

Prerequisites:

1. Complete IBM Cloud prerequisites:

<https://github.com/SuperhackMelbourne/superhack2017/blob/master/bluemix/README.md>

Steps:

1. Open your Bluemix dashboard (<https://console.bluemix.net/dashboard>)

2. Click on the service with the “Internet of Things Platform” service offering

Cloud Foundry Services 2/100 Used		
Name ^	Service Offering	Plan
superhack-ay1-cloudantNoSQLDB	Cloudant NoSQL DB	Lite
superhack-ay1-iotf-service	Internet of Things Platf...	Lite

3. Click “Launch” on the next page



Let's get started with Watson IoT Platform

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

Launch

Docs

4. Click on “Devices” in the left sidebar



5. Click on the “Device Types” tab

Device Types

6. Click on “+ Add Device Type” near the top right

+ Add Device Type

7. Select “Gateway” for the type, enter ESP32 for the name and click “Next”

Add Type

Identity

Device Information

×

Select Type

Device types group devices that have similar characteristics, such as model number, firmware version, or location. Give the device type a unique name and a description that identifies characteristics that are shared by devices of this type.

Type

Device

Or

Gateway

Name

ESP32

The device type name is used to identify the device type uniquely and uses a restricted set of characters to make it suitable for API use.

Description

Cancel

Next

8. All the fields are optional, click “Done”

Add Type

Identity

Device Information

×

Device Information

You can enter more information about the device type for identification purposes.

Serial Number

Enter Serial Number

Manufacturer

Enter Manufacturer

Model

Enter Model

Device Class

Enter Device Class

Description

Enter Description

Firmware Version

Enter Firmware Version

Hardware Version

Enter Hardware Version

Descriptive Location

Enter Descriptive Location

+ Add Metadata

◀

Done

9. Click “Register Devices”

Register Device

Advanced Flow

Optional

Register Devices, Define Interfaces

Now that you added a device type, you can register and connect devices for this type.

Register Devices

10. Check the Device Type is ESP32, enter Device ID as hex-board (or any name your like) and click “Next”

Add Device

Identity

Device Information

Security

Summary

×

Identity

Select a device type for the device that you are adding and give the device a unique ID.

Select Existing Device Type

ESP32

Device ID

hex-board

Cancel

Next

11. All fields are optional here, click “Next”

Add Device

Identity

Device Information

Security

Summary

×

Device Information

You can modify the default device information and enter more information about the device for identification purposes.

Serial Number

Enter Serial Number

Model

Enter Model

Description

Enter Description

Hardware Version

Enter Hardware Version

Manufacturer

Enter Manufacturer

Device Class

Enter Device Class

Firmware Version

Enter Firmware Version

Descriptive Location

Enter Descriptive Location

+ Add Metadata

◀

Next

12. Enter a token (don't lose this!) between 8 and 36 characters and click "Next"

Add DeviceIdentityDevice InformationSecuritySummary

Device Security

There are two options for selecting a device authentication token.

Auto-generated authentication token (default)

Allow the service to generate an authentication token for you. Tokens are 18 characters and contain a mix of alphanumeric characters and symbols. The token is returned to you at the end of the device registration process.

Self-provided authentication token

Provide your own authentication token for this device. The token must be between 8 and 36 characters and contain a mix lowercase and uppercase letters, numbers, and symbols, which can include hyphens, underscores, and periods. Do not use repeated characters, dictionary words, user names, or other predefined sequences.

Authentication Token

myhexboardtoken

Make a note of the generated token. Lost authentication tokens cannot be recovered. Tokens are encrypted before being stored.

Authentication token are encrypted before we store them.

<

Next

13. Click "Done"

14. Take note of your device credentials

Device Credentials

You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details.

Organization ID	y1ij2
Device Type	ESP32
Device ID	hex-board
Authentication Method	use-token-auth
Authentication Token	myhexboardtoken

!

Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

14.1 Click on Security on the left. Add a new custom rule. Set the scope to ESP32 and security level to TLS optional. Click "Refresh compliance" and then save

← Back

Cancel

Save

Connection Security

Use the Connection Security policy to set the default security level that is applied to all devices. You can then add custom rules for specific devices. When the default rule and custom rules are defined, you can view the compliance levels for your organization.

1 Device in organization

Refresh compliance

Updated 24 November 2017 21:59

Default Rule

Define the default connection security level to use for all device types that do not have custom rules defined. You can view the number of devices that are affected and then predicted level of compliance.

Note: The device number and predicted compliance values are estimates based on a report that runs at varying intervals.

