Internet of Things @UL

IoT Summer School

Node-RED

Lab

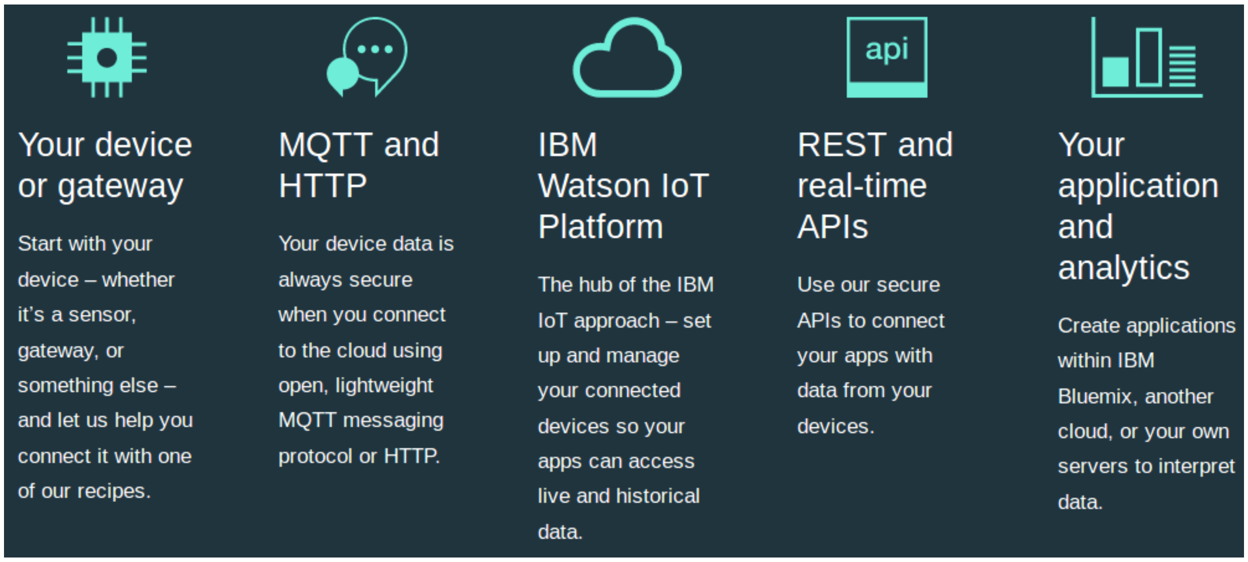
Table of Contents

1 IBM IoT

1.1 Bluemix

Bluemix is a fully managed, cloud-hosted service designed to simplify and derive value from your IoT devices. Companies are using Watson Internet of Things to transform their business from the inside out.

Bluemix is an implementation of IBM's Open Cloud Architecture based on Cloud Foundry, an open source Platform as a Service (PaaS). Bluemix delivers enterprise-level services that can easily integrate with your cloud applications without you needing to know how to install or configure them.



Your device or gateway

Start with your device – whether it's a sensor, gateway, or something else – and let us help you connect it with one of our recipes.

MQTT and HTTP

Your device data is always secure when you connect to the cloud using open, lightweight MQTT messaging protocol or HTTP.

IBM Watson IoT Platform

The hub of the IBM IoT approach – set up and manage your connected devices so your apps can access live and historical data.

REST and real-time APIs

Use our secure APIs to connect your apps with data from your devices.

Your application and analytics

Create applications within IBM Bluemix, another cloud, or your own servers to interpret data.

IBM Watson Platform

IBM Watson IoT Platform can help you get a quick start on your next Internet of Things project. It is a fully managed, cloud-hosted service designed to make it simple to derive value from your Internet of Things devices. It provides capabilities such as device registration, connectivity, control, rapid visualization and storage of Internet of Things data.

By combining IoT data with cognitive computing, businesses can extract valuable insights to improve virtually every aspect of their operations and enable innovative, new business models.

Node-RED

Node-RED is a tool for wiring together the Internet of Things in new and interesting ways, including hardware devices, APIs, and online services. It is built on top of Node.js and takes advantage of the huge node module ecosystem to provide a tool that is capable of integrating many different systems. Its lightweight nature makes it ideal to run at the edge of the network.

2 Background Knowledge

Reading this section is optional, the lab can be safely completed without reading it.

2.1 Extract Transform Load

The three words in Extract Transform Load each describe a process in the moving of data from its source to a formal data storage system (most often a data warehouse).

- **Extract**—The extraction process is the first phase of ETL, in which data is collected from one or more data sources and held in temporary storage where the subsequent two phases can be executed. During extraction, validation rules are applied to test whether data has expected values essential to the data warehouse. Data that fails the validation is rejected and further processed to discover why it failed validation and remediate if possible.

- **Transform**—In the transformation phase, the data is processed to make values and structure consistent across all data. Typical transformations include things like date formatting, resorting rows or columns of data, joining data from two values into one, or, conversely, splitting data from one value into two. The goal of transformation is to make all the data conform to a uniform schema.

- **Load**—The load phase moves the transformed data into the permanent, target database. Once loaded, the ETL process is complete, although in many organizations ETL is performed regularly in order to keep the data warehouse updated with the latest data.

Why do you need ETL?

ETL takes data that is heterogeneous and makes it homogeneous. Without ETL it would be impossible to programmatically analyze heterogeneous data and derive business intelligence from it.

What is the difference between heterogeneous and homogeneous?

A DDBMS may be classified as homogeneous or heterogeneous. In a homogeneous system, all sites use the same DBMS product. In a heterogeneous system, sites may run different DBMS products, which need not be based on the same underlying data model, and so the system may be composed of relational, network, hierarchical and object-oriented DBMSs.

Homogeneous systems are much easier to design and manage. This approach provides incremental growth, making the addition of a new site to the DDBMS easy, and allows increased performance by exploiting the parallel processing capability of multiple sites.

2.2 Cloudant

Cloudant is an IBM software product, which is primarily delivered as a cloud-based service. Cloudant is a non-relational, distributed database service of the same name. Cloudant is based on the Apache-backed CouchDB project and the open source BigCouch project.

Cloudant's service provides integrated data management, search, and analytics engine designed for web applications. Cloudant scales databases on the CouchDB framework and provides hosting, administrative tools, analytics and commercial support for CouchDB and BigCouch.

2.2.1 Cloudant Geospatial

Cloudant Geospatial combines the advanced geospatial queries of a Geographic Information System (GIS) with the flexibility and scalability of Cloudant's database-as-a-service (DBaaS) capabilities.

Cloudant Geo:

- Enables web and mobile developers to enhance their applications using geospatial operations that go beyond simple bounding boxes.
- Integrates with existing GIS applications, so that they can scale to accommodate different data sizes, concurrent users, and multiple locations.
- Provides a NoSQL capability for GIS applications, so that large streams of data can be acquired from devices, sensors, and satellites. This data can then be stored, processed, and syndicated across other web applications.

2.2.2 Data flexibility: JSON data store

The majority of requests and responses to and from Cloudant use the JavaScript Object Notation (JSON) for formatting the content and structure of the data and responses.

JSON is used because it is the simplest and easiest solution for working with data using a web browser. This is because JSON structures can be evaluated and used as JavaScript objects within the web browser environment. JSON also integrates with the server-side JavaScript used within Cloudant.

Cloudant's RESTful API makes every document in your database accessible as JSON. It is also compatible with Apache CouchDB™, enabling you to access an abundance of language libraries and tools. Schema flexibility makes Cloudant an excellent fit for multi-structured data, unstructured data and fast-changing data models. https://www.ibm.com/analytics/us/en/technology/cloud-data-services/cloudant/

2.3 JSON

JSON, or JavaScript Object Notation, is a minimal, readable format for structuring data. It is used primarily to transmit data between a server and web application.

2.3.1 Keys and Values

The two primary parts that make up JSON are keys and values. Together they make a key/value pair.

- **Key:** A key is always a string enclosed in quotation marks.
- **Value:** A value can be a string, number, boolean expression, array, or object.
- **Key/Value Pair:** A key value pair follows a specific syntax, with the key followed by a colon followed by the value. Key/value pairs are comma separated.

Let's take one line from a JSON sample and identify each part of the code.

```
"foo" : "bar"
```

This example is a key/value pair. The key is "foo" and the value is "bar".

2.3.2 Objects

An object is indicated by curly brackets. Everything inside of the curly brackets is part of the object. "foo" and the corresponding object are a key/value pair.

```
"foo" : {

  "bar" : "Hello"

}
```

The key/value pair "bar" : "Hello" is nested inside the key/value pair "foo" : { ... }. That's an example of a hierarchy in JSON data. https://developers.squarespace.com/what-is-json/

2.3.3 Arrays

```
"foo" : {

  "bar" : "Hello",

  "baz" : [ "quuz", "norf" ]

}
```

3 Node-RED

Node-RED provides a browser-based flow editor that makes it easy to wire together flows using the wide range of nodes in the palette. Flows can be then deployed to the runtime in a single-click. While Node-Red is based on Node.js, JavaScript functions can be created within the editor using a rich text editor. A built-in library allows you to save useful functions, templates or flows for re-use

Node-RED is included in the Node-RED starter application in Bluemix (Bluemix is IBM's Platform as a Service, free of charge) but you can also deploy it as a stand alone Node.js application. Node-RED can not only be used for IoT applications, but it is a generic event-processing engine. For example you can use it to listen to events from http, websockets, tcp, Twitter and more and store this data in databases without having to program much if at all.

3.1 Flows

Node-RED programs or flows are a collection of nodes wired together to exchange messages. Under the hood, a flow consists of a list of JavaScript objects that describe the nodes and their configurations, as well as the list of downstream nodes they are connected to, the wires.

3.2 Messages

Messages passed between nodes in Node-RED are, by convention, JavaScript Objects called msg, consisting of a set of named properties. These messages often contain a msg.payload property with the payload of the message. Nodes may attach other properties to a message, which can be used to carry other information onto the next node in the flow. When this happens, these extra properties will be documented in the node documentation that appears in the node info pane when you select a node in the Node-RED workspace.

Messages are the primary data structure used in Node-RED and are, in most cases, the only data that a node has to work with when it is activated. This ensures that a Node-RED flow is conceptually clean and stateless – each node is self-contained, working with input messages and creating output messages. Apart from the use of context data (see later in this lecture), this means that the effect of a node's processing is either contained in its output messages, or caused by internal node logic that changes external things such as files, IO pins on the Raspberry Pi or Dropbox files; there are no side effects that could affect the behaviour of other nodes or subsequent calls to the same node.

3.3 Nodes

Nodes are blocks that represent components of a larger system, in Node-RED's case usually the devices, software platforms and web services that are to be connected. Further blocks can be placed in between these components to represent software functions that wrangle and transform the data in transit.

The sets of available nodes differ, Bluemix has extra nodes for DB access, but does not expose the File nodes. Node-RED in Bluemix stores its persistent data (flows, libraries, credentials) in the co-installed Cloudant database named nodered. When using a Cloudant node with Node-RED on BlueMix, the list of available instances is automatically listed. Node-RED in Bluemix has built-in credential management, so you don't have to worry about exposing your services authentication data, they will be filled-in automatically from the services' credentials defined for the application in Bluemix.

there are three core node types:

- Input nodes – generate messages for downstream nodes.
- Output nodes – consume messages, for example to send data to an external service or pin on a device, and may generate response messages.
- Processing nodes – messages that process data in some way, emitting new or modified messages.

3.4 Nodes used in this lab

3.4.1 Inject node

The Inject node allows you to inject messages into a flow, either by clicking the button on the node, or setting a time interval between injects.

3.4.2 Function node

The Function node allows JavaScript code to be run against the messages that are passed in and then return zero or more messages to continue the flow.

The message is passed in as an object called `msg`. By convention it will have a `msg.payload` property containing the body of the message.

3.4.3 Debug node

Outputs the content of a msg object to the debug tab.

3.4.4 mqtt in node

MQTT input node. Connects to a broker and subscribes to the specified topic. The topic may contain MQTT wildcards.

Outputs an object called msg containing

`msg.topic, msg.payload, msg.qos, msg.retain.`

msg.payload is usually a string, but can also be a binary buffer.

3.4.5 mqtt out node

Connects to a MQTT broker and publishes msg.payload either to the msg.topic or to the topic specified in the edit window. The value in the edit window has precedence.

Likewise QoS and/or retain values in the edit panel will overwrite any msg.qos and msg.retain properties. If nothing is set they default to 0 and false respectively.

3.4.6 Watson Language translator node

The Language Translator service enables you to translate text from one language to another.

3.4.7 Cloudant nodes

Allows basic access to a Cloudant database to `insert`, `update`, `delete` and `search` for documents.

