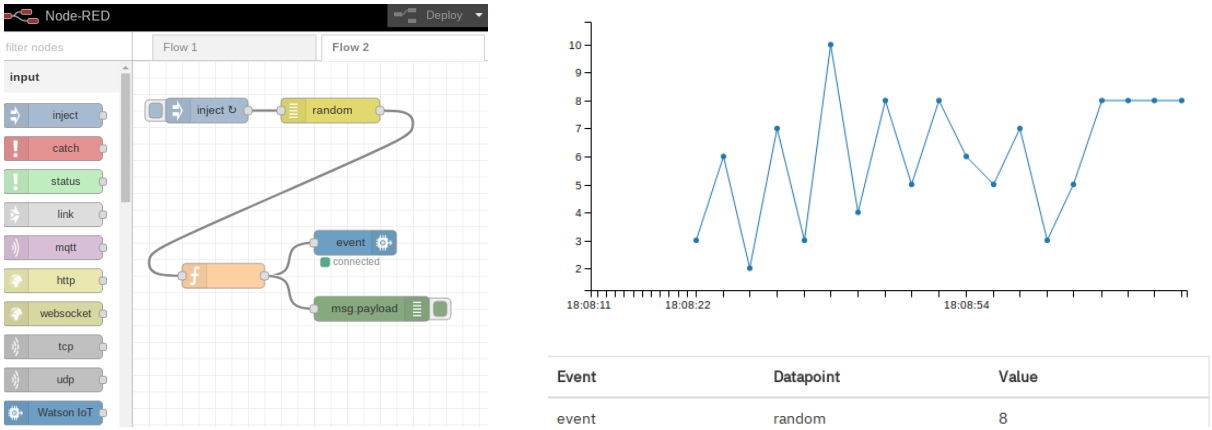
1. Sandbox

   a No security or pre-registration of device

   b Allow publishing data on the application and data visualization

2. Hot to use Output/IBM IoT Device node?

**IoT quick start service**




(a) Quick start node

(b) Data visulization

Figure 5: IBM IoT quick start

```
// Double click and configure quick startup
// Click quickstart id link //Visualization page using the generated device ID
// Add an Input/inject node
    Payload: string; Repeat: interval, every 3 seconds // Configure to send an empty
        string every 3 seconds
// Add a Function/random node to send the random data to the quick start application
// Add a Function node to format the data (for the platform to understand before
    sending the data to Quickstart to visualize the data)
     msg.payload = {"d" : {"random" : msg.payload}};
     return msg;
// Add an Output/debug node
```

## 3.4 Introduction to the Watson IoT Platform

1. Wason IoT

   a Device Registry

   - Ensures that all data received is from a trusted device
   - Holds security and metadata on all your devices in one place

   b Connectivity

   - Uses MQTT, the MQ Telemetry Transport protocol as the underlying mechanism
   - To allow devices to communicate with the platform

   c Gateway devices support

d Device management

e External services integration (e.g. Jasper)

f An API that allows access to configuration and historic data

g `bluemix.net`

**Go through the IoT Platform** (New updates in the platform)

```
// Usage (old Overview)
// Go to IoT dash board (or use API access) //
    https://console.bluemix.net/services/iotf-service/663081c4-fd39-4c07-af94-9044e1a13275/?pan
//  Launch IoT
   // https://userid.internetofthings.ibmcloud.com/dashboard/settings
   // Organisation Concept (6 digits user id)
        See /Dashboard/Setting/Identity (also in the top corner on the dashboard)
        // https://wm377f.internetofthings.ibmcloud.com/dashboard/settings
```

2. Devices, Applications and Gateways part 1

   a Documentation `https://console.bluemix.net/docs/overview/ibm-cloud-platform.html#whatis`

   b Three types of applications can be connected to the platform

   - Device: Sends events to platform (Create a device type to organise your devices into their capabilities, pre-register a device, and send data to the platform).
   - Gateway: A special type of device. For sensors or sensor networks that aren't able to connect directly to the internet and IOT platform. The Gateway acts as a bridge to connect the sensor network to the internet and the IBM IoT platform. Like a device, a gateway must be registered before it can communicate with the platform.
   - Application: Receives events and and can issue commands to be sent to devices. An application needs to present/ send an API key to authenticate itself).
   - And Events: Data flows in you!