Scope	Security Level	Predicted Compliance ⓘ	# of Devices
Default	TLS with Token Authentication ▼	<div></div> 0 Pass 0 Fail 0 Unknown	0 devices

Custom Rules

You can define custom connection rules for specific device types. Custom rules overwrite the default rule for the specified device types. The predicted compliance value is updated to reflect the default settings and the custom settings.

Add Custom Rule

ⓘ No available device types

<input type="checkbox"/>	Scope ↕	Security Level ↕	Predicted Compliance ⓘ ↕	# of Devices ↕	
<input type="checkbox"/>	ESP32	TLS Optional ▼	<div></div> 1 Pass 0 Fail 0 Unknown	1	

15. Download and install Mongoose OS from <https://mongoose-os.com/software.html>

16. Download and install CP210x USB to UART Bridge VCP Drivers from <https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

17. Connect battery to hex board

18. Connect hex board to your computer using supplied USB cable

19. Run Mongoose OS, select COM3 in the drop-down and click “Select”. You should see Flash and Set WiFi automatically done.

Device status: online ● ✕

Connect

Plug in your device, choose device address, click Select

COM3

?

Select

Flash

Choose platform, select app, click Flash

esp32

App...

Flash

Mongoose OS app is flashed on this device:
esp32 | Nov17 | 172.20.10.2 | 116k/228k | 83k/155k

Set WiFi

Configure WiFi

AndrewsiPadAir

●●●●●●●●

Set

MONGOOSE OS
mos tool version 1.20.1

Done

20. Click on “Device Files” on the left and “init.js” under the Device File Manager. Ensure that your code looks like the following screenshot. If so, proceed to Step 21. If not, copy and paste the code below into your screen and click “Save + Reboot”. You can replace hex-board with your own device ID if you have set your own one

Device File Manager

esp32 | Nov17 | 172.20.10.2 | 116k/228k | 83k/155k
 ● device setup | dashboard

Device Files

File	Size
ca.pem	14308
conf0.json	4743
conf9.json	446
init.js	462
mgos_config_schema.json	13780
mgos_ro_vars_schema.json	332

Refresh File List

Save File

Save + Reboot

```

1 load('api_mqtt.js');
2 load('api_gpio.js');
3
4 let pin = 32, topic = 'iot-2/type/ESP32/id/hex-board/cmd/update/fmt/json';
5
6 GPIO.set_button_handler(pin, GPIO.PULL_UP, GPIO.INT_EDGE_POS, 200, function() {
7   MQTT.pub('iot-2/type/ESP32/id/hex-board/evt/event/fmt/json', JSON.stringify({ message: "I need he
8 }, null);
9
10 MQTT.sub('iot-2/type/ESP32/id/hex-board/cmd/update/fmt/json', function(conn, topic, msg) {
11   print('Topic:', topic, 'message:', msg);
12 }, null);
13
          
```

Code:

```
load('api_mqtt.js');
```

```
load('api_gpio.js');
```

```
let pin = 32, topic = 'iot-2/type/ESP32/id/hex-board/cmd/update/fmt/json';
```

```
GPIO.set_button_handler(pin, GPIO.PULL_UP, GPIO.INT_EDGE_POS, 200, function() {
```

```
  MQTT.pub('iot-2/type/ESP32/id/hex-board/evt/event/fmt/json', JSON.stringify({ message: "I need help!"}));
```

```
}, null);
```

```
MQTT.sub('iot-2/type/ESP32/id/hex-board/cmd/update/fmt/json', function(conn, topic, msg) {  
  
    print('Topic:', topic, 'message:', msg);  
  
}, null);
```

21. Click on “Device Config” on the left and set the MQTT server to xxxxxx.messaging.internetofthings.ibmcloud.com:1883, where xxxxxx is your Organization ID from Step 14. Ensure User is use-token-auth and set the password to your token from Step 12. Click “Save configuration”

The screenshot shows the 'Device Configuration' window. On the left is a sidebar with icons for Projects, Device Files, Device Config (highlighted), Device Service, Terminal, and Forum. The main area has a title bar 'Device Configuration' and two tabs: 'Simple View' (selected) and 'Expert View'. Below the tabs are two sections: 'Debug Settings' and 'MQTT Settings'. In 'Debug Settings', 'Hexdump traffic:' has an unchecked checkbox, and 'Log level:' is set to '2'. In 'MQTT Settings', 'Enable MQTT' has a checked checkbox. Below this are four input fields: 'MQTT server' with the value 'y1ij2.messaging.internetofthings.ibmcloud.cc', 'User' with 'use-token-auth', and 'Password' with 'myhexboardtoken'.