4 MQTT

The protocol is intended for use on wireless and low-bandwidth networks. A mobile application that uses MQTT sends and receives messages by calling an MQTT library. The messages are exchanged through an MQTT messaging server. The MQTT client and server handle the complexities of delivering messages reliably for the mobile app and keep the cost of network management small.

4.1 MQTT protocol

The MQTT protocol is lightweight in the sense that clients are small, and it uses network bandwidth efficiently. The MQTT protocol supports assured delivery and fire-and-forget transfers. In the protocol, message delivery is decoupled from the application. The extent of decoupling in an application depends on the way an MQTT client and MQTT server are written. Decoupled delivery frees up an application from any server connection, and from waiting for messages.

The MQTT V3.1 protocol is published; see MQTT V3.1 Protocol Specification. The specification identifies a number of distinctive features about the protocol:

- ➢ It is a publish/subscribe protocol.
  - In addition to providing one-to-many message distribution, publish/subscribe decouples applications. Both features are useful in applications that have many clients.
- ➢ It is not dependent in any way on the message content.
- ➢ It runs over TCP/IP, which provides basic network connectivity.

4.2 Quality of service

4.2.1 At most once

Messages are delivered according to the best efforts of the underlying Internet Protocol network. Message loss might occur. Use this quality of service with communicating ambient sensor data, for example. It does not matter if an individual reading is lost, if the next one is published soon after.

4.2.2 At least once

Messages are assured to arrive but duplicates might occur.

4.2.3 Exactly once

Messages are assured to arrive exactly once.

Use this quality of service with billing systems, for example. Duplicate or lost messages might lead to inconvenience or imposing incorrect charges.

4.3 Client

When talking about a client it almost always means an MQTT client. This includes publisher or subscribers, both of them label an MQTT client that is only doing publishing or subscribing. (In general a MQTT client can be both a publisher & subscriber at the same time). **A MQTT client is any device from a micro controller up to a full fledged server, that has a MQTT library running and is connecting to an MQTT broker over any kind of network.** This could be a really small and resource constrained device, that is connected over a wireless network and has a library strapped to the minimum or a typical computer running a graphical MQTT client for testing purposes, basically any device that has a TCP/IP stack and speaks MQTT over it. The client implementation of the MQTT protocol is very straight-forward and really reduced to the essence. That's one aspect, why MQTT is ideally suitable for small devices.

4.4 Broker

The counterpart to a MQTT client is the MQTT broker, which is the heart of any publish/subscribe protocol. Depending on the concrete implementation, a broker can handle up to thousands of concurrently connected MQTT clients. **The broker is primarily responsible for receiving all messages, filtering them, decide who is interested in it and then sending the message to all subscribed clients.** It also holds the session of all persisted clients including subscriptions and missed messages. Another responsibility of the broker is the authentication and authorization of clients. And at most of the times a broker is also extensible, which allows to easily integrate custom authentication, authorization and integration into backend systems. Especially the integration is an important aspect, because often the broker is the component, which is directly exposed on the internet and handles a lot of clients and then passes messages along to downstream analyzing and processing systems.

4.5 MQTT Connection

The MQTT protocol is based on top of TCP/IP and both client and broker need to have a TCP/IP stack. The MQTT connection itself is always between one client and the broker, no client is connected to another client directly. **The connection is initiated through a client sending a CONNECT message to the broker. The broker response with a CONNACK** and a status code. Once the connection is established, the broker will keep it open as long as the client doesn't send a disconnect command or it looses the connection.

5 Lab objectives

In part 1 you will sign up to Bluemix and setup a blank Node-RED flow.

In part 2 you will send messages to members of your group using mqtt nodes and topics.

In part 3 you will create a new flow and explore Cloudant and Watson Analytics nodes.

5.1 Learning outcomes

On successful completion of this lab you will have learned about

- MQTT pub/sub protocol and QoS

- Controlling data through MQTT topics using Node-RED flows

- How messages flow through nodes

- Debug and mqtt nodes

- Cloudant database nodes

- Watson Analytics nodes

5.2 Topic structure

→ ***iot-summer-school***

→ **group-1-iotp**

→ **Ann**

→ **Bob**

→ **Joe**

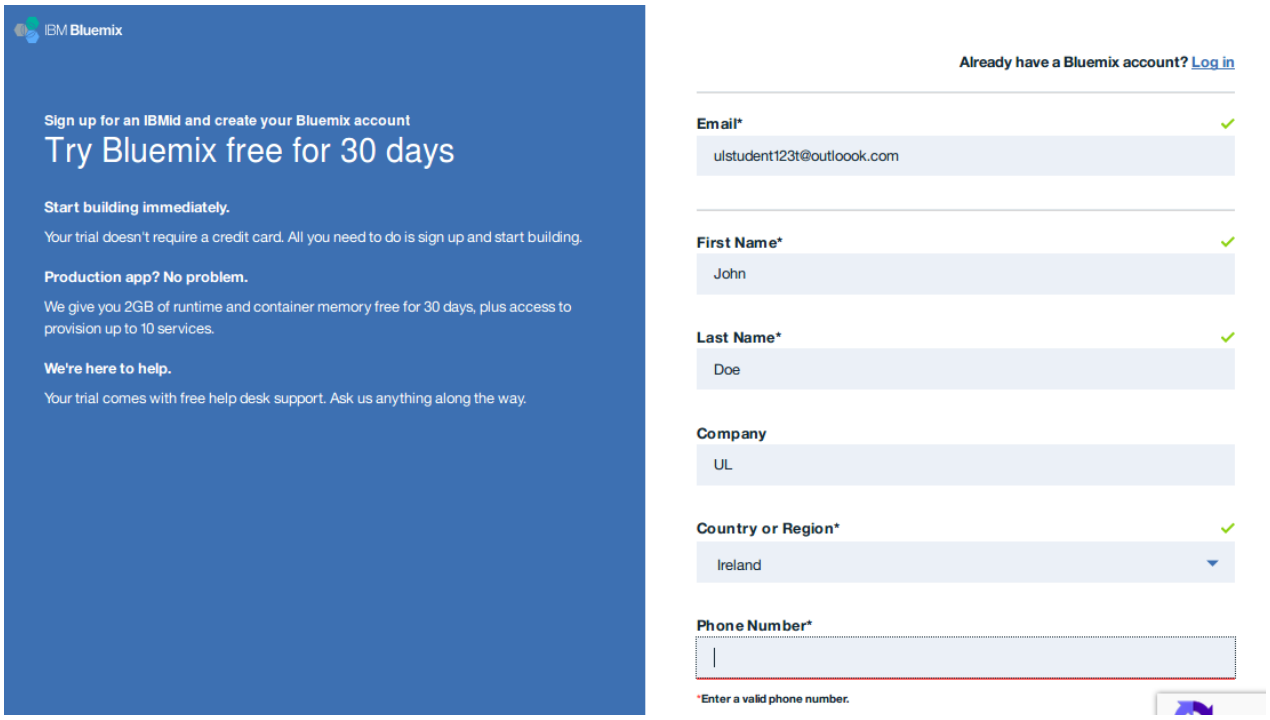
→ **group-2-iotp**

→ **Jim**

→ **…**

6 Lab part 1: Getting started

Step 1 : Sign up for 30 day free trial



IBM Bluemix

Sign up for an IBMid and create your Bluemix account

Try Bluemix free for 30 days

Start building immediately.

Your trial doesn't require a credit card. All you need to do is sign up and start building.

Production app? No problem.

We give you 2GB of runtime and container memory free for 30 days, plus access to provision up to 10 services.

We're here to help.

Your trial comes with free help desk support. Ask us anything along the way.

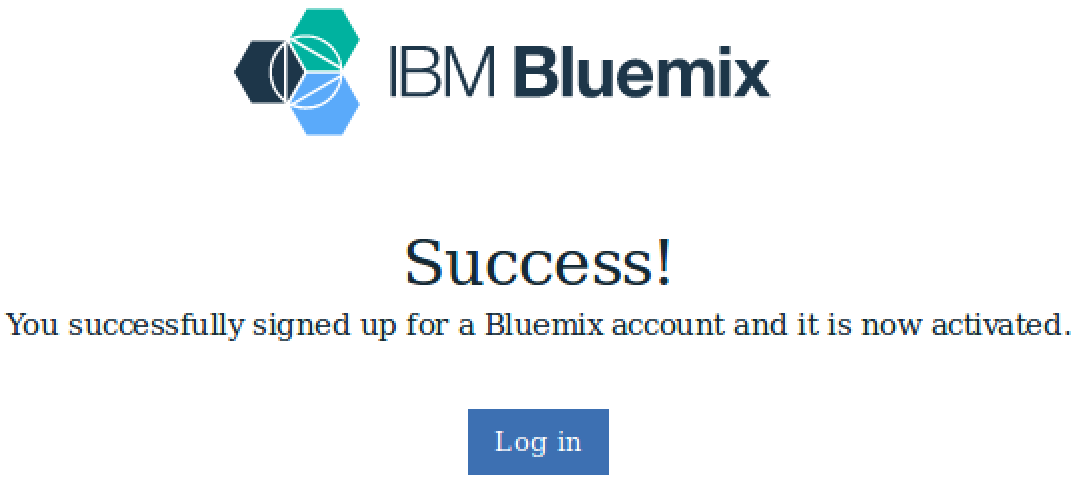
Already have a Bluemix account? Log in

Email*	ulstudent123t@outloook.com
First Name*	John
Last Name*	Doe
Company	UL
Country or Region*	Ireland
Phone Number*	

*Enter a valid phone number.

Step 2: Click on link in the email to complete registration

Click on "Log in"

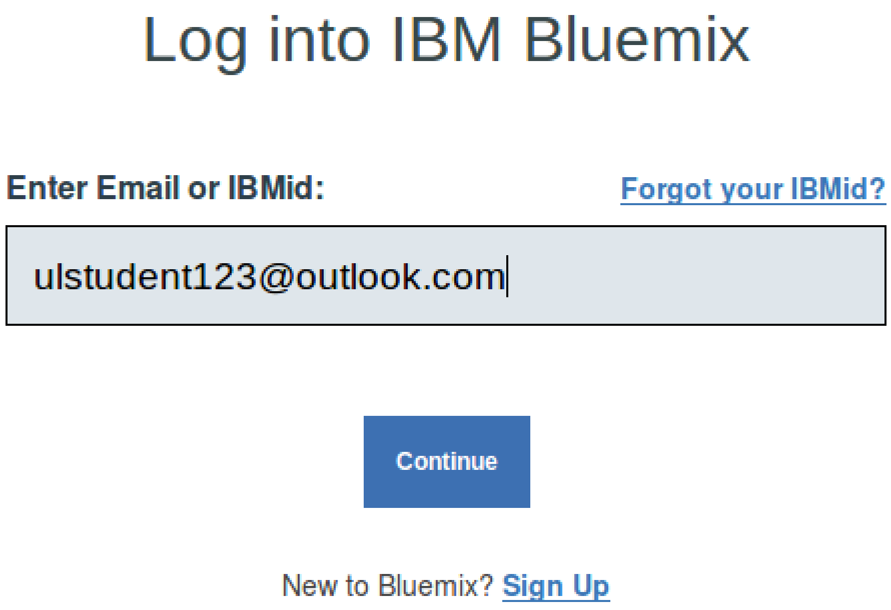


IBM Bluemix

Success!

You successfully signed up for a Bluemix account and it is now activated.