3. Devices, Applications and Gateways part 2

**How to register a RPI device?** `https://console.bluemix.net/docs/services/IoT/iotplatform_task.html#iotplatform_task`

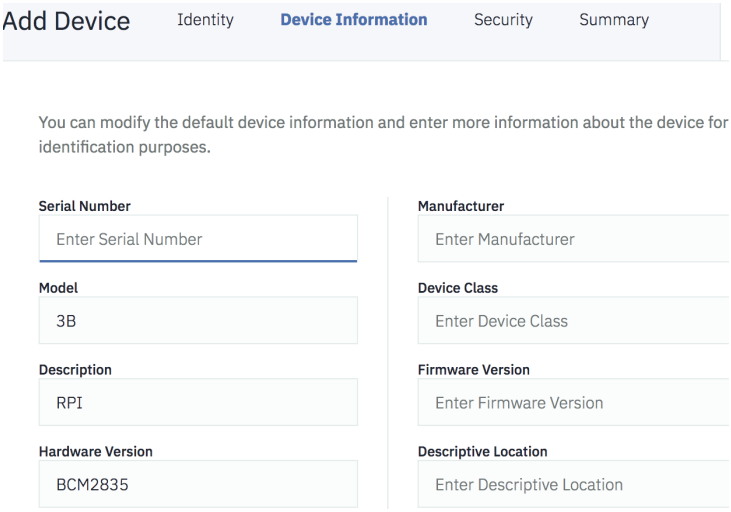
```
// In pi shell
    // Find the serial number
      cat /proc/cpuinfo
// On IoT platform
    Device/Device Types/Add a new device type "RPI"
    Device/Add a device
         Device: RPI
         Device ID: applepi
         Add Meta information (optional. serial number, hardware)
         Security: passw0rd // a generally used authentication token (not for
             production)
```

### How to register an gateway?

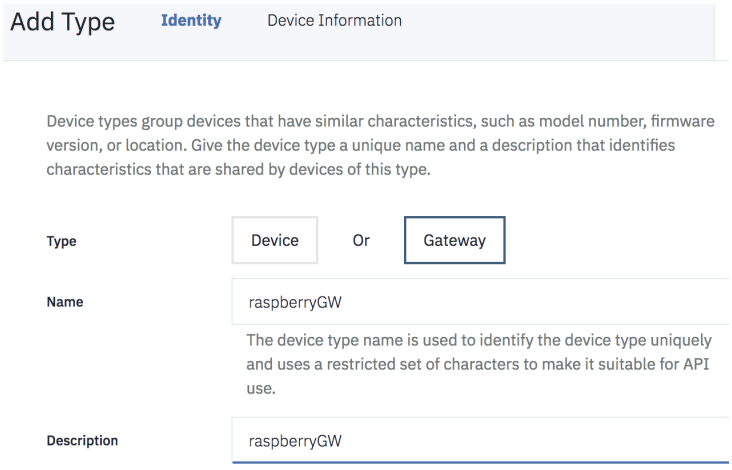
```
// On IoT platform
    Device/Device Types/Add a new device type "raspberryGW"
    Device/Add a gateway
         Device: RPI
         Device ID: applepigw
         Security: passw0rd // a generally used authentication token (not for
             production)
```