Device Configuration	
<div>Save configuration Simple View Expert View</div>	
Debug Settings	
Hexdump traffic:	<input type="checkbox"/>
Log level:	2
MQTT Settings	
Enable MQTT	<input checked="" type="checkbox"/>
MQTT server	y1ij2.messaging.internetofthings.ibmcloud.cc
User	use-token-auth
Password	myhexboardtoken

22. Click on “Expert View” and at line 63, change client ID to g:xxxxxx:ESP32:yyyyyy where xxxxxx is your Organization ID from Step 14 and yyyyyy is your device ID from Step 10. Click “Save configuration”

Device Configuration esp32

Save configuration Simple View Expert View

```
54     "enable": false,
55     "freq": 100000,
56     "debug": false,
57     "sda_gpio": 32,
58     "scl_gpio": 33
59 },
60 "mqtt": {
61     "enable": true,
62     "server": "y1iij2.messaging.internetofthings.ibmcloud.com:1883",
63     "client_id": "g:y1iij2:ESP32:hex-board",
64     "user": "use-token-auth",
65     "pass": "myhexboardtoken",
```

23. Open your Bluemix dashboard (<https://console.bluemix.net/dashboard>) and click on the link of your Cloud Foundry App

Cloud Foundry Apps 256 MB/256 MB Used

Name	Route	State
superhack-ay1	superhack-ay1.myblue...	Awake (1/1)

24. Click the “hamburger” icon on the top right and click “Manage palette”



25. Click the “Install” tab and search for slack

User Settings Close

View: Nodes Install

Keyboard: sort: a-z recent ↺

🔍 slack 9 / 1199

Palette

- node-red-contrib-loose** [🔗](#)
Node Red Slack client
📅 0.0.2 📅 2 years ago install
- node-red-contrib-slack** [🔗](#)
A node-red module to post to Slack.com
📅 0.1.2 📅 10 months ago install

26. Install node-red-contrib-slack by clicking “Install”

27. Click “Install”

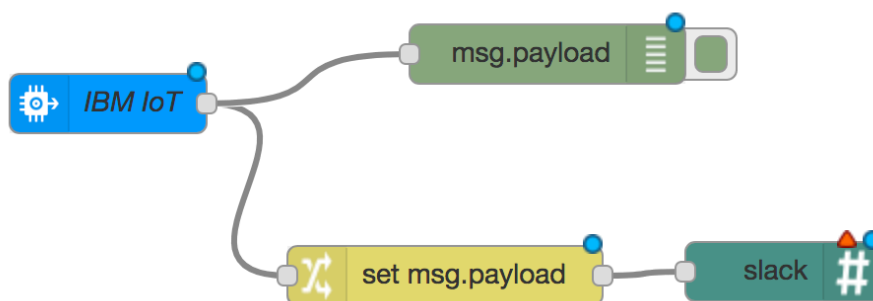
Install nodes

Before installing, please read the node's documentation. Some nodes have dependencies that cannot be automatically resolved and can require a restart of Node-RED.

Cancel
Open node information
Install

28. The installation will fail for the first time. Click “Install” again and it will complete successfully. After the installation completes, click “Close” and click “+” to create a new flow

29. From the left, drag and drop the ibmiot, change, debug and slack nodes into the canvas and connect them together as per diagram below



30. Double-click the IBM IoT node, change Authentication to Bluemix Service and enter your device ID

Edit ibmiot in node

Delete
Cancel
Done

node properties

Authentication

Bluemix Service

Input Type

Device Event

Device Type

☒ All or

+

Device Id

☐ All or

hex-board

Event

☒ All or

+

Format

☐ All or

json

QoS

0

Name

IBM IoT

31. Double-click the change node and modify the “to” field to msg.payload.message

Edit change node

Delete
Cancel
Done

▼ node properties

📌 Name
Name

☰ Rules

Set
▼ msg. payload

to
▼ msg. payload.message

32. Double-click the slack node, fill in the WebHook URL with <https://hooks.slack.com/services/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>, the UserName with your team name/number. Optionally set an emoji icon and name.

Edit slack node

Delete
Cancel
Done

▼ node properties

⚙️ WebHook URL
https://hooks.slack.com/services/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

👤 Posting UserName
Team-1

😊 Emoji Icon
:baby:

🔗 Channel
optional channel override

📌 Name
Get help

33. Click “Deploy”

34. Join the #help channel in the SuperHack2017.slack.com workspace

35. Press and release the button “USER BTN 1” on the hex board. You should see a message in the Slack channel requesting for help.

Team-1 APP 10:02 PM
new messages

😊 I need help!

+ Message #help
@
😊

36. Congratulations, you have finished the first tutorial!