Step 3: Login



Log into IBM Bluemix

Enter Email or IBMid:

ulstudent123@outlook.com

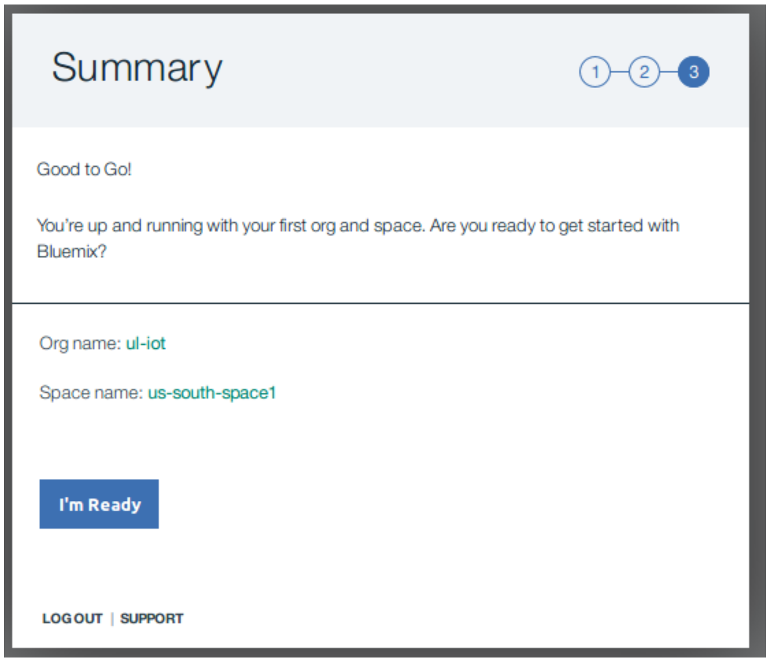
Step 4: Create organization

Create organization in the <u>United Kingdom</u> region. If stuck for a name, use “uk-org1”

Step 5: Create space

If stuck for a name, use “uk-space1”

Step 6: Click “I’m Ready”



Summary

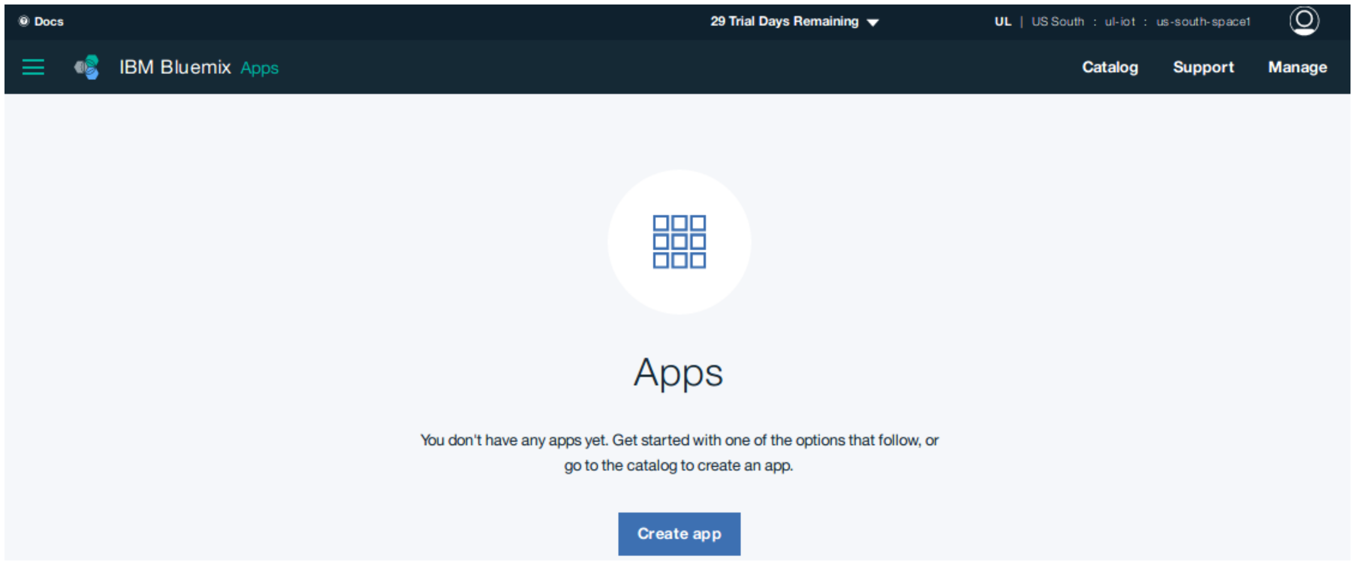
Good to Go!

You're up and running with your first org and space. Are you ready to get started with Bluemix?

Org name: ul-iot

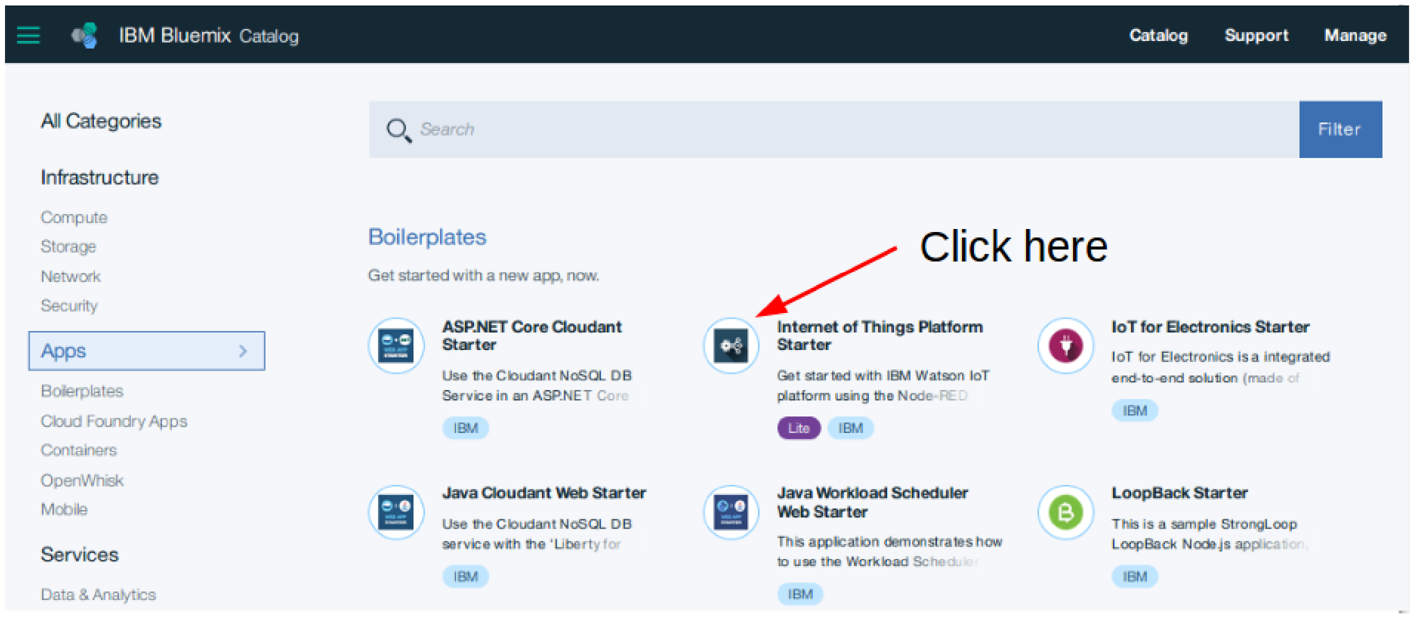
Space name: us-south-space1

Step 7: Click on “Create app”

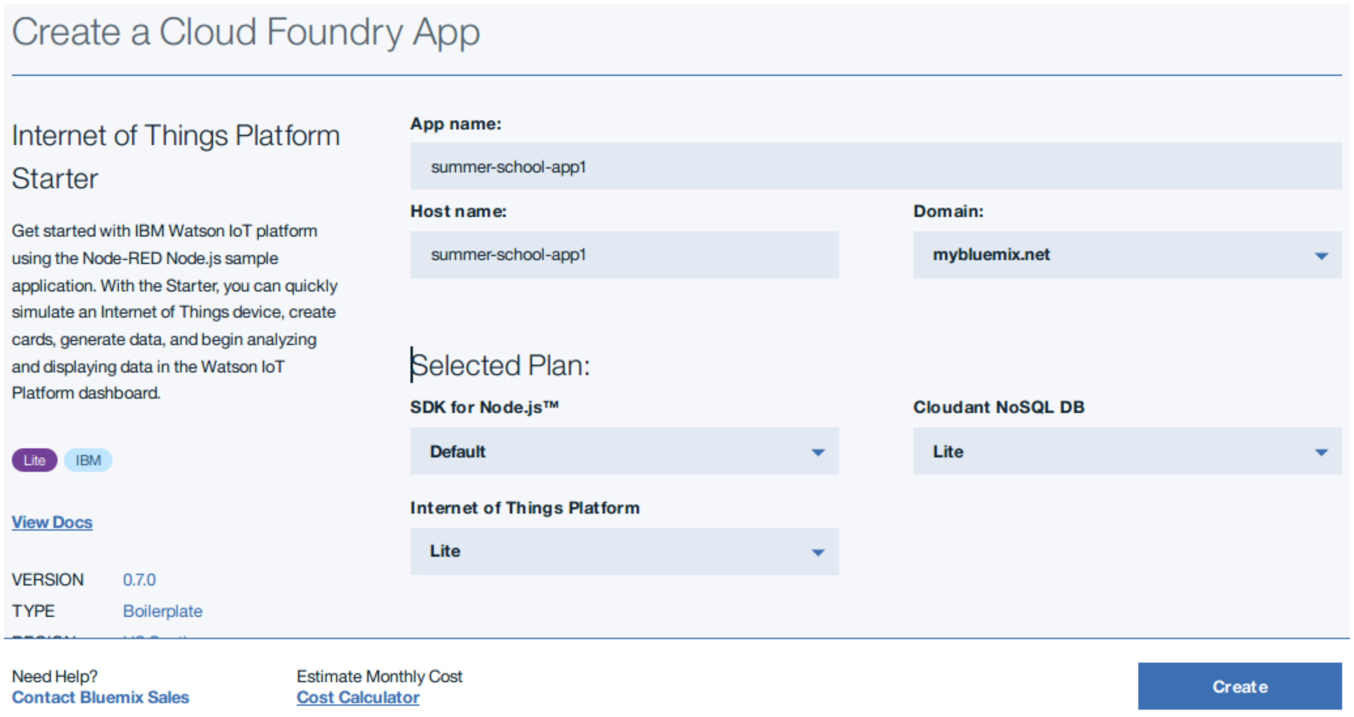


You don't have any apps yet. Get started with one of the options that follow, or go to the catalog to create an app.

Step 8: Select “Internet of Things Platform Starter”


IBM Bluemix Catalog
Catalog
Support
Manage
All Categories
Infrastructure
Compute
Storage
Network
Security
Apps
Boilerplates
Cloud Foundry Apps
Containers
OpenWhisk
Mobile
Services
Data & Analytics
Search
Filter
Boilerplates
Get started with a new app, now.
Click here
ASP.NET Core Cloudant Starter
Use the Cloudant NoSQL DB Service in an ASP.NET Core
IBM
Internet of Things Platform Starter
Get started with IBM Watson IoT platform using the Node-RED
Lite
IBM
IoT for Electronics Starter
IoT for Electronics is a integrated end-to-end solution (made of
IBM
Java Cloudant Web Starter
Use the Cloudant NoSQL DB service with the 'Liberty for
IBM
Java Workload Scheduler Web Starter
This application demonstrates how to use the Workload Scheduler
IBM
LoopBack Starter
This is a sample StrongLoop LoopBack Node.js application
IBM

Step 9: Create a Cloud Foundry App


Create a Cloud Foundry App

Internet of Things Platform Starter

Get started with IBM Watson IoT platform using the Node-RED Node.js sample application. With the Starter, you can quickly simulate an Internet of Things device, create cards, generate data, and begin analyzing and displaying data in the Watson IoT Platform dashboard.

VERSION	0.7.0
TYPE	Boilerplate

App name: summer-school-app1

Host name: summer-school-app1

Domain: mybluemix.net

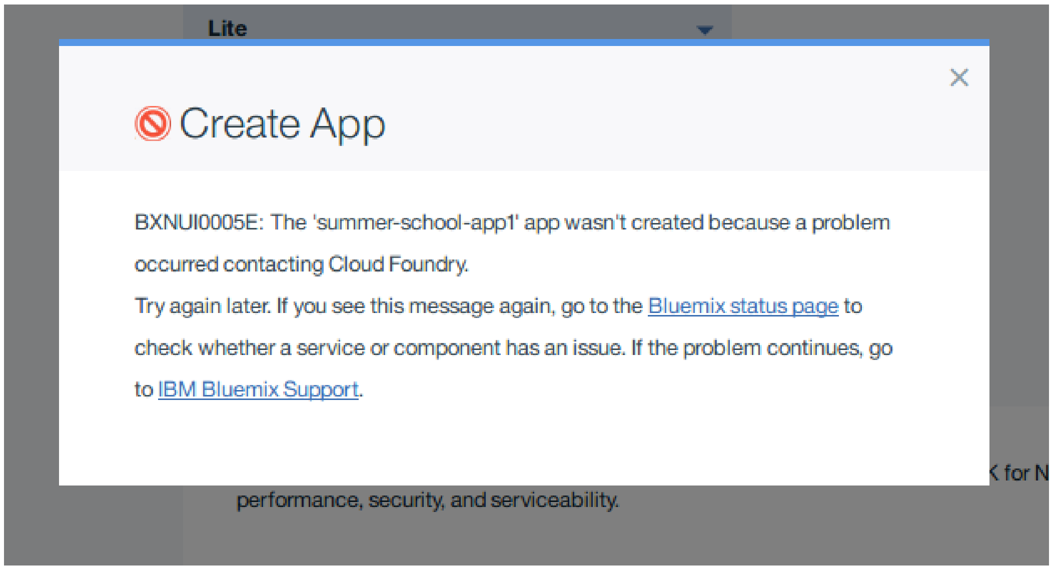
Selected Plan:

SDK for Node.js™: Default

Cloudant NoSQL DB: Lite

Internet of Things Platform: Lite

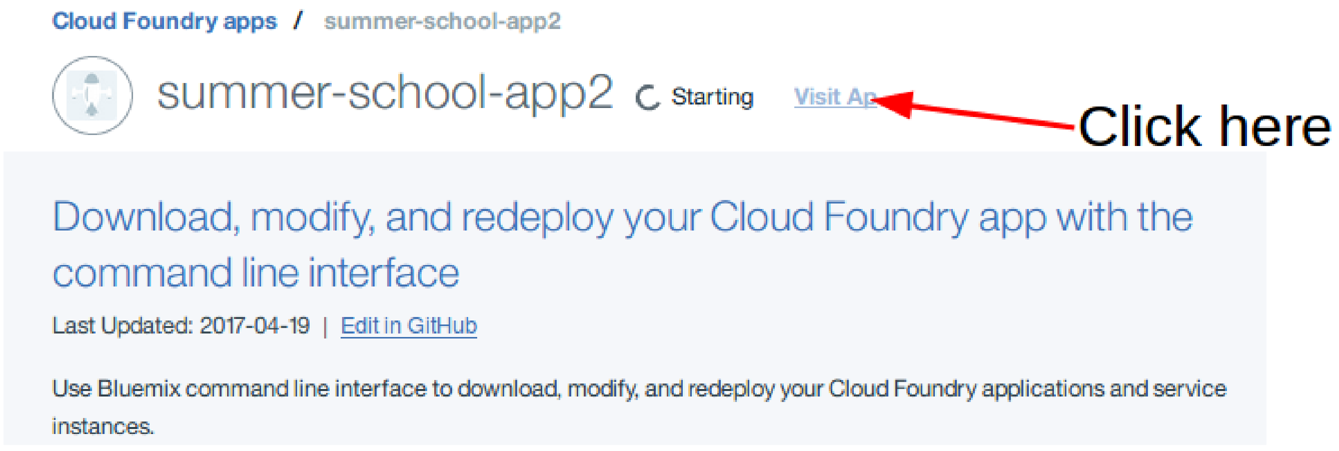
I am getting this error when I click Create?


Create App

BXNUI0005E: The 'summer-school-app1' app wasn't created because a problem occurred contacting Cloud Foundry.
Try again later. If you see this message again, go to the Bluemix status page to check whether a service or component has an issue. If the problem continues, go to IBM Bluemix Support.

Try renaming the app and then restart the application. If the problem continues, please call for assistance.

Step 10: Wait for app to start running



Cloud Foundry apps / summer-school-app2

summer-school-app2 Starting Visit Ap

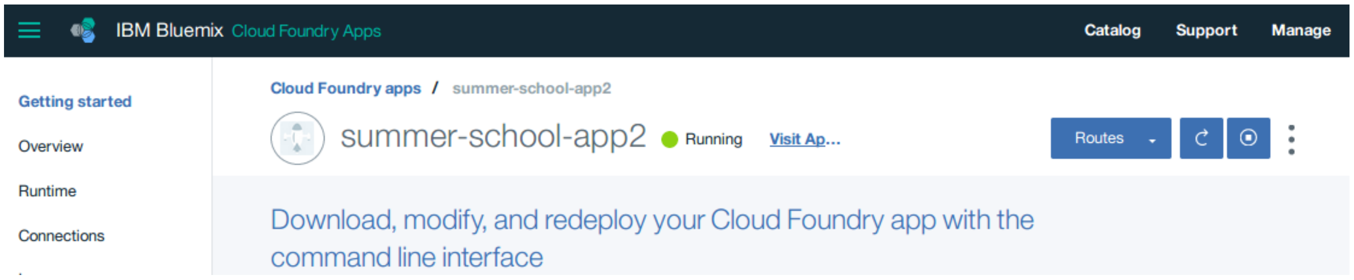
Click here

Download, modify, and redeploy your Cloud Foundry app with the command line interface

Last Updated: 2017-04-19 | Edit in GitHub

Use Bluemix command line interface to download, modify, and redeploy your Cloud Foundry applications and service instances.

Step 11: Click on the URL once the app is running

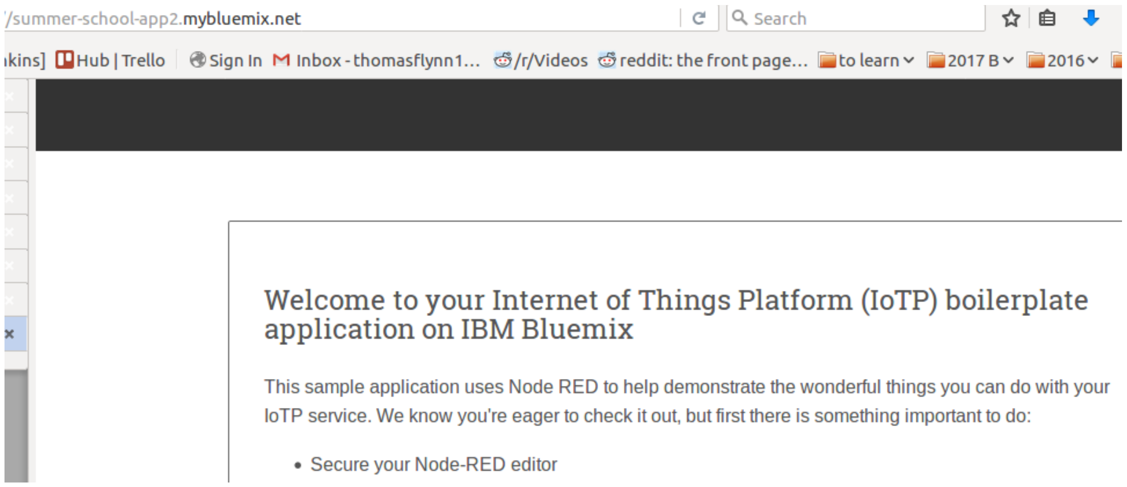


Cloud Foundry apps / summer-school-app2

summer-school-app2 Running Visit Ap...

Download, modify, and redeploy your Cloud Foundry app with the command line interface

Step 12: Node-RED welcome message



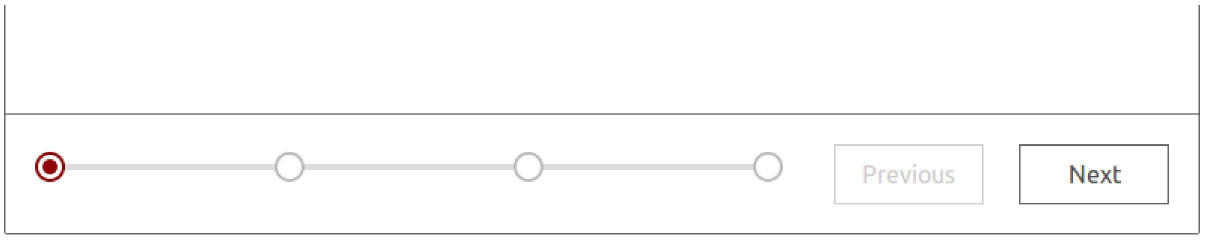
/summer-school-app2.mybluemix.net
Search
kins] Hub | Trello Sign In Inbox - thomasflynn1... /r/Videos reddit: the front page... to learn 2017 B 2016

Welcome to your Internet of Things Platform (IoTP) boilerplate application on IBM Bluemix

This sample application uses Node RED to help demonstrate the wonderful things you can do with your IoTP service. We know you're eager to check it out, but first there is something important to do:

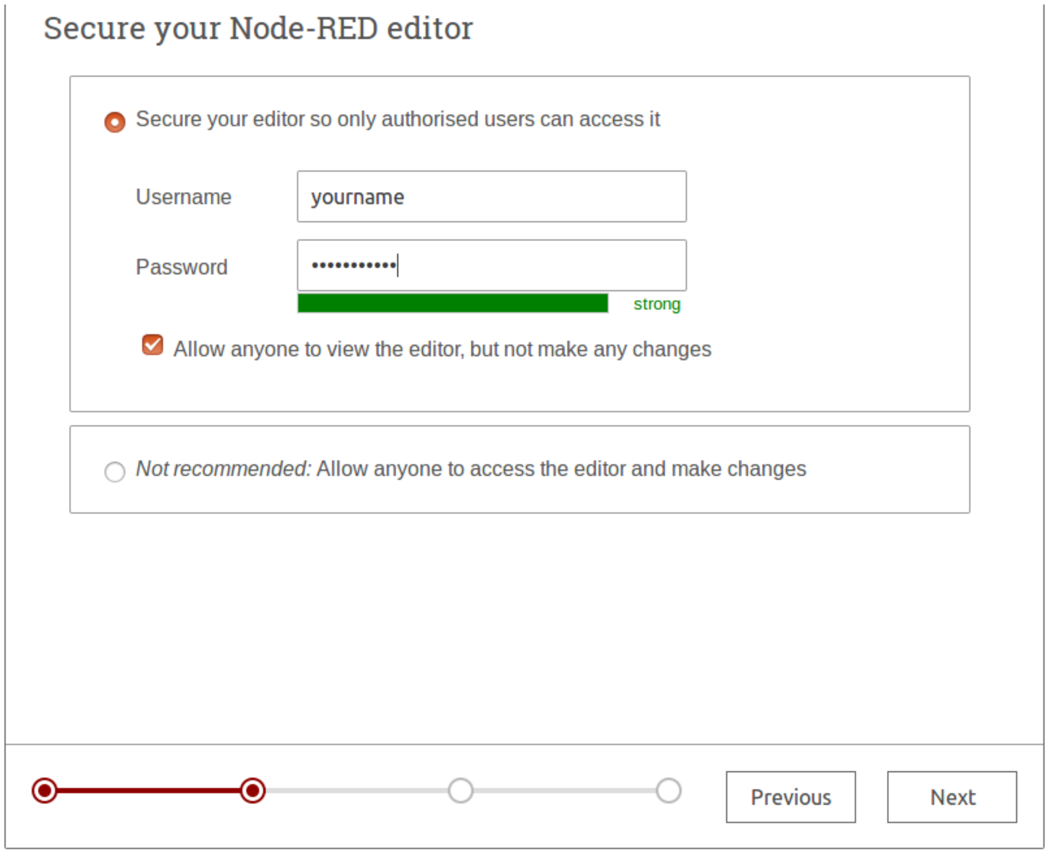
- Secure your Node-RED editor

Step 13: Click next



Step 14: Secure your Node-RED editor

Please remember your username and password as these credentials are different to your Bluemix credentials.