### How to generate an API key for an application?

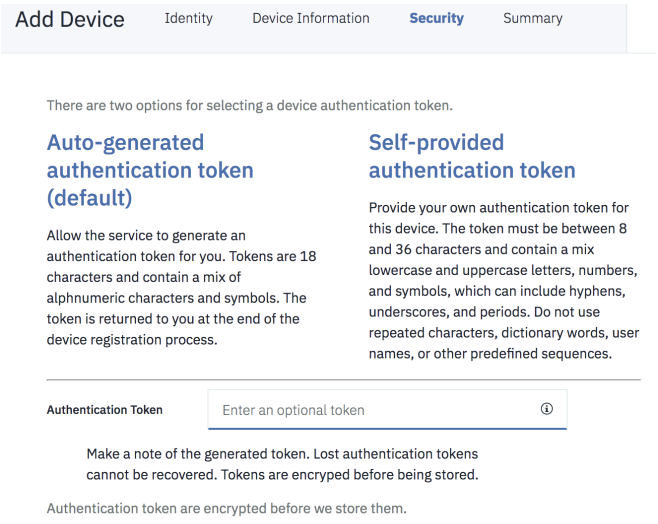
```
// On IoT platform
    Apps/Browse API Keys (Old "Access" in the video)
    Generate a new key
    Description: standardapp1
    Role: Standard application (Multi roles, different from the video)
```




(a) Register a RPI device




(b) Register a Gateway




(c) Generate an API key for an application

Figure 6: Register devices, gateways, and authorise applications

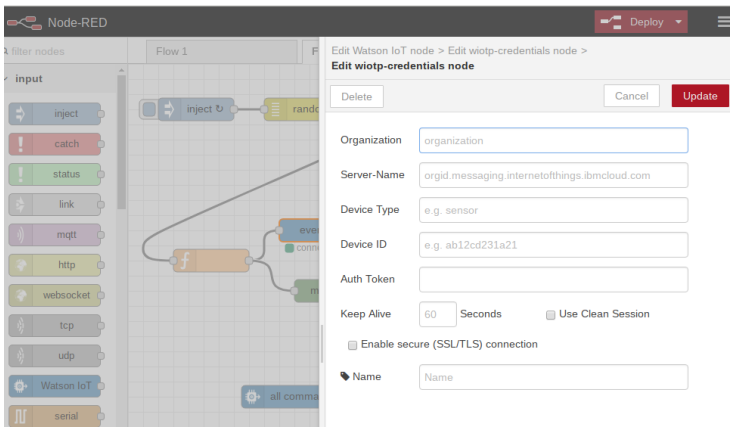
**Manual setting**

- Securely connect a device to an organisation as a registered device type, and publish data to the organisation.

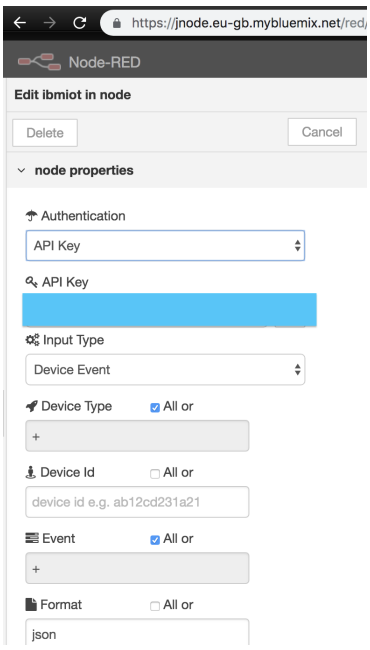
```
// In Pi Node-RED
    // Modify the previous Output/Waston IoT node to connect to our organization
        onto our registered device type
     // Error: "[wiot:connectionPool:getClient] [DeviceClient:publish] Client is
         not connected"
// On Bluemix Node-RED
    // https://jnode.eu-gb.mybluemix.net/red/#flow/f87a5372.e6bdd
    // The IoT node connects to the device as an application
        // Select input/ibmiot node
```

- Bind a cloud application that is running on bluemix, to a service where devices are registered

- Securely connect a gateway to an organisation as a registered device type, and publish data from sensors to the organization.