Secure your Node-RED editor

● Secure your editor so only authorised users can access it

Username: yourname

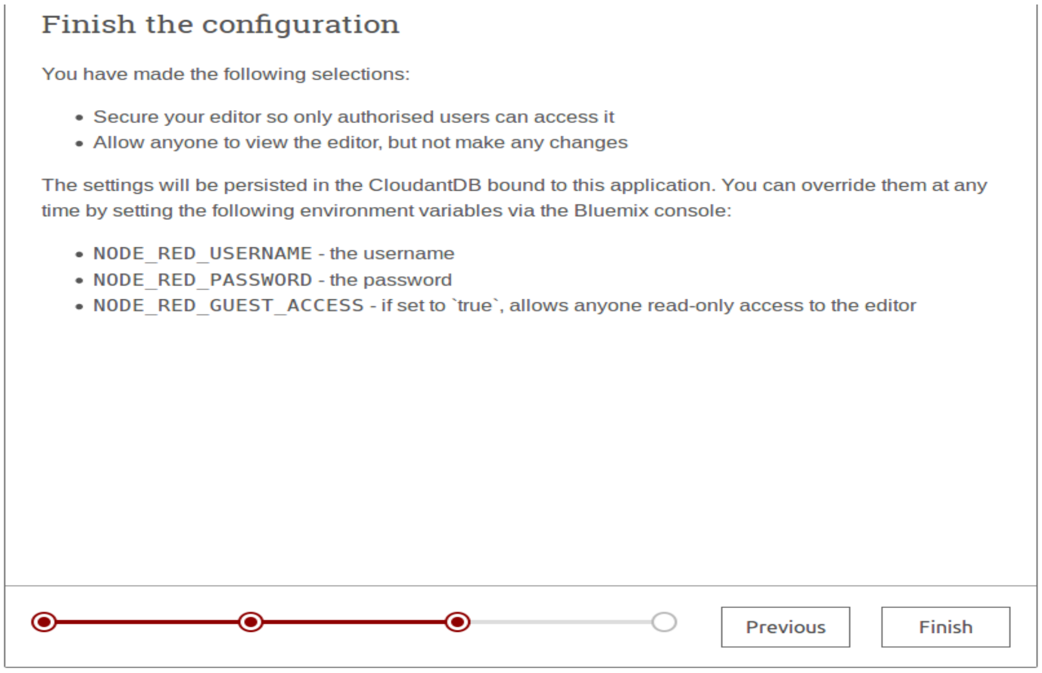
Password: •••••••••••

strong

✓ Allow anyone to view the editor, but not make any changes

○ *Not recommended:* Allow anyone to access the editor and make changes

Step 15: Finish the configuration



Finish the configuration

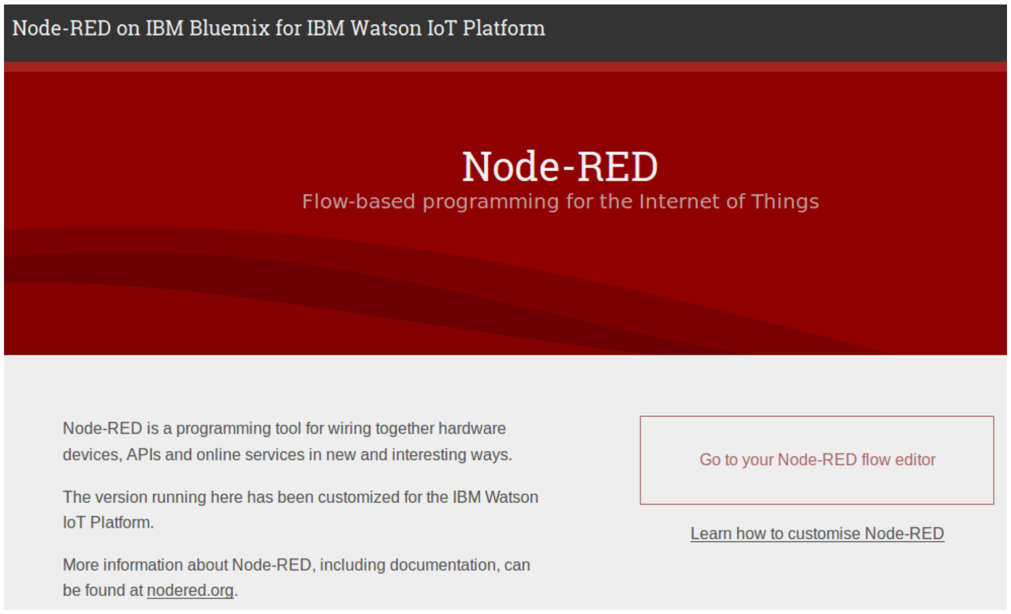
You have made the following selections:

- Secure your editor so only authorised users can access it
- Allow anyone to view the editor, but not make any changes

The settings will be persisted in the CloudantDB bound to this application. You can override them at any time by setting the following environment variables via the Bluemix console:

- `NODE_RED_USERNAME` - the username
- `NODE_RED_PASSWORD` - the password
- `NODE_RED_GUEST_ACCESS` - if set to `true`, allows anyone read-only access to the editor

Step 16: Click on “Go to your Node-RED flow editor”


Node-RED on IBM Bluemix for IBM Watson IoT Platform

Node-RED
Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

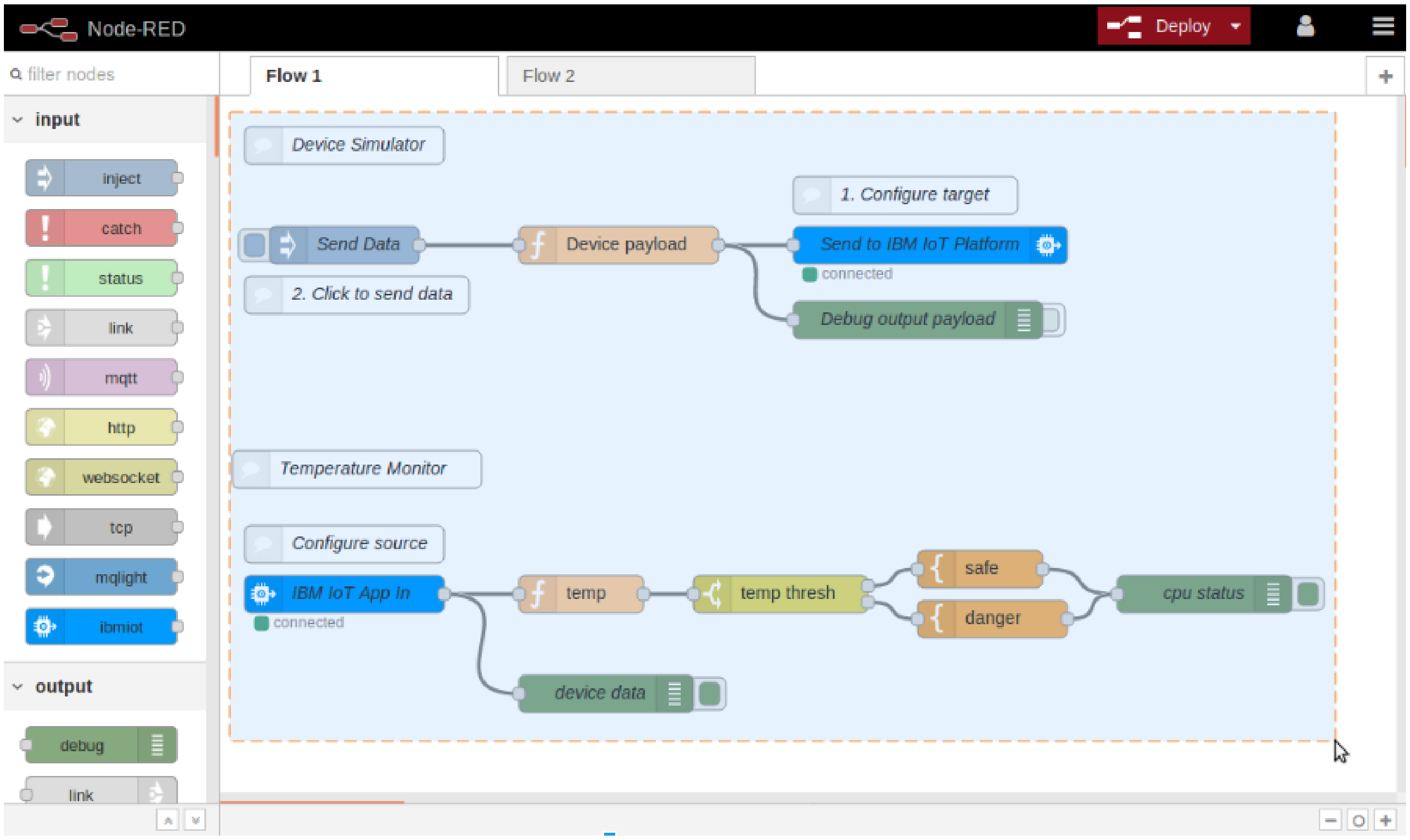
The version running here has been customized for the IBM Watson IoT Platform.

More information about Node-RED, including documentation, can be found at nodered.org.

Go to your Node-RED flow editor
Learn how to customise Node-RED

Step 17: Delete nodes in Flow 1

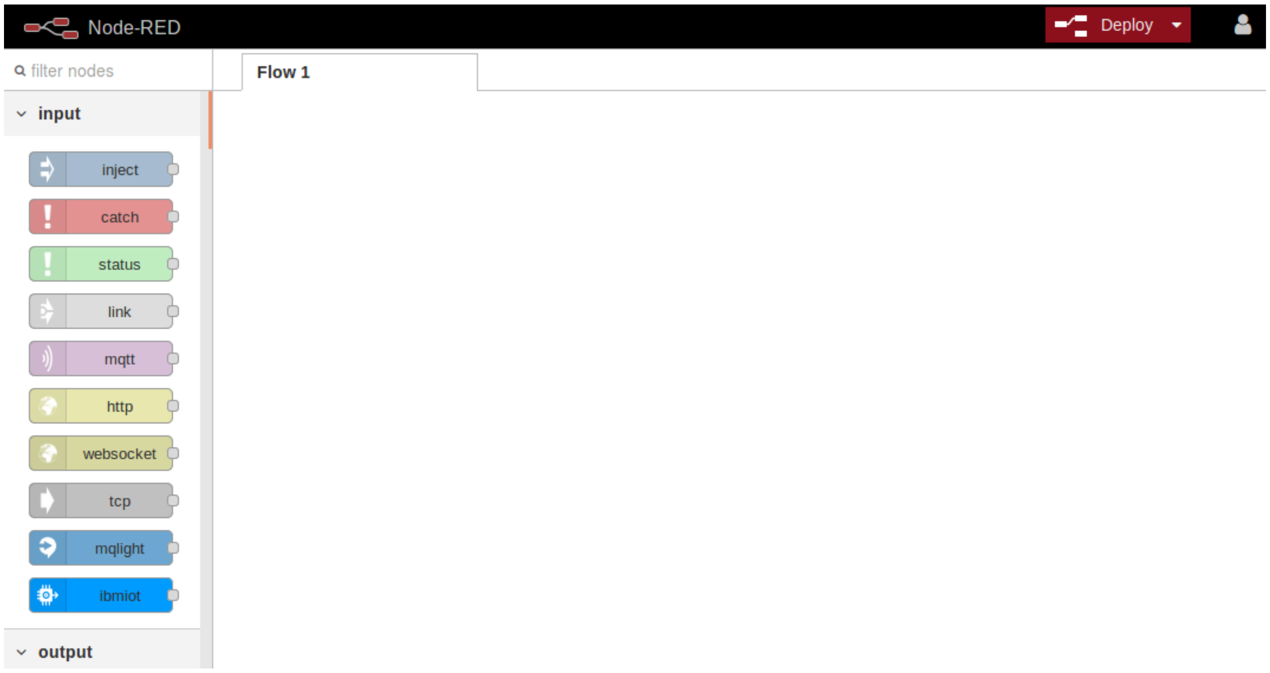
To delete nodes, drag the mouse over all the nodes in the flow. Then hit the delete key on the keyboard to delete them.



Step 18: Finished setup phase

Once you have reached this step raise your hand and inform the teaching assistant.

If the teaching assistant is occupied, continue onto part 2 of the lab.



7 Lab part 2: Your first flow

Translated words:

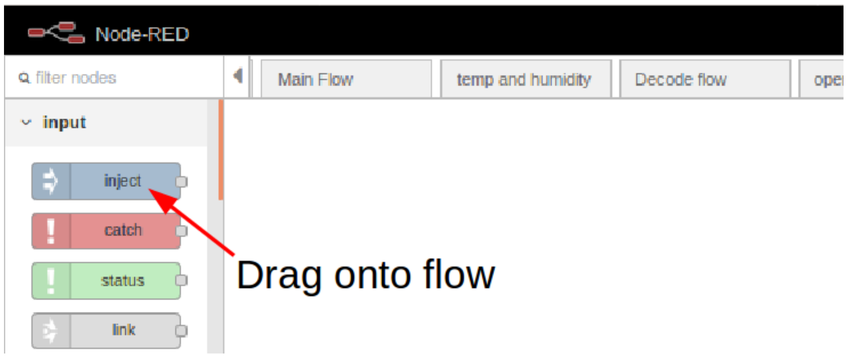
input / output	输入 / 输出
Inject	注入
insert your name	插入你的名字
function	功能
payload	有效载荷
publish	发布
subscribe	订阅
broker	代理
topic	主题
send/sent	发送
receive/received	接收
debug	调试
insert group number	插入组号
deploy	部署
complete	完成
format	格式
message	信息
quality of service (qos)	服务质量