(a) In RPi: Node-RED IoT node




(b) On IBM Bluemix: NodeRed IoT node

Figure 7: Connect IoT node to a registered device

### 3.5 Controlling a device

1. Sending commands to a device
2. SenseHAT and SenseHAT simulator nodes in NodeRED

## 4 Lower level programming for the Internet of Things

### 4.1 Watson IoT APIs

1. IoT platform APIs
    a Doc `https://console.bluemix.net/docs/services/apiconnect/tutorials/tut_discover_apis.html#discovering-apis`
    b Rest, high-level (python, node.js, and C etc), and low level APIs
2. SenseHATpython APIs
    a Doc `https://pythonhosted.org/sense-hat/api/`
    b Python IDE2 is used in the course
    c Emulator

### 4.2 Message Queue Telemetry Transport (MQTT)

1. An low-level open source protocol that was designed to
    a solve the problems of machine to machine communication
    b be very, very small in code footprint
    c use low power consumption
    d use low network bandwidth
2. A pub-sub system where the sender and receiver are decoupled
    a A broker in the middle and many senders
    b A message can be sent to multiple recipients without the sender needing to know who those recipients are.
        - A sender sends data against a topic to the broker
        - Anybody interested in receiving that data, subscribe to the topic (like a tree!).

- When data is published against that topic, you're going to receive a copy of the message

3. The core of messaging on the IBM IoT platform

   a A broker: is the organisation

   b Data is typically in json format

   c Doc `https://developer.ibm.com/tutorials/cl-mqtt-bluemix-iot-node-red-app`

   d MQTT client identifier format

4. More about MQTT

   a Major design points: Topics and topic structures which are not preregistered.

   b A sender can create topics on the fly, the only limit being that they are less than 220 characters.

   c Use "/ " character to create topic hierarchies

   d Doc `https://opensource.com/article/18/6/mqtt`

### 4.3 Deploying Applications to the IBM Cloud

1. Manage my resources on the IBM cloud `https://cloud.ibm.com/resources`

   a Built on top of the open-source Cloud Foundry platform.

   b When developing an application for a platform as a service, you need to be aware that you have to take how the platform manages your application into account when implementing the solution.

   c With the platform as a service, you don't have any options to go in and configure the lower level operating system or middleware configuration.

   d Don't use any local resources such as the local file system or require any in memory cache within your application runtime.

2. Devops environment

3. The IBM cloud tells you where to listen to traffic. It's also telling you where it's going to monitor your application from

1. It is not typical to do development work on the Pi itself. Setting up the Pi with Predix Guide `https://www.predix.io/resources/tutorials/journey.html#1750`

2. In Mac: create the example application (CloudFoundry command line tool available for the Raspberry Pi.)

3. To deploy the app Use the file manifest.yml

## 4.4 Wrap-up

# 5 Error FAQ

Installation

```
sudo rpi-update
    //Error: Failed to download update for rpi-update! Make sure you have
        ca-certificates installed and that the time is set correctly
        sudo CURL_CA_BUNDLE=/etc/ssl/certs/ca-certificates.crt rpi-update
           Ref: https://github.com/Hexxeh/rpi-update/issues/206
ssh pi@ipadress

//Error: can not access RPI from mac using ssh, VNC
        // In RPI
        Enable ssh, VNC in RPI
        // In Mac // not working
          cd /Users/j.ssh/
          rm known_hosts //https://www.raspberrypi.org/forums/viewtopic.php?t=172423
        //Not solved!!

//In RPI: Can not fetch archives
    sudo apt-get clean
    sudo apt-get update
//In RPI: dpkg interupted
    sudo dpkg --configure -a
```

Run Node-RED

```
// In RPI: Click run Node-RED
    // Failed to execute child process "xterm" (No such file or directory)
    sudo apt install xterm

npm install crc
    //npm notice created a lockfile as package-lock.json and audited 4 packages
       // use sudo npm install crc
```