retain → 保留

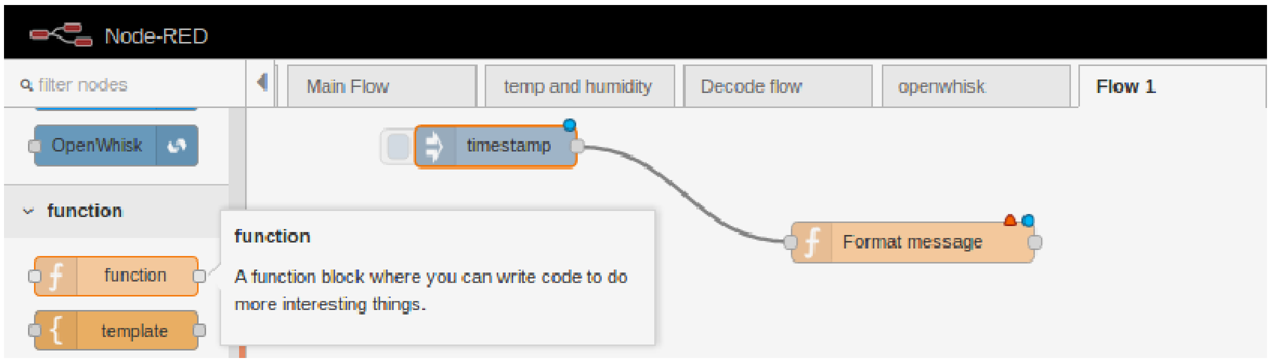
notice → 注意

your name → 你的名字

Step 1: Add inject node to Flow 1



Step 2: Add function node to Flow 1

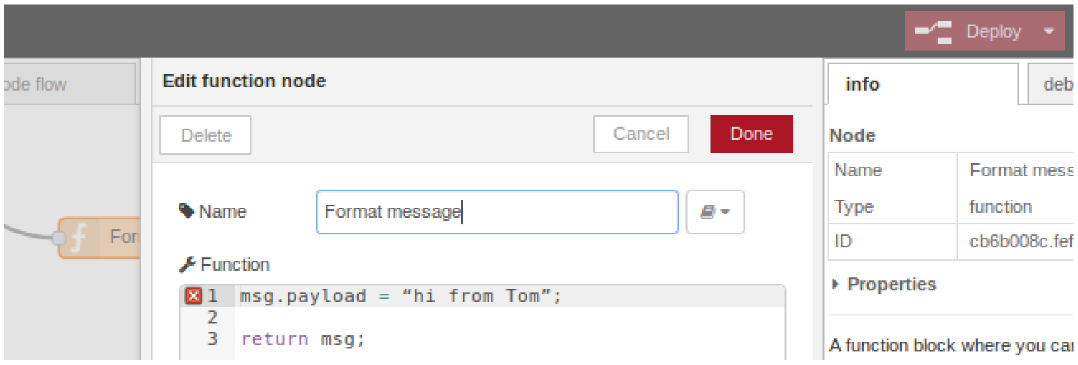


function

A function block where you can write code to do more interesting things.

Connect the output of the inject node to the input of the function node as shown in the above figure.

Messages can be modified within the node by double clicking on the function node.



Name: Format message

Function

```
msg.payload = "hi from Tom";

return msg;
```

Node

Name	Format mess
Type	function
ID	cb6b008c.fef

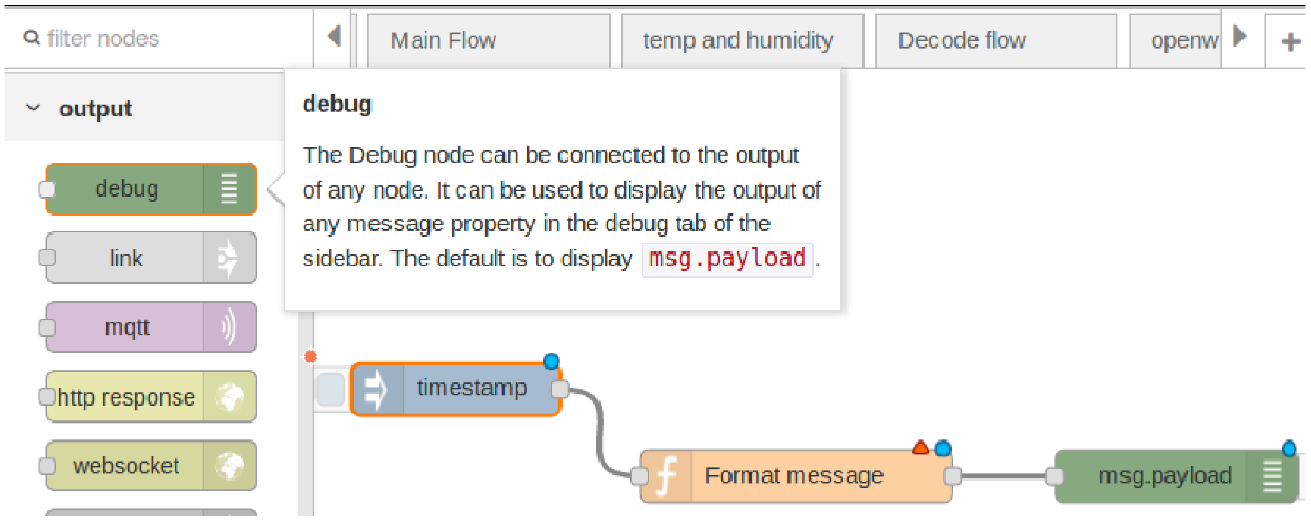
A function block where you ca

Step 3: Add line of code

Add the following line but do not copy and paste.

msg.payload = “hi from <insert your name>”;

Step 4: Add debug node to Flow 1



debug

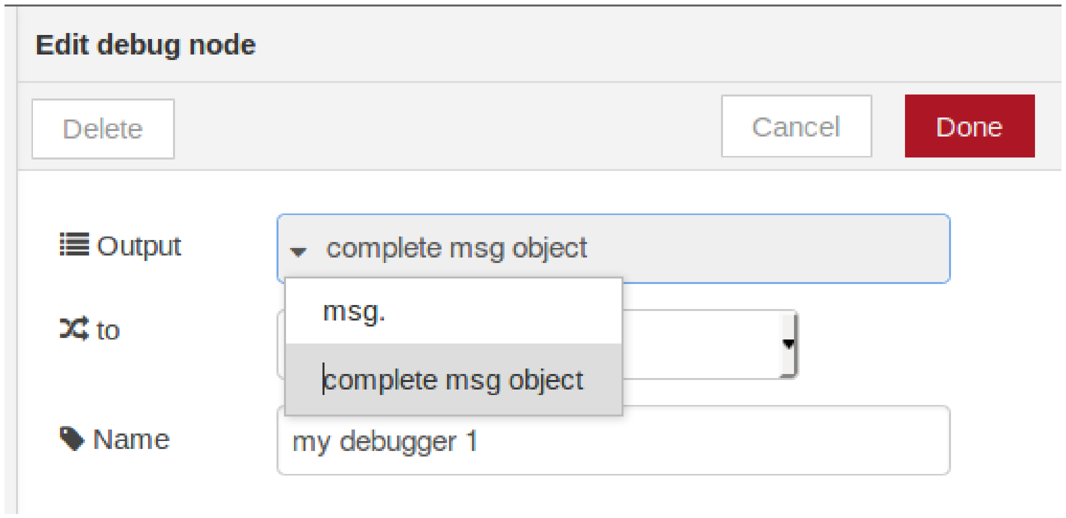
The Debug node can be connected to the output of any node. It can be used to display the output of any message property in the debug tab of the sidebar. The default is to display `msg.payload`.

Step 5: Edit debug node

Change the output option to “complete msg object”.

Why do we need to see the complete message?

To help us understand the structure and attributes of messages as they pass through nodes.



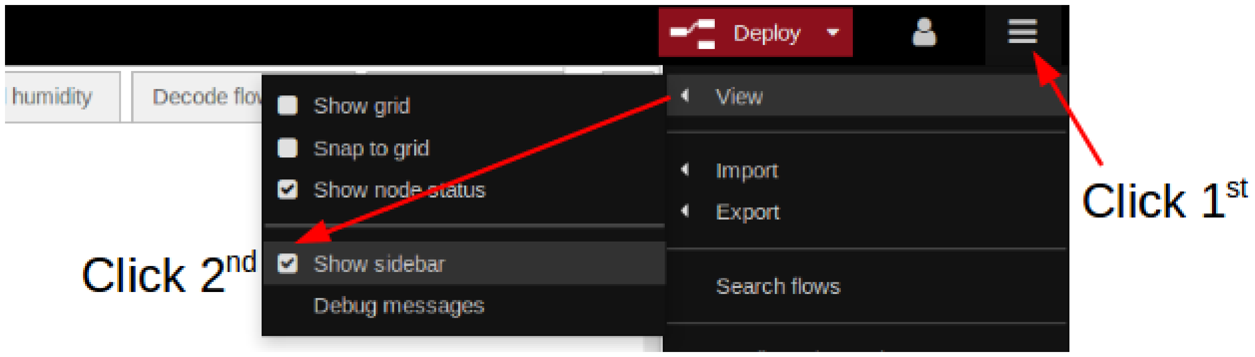
Edit debug node

Output: complete msg object

to

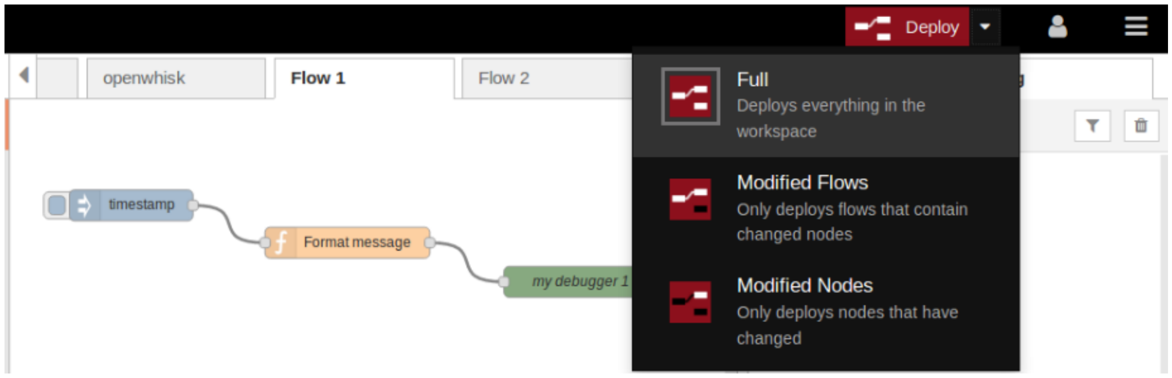
Name: my debugger 1

I cannot see my debug tab?

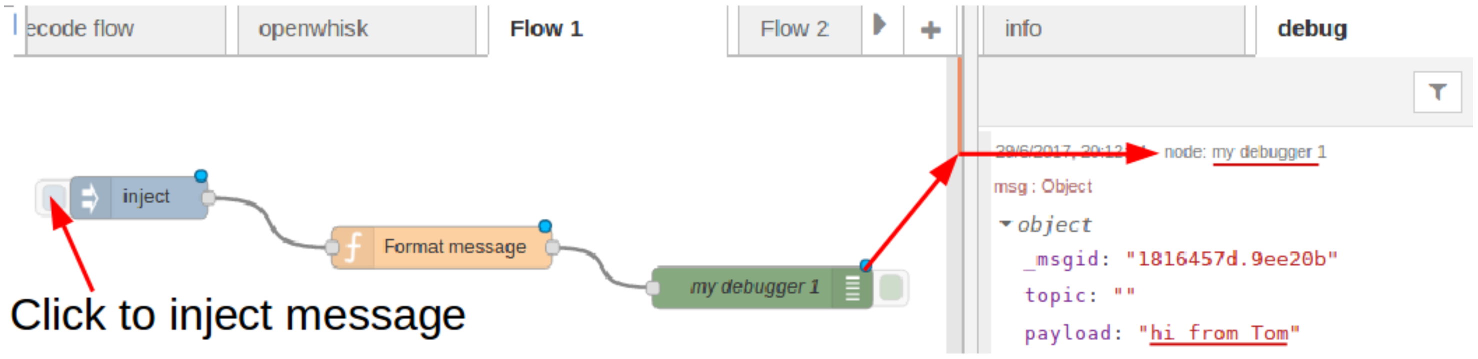


Step 6: Deploy Flow 1

After clicking on the drop-down arrow, click “Full”. Then click on the deploy button to deploy the flow.


Deploy
openwhisk
Flow 1
Flow 2
Full
Deploys everything in the workspace
Modified Flows
Only deploys flows that contain changed nodes
Modified Nodes
Only deploys nodes that have changed
timestamp
Format message
my debugger 1

Step 7: Test debug node



```
msg : Object
▾object
  _msgid: "1816457d.9ee20b"
  topic: ""
  payload: "hi from Tom"
```

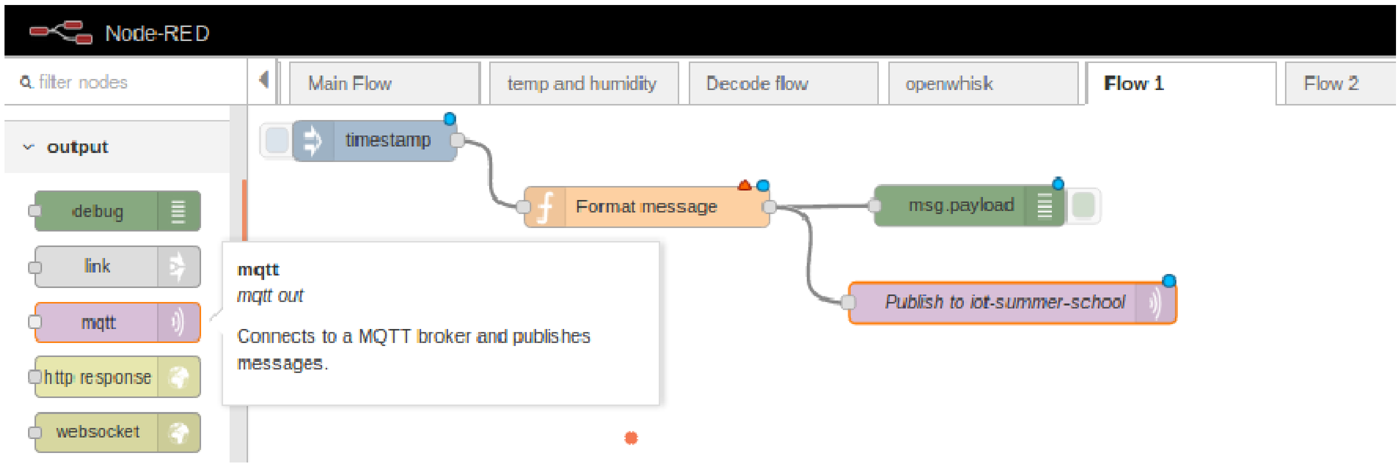
In the debug tab on the right, notice how each new message has a uniquely generated **msg._msgid**

Notice that **msg.topic** is blank.

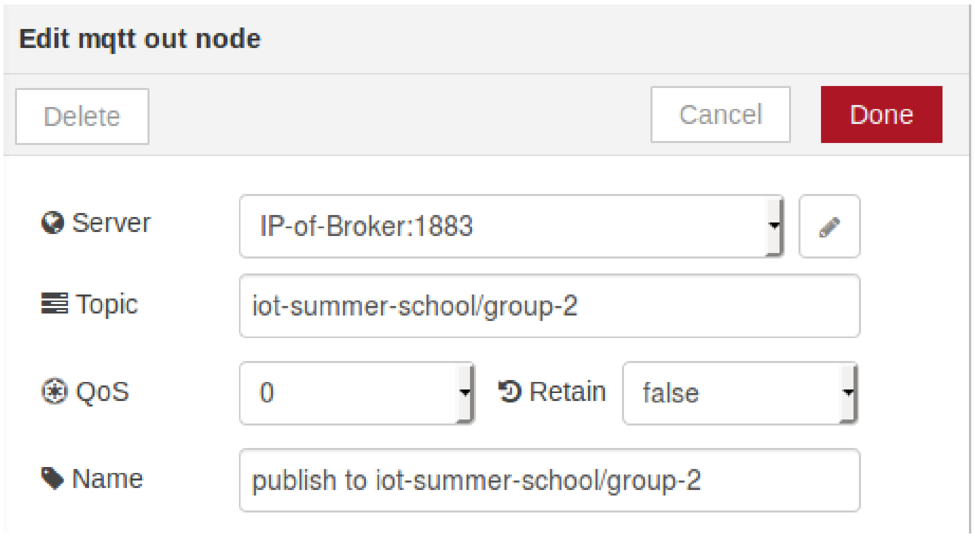
Notice the value of **msg.payload** is the value applied in the function node (“hi from <your name>”)

Step 8: Publish to the *“iot-summer-school/group-<insert group number>”* topic

Add the mqtt out node and wire it to the output of the function node



Double click on the mqtt output node and make the following changes.


Edit mqtt out node

Delete | Cancel | Done

Server: IP-of-Broker:1883

Topic: iot-summer-school/group-2

QoS: 0 | Retain: false

Name: publish to iot-summer-school/group-2

What is the mqtt <u>out</u> node going to do with the message?

The node will publish to the broker on "iot-summer-school/group-2" topic.

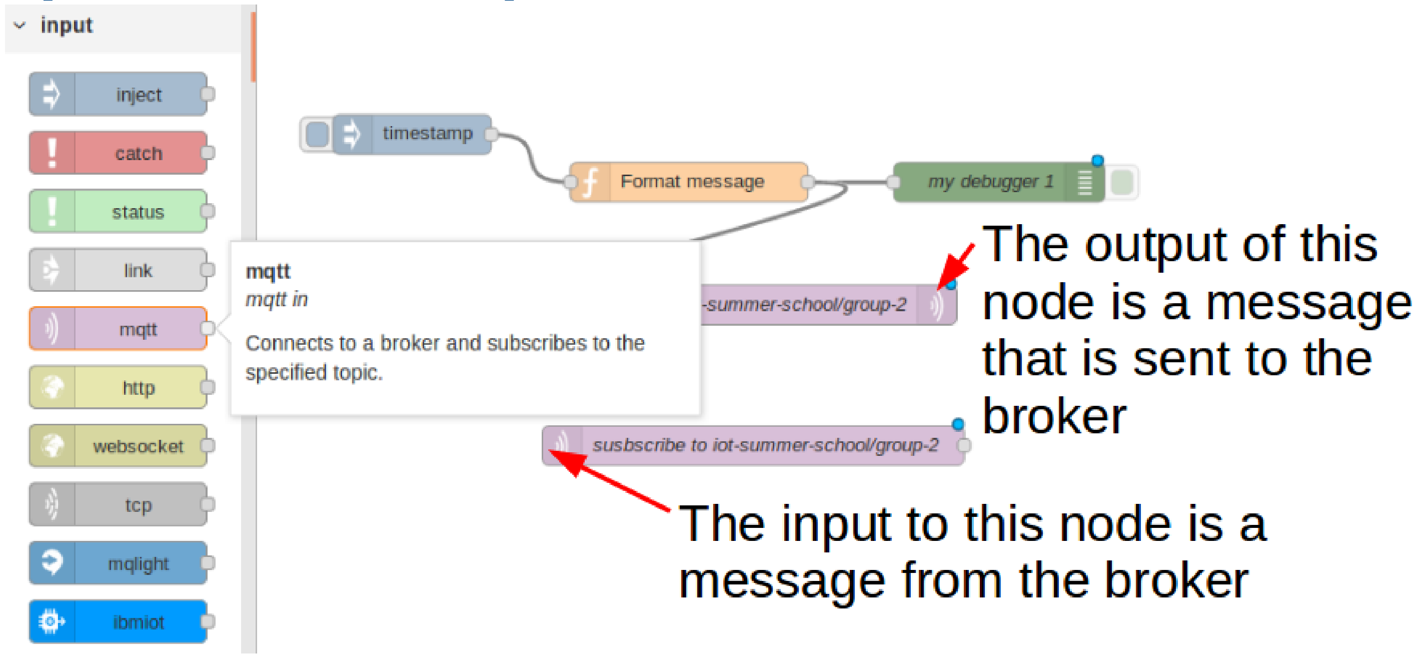
Any mqtt <u>in</u> node that is subscribed to that topic will receive a message with a payload containing

"hi from <insert your name>"

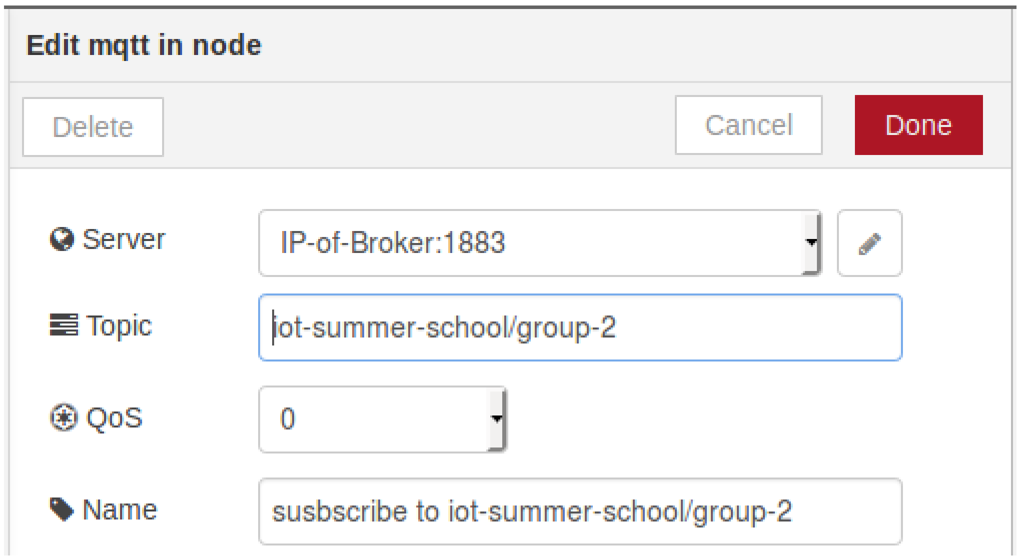
Will my flow also receive the message it just published?

No, we have only used an mqtt <u>out</u> node. This node only <u>publishes</u> messages and does <u>not</u> <u>subscribe</u>.

Step 9: Add subscribe node or "mqtt in" node


input
inject
catch
status
link
mqtt
http
websocket
tcp
mqlight
ibmiot
timestamp
Format message
my debugger 1
-summer-school/group-2
susbscribe to iot-summer-school/group-2
mqtt
mqtt in
Connects to a broker and subscribes to the specified topic.
The output of this node is a message that is sent to the broker
The input to this node is a message from the broker

Step 10: Edit the mqtt in node

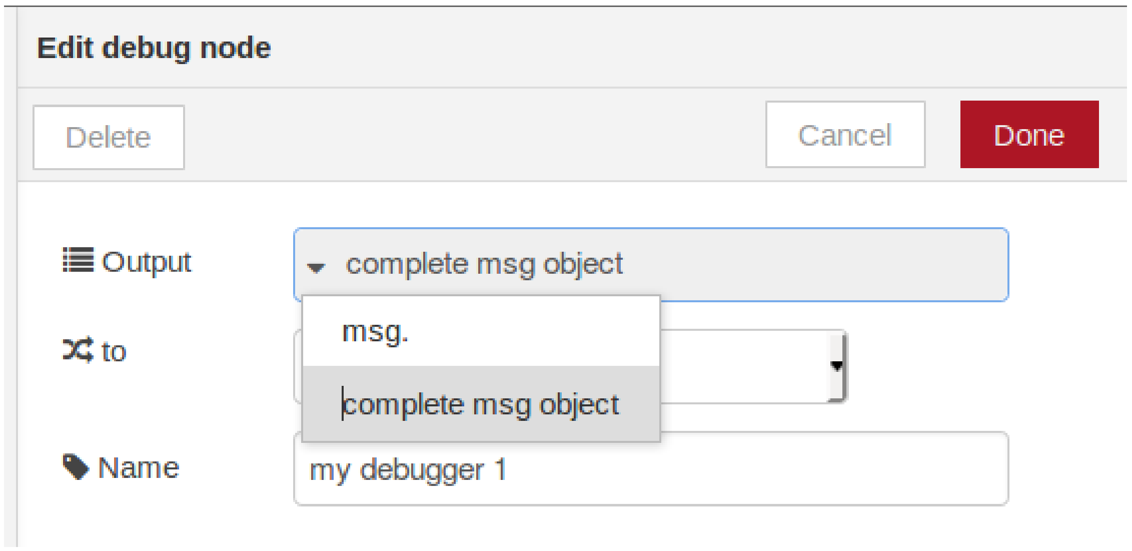


Server: IP-of-Broker:1883
Topic: iot-summer-school/group-2
QoS: 0
Name: susbscribe to iot-summer-school/group-2

Step 11: Add another debug node

Make the same change as last time, we do this because we want to see all message attributes.

Then wire the output of the mqqt in node to the input of the new debug node.

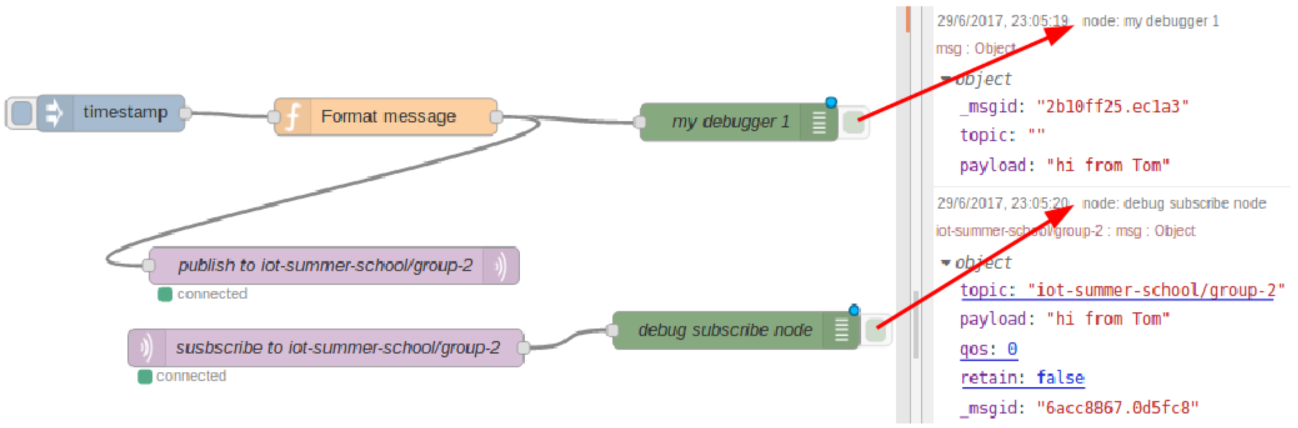


Output: complete msg object
to
Name: my debugger 1

Step 12: Deploy the application

Remember to hit the deploy button after making a change to a flow.

Step 13: Check the debug tab


timestamp
Format message
my debugger 1
publish to iot-summer-school/group-2
connected
susbscribe to iot-summer-school/group-2
connected
debug subscribe node

```
29/6/2017, 23:05:19  node: my debugger 1
msg : Object
▾object
  _msgid: "2b10ff25.ec1a3"
  topic: ""
  payload: "hi from Tom"
29/6/2017, 23:05:20  node: debug subscribe node
iot-summer-school/group-2 : msg : Object
▾object
  topic: "iot-summer-school/group-2"
  payload: "hi from Tom"
  qos: 0
  retain: false
  _msgid: "6acc8867.0d5fc8"
```

Notice the message captured in the “debug subscribe node” has the following attributes

topic: “iot-summer-school/group”

qos: 0

retain: false

And most importantly the payload contains “hi from <your name>”

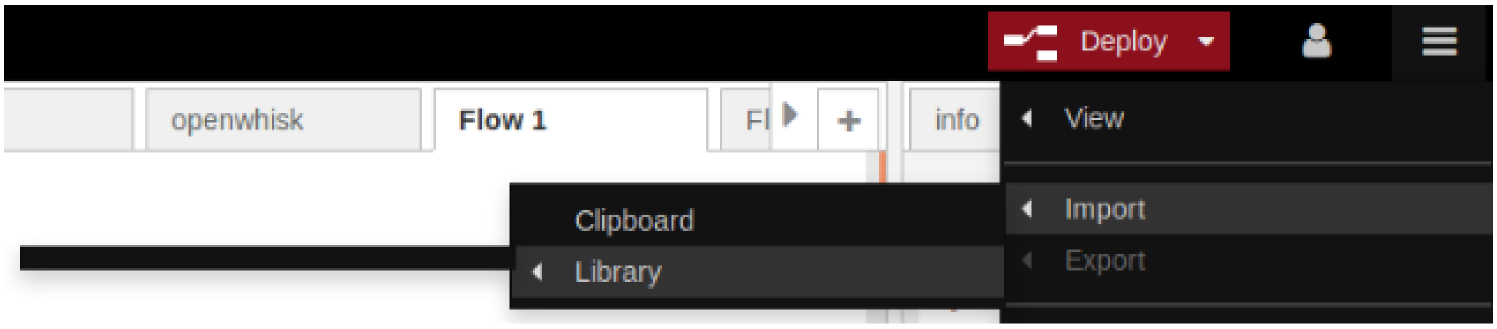
Step 14: Help your group members

After you get the subscribe node working, go help group members that are having problems.

When everyone in the group can send and receive messages, continue to the next stage and create a new flow called “Flow 2”.

If you are struggling to keep up or having problems with your flow, do not worry.

Everyone will be importing the part 1 solution provided below to use as the starting point for Flow 2.



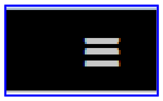
Flow 2 Starting point

[{"id":"31a61f50.5b8548","type":"tab","label":"Flow 3"},
{"id":"1186df17.507979","type":"inject","z":"31a61f50.5b8548","name":"","topic":"","payload":"","payloadType":"date","repeat":"","crontab":"","once":false,"x":105.27777099609375,"y":136.30001831054688,"wires":[["cb6b008c.fef0e"]]},
{"id":"cb6b008c.fef0e","type":"function","z":"31a61f50.5b8548","name":"Format message","func":"msg.payload = \"hi from John Doe\";\n\nreturn msg;","outputs":1,"noerr":0,"x":327.4721984863281,"y":139.17776489257812,"wires":[["8379426f.cbb51","b99ddb50.e3c97"]]},
{"id":"8379426f.cbb51","type":"debug","z":"31a61f50.5b8548","name":"my debugger 1","active":true,"console":"false","complete":"true","x":613.7777099609375,"y":143.73880004882812,"wires":[]},{"id":"b99ddb50.e3c97","type":"mqtt out","z":"31a61f50.5b8548","name":"publish to iot-summer-school/group-2","topic":"iot-summer-school/group-2","qos":"0","retain":"false","broker":"de850a9d.6952c8","x":286.111083984375,"y":260.0000305175781,"wires":[]},{"id":"fa49e949.94837","type":"mqtt in","z":"31a61f50.5b8548","name":"susbscribe to iot-summer-school/group-2","topic":"iot-summer-school/group-2","qos":"0","broker":"de850a9d.6952c8","x":279.4444580078125,"y":326.6666259765625,"wires":[["97725758.16c928"]]},
{"id":"97725758.16c928","type":"debug","z":"31a61f50.5b8548","name":"debug subscribe node","active":true,"console":"false","complete":"true","x":611.6666870117188,"y":312.22216796875,"wires":[]},{"id":"de850a9d.6952c8","type":"mqtt-broker","z":"","broker":"134.168.46.151","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":true,"willTopic":"","willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""}]

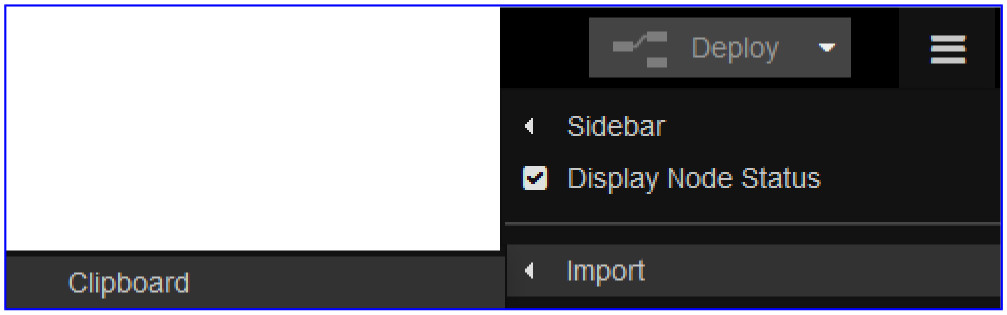
8 Lab part 3: Watson translate service

Step 1: Import Flow 2 starting point

Select the flow above “Flow 2 starting point” and copy it to the clipboard (Ctrl-C). Import the flow into Node-RED using by selecting the node-RED menu



and select the import from clipboard option.



You will be presented with a form in which you create nodes by entering json data.

Import your copied flow by pasting (Ctrl-P) from the clipboard into the the form

Step 2: Edit function node

Change from

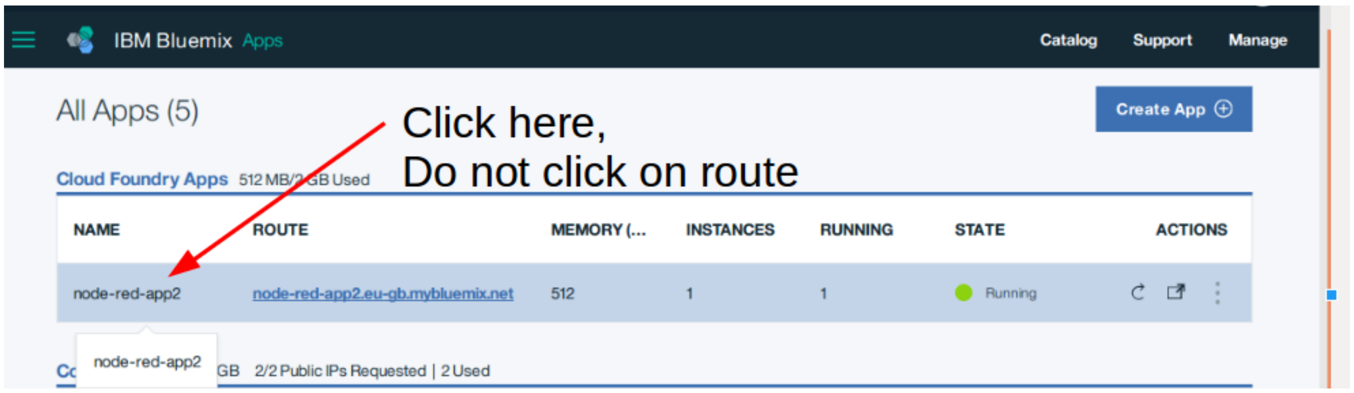
msg.payload = “hi from <your name>”

to

msg.payload = "你好，世界";

You are now going to use the Watson language translate service to convert msg.payload from Chinese into english.

Step 3: Go to Cloud Foundry App control panel

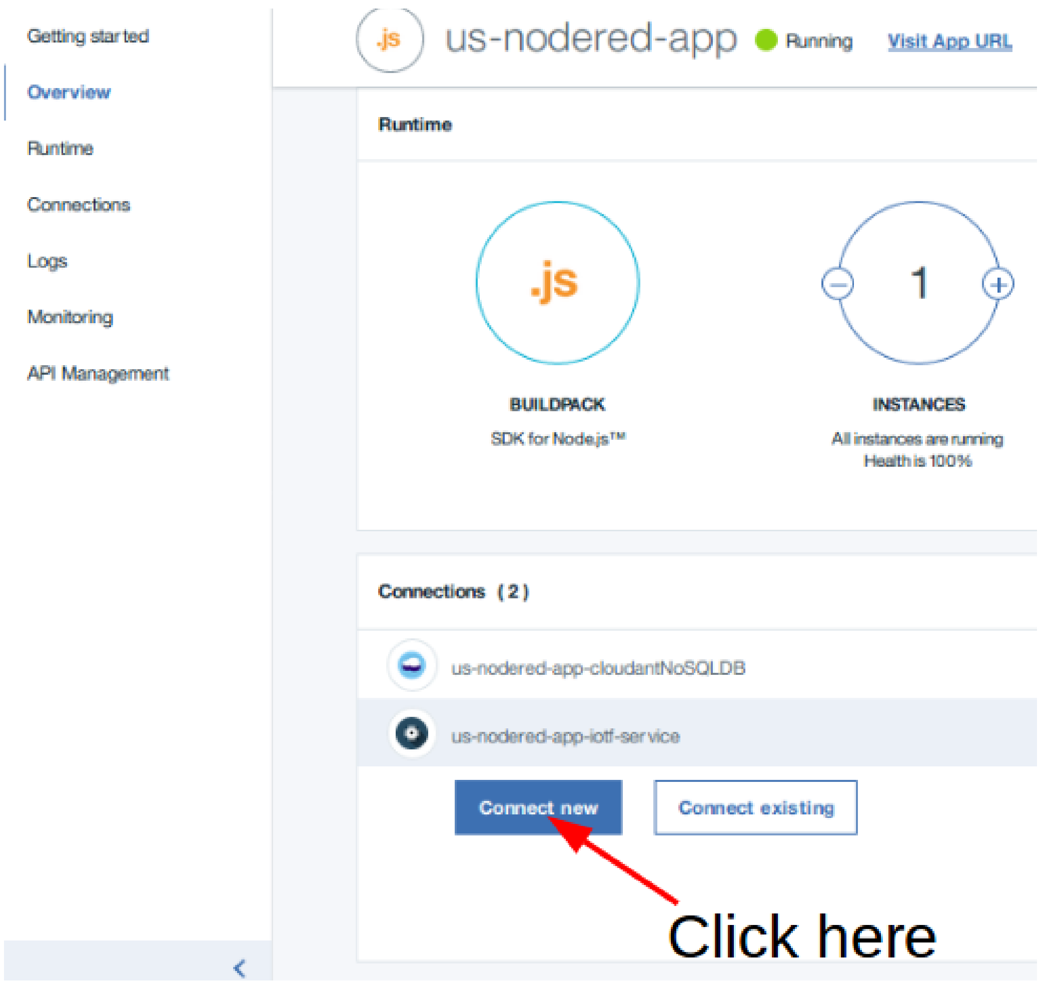


NAME	ROUTE	MEMORY (...	INSTANCES	RUNNING	STATE	ACTIONS
node-red-app2	node-red-app2.eu-gb.mybluemix.net	512	1	1	● Running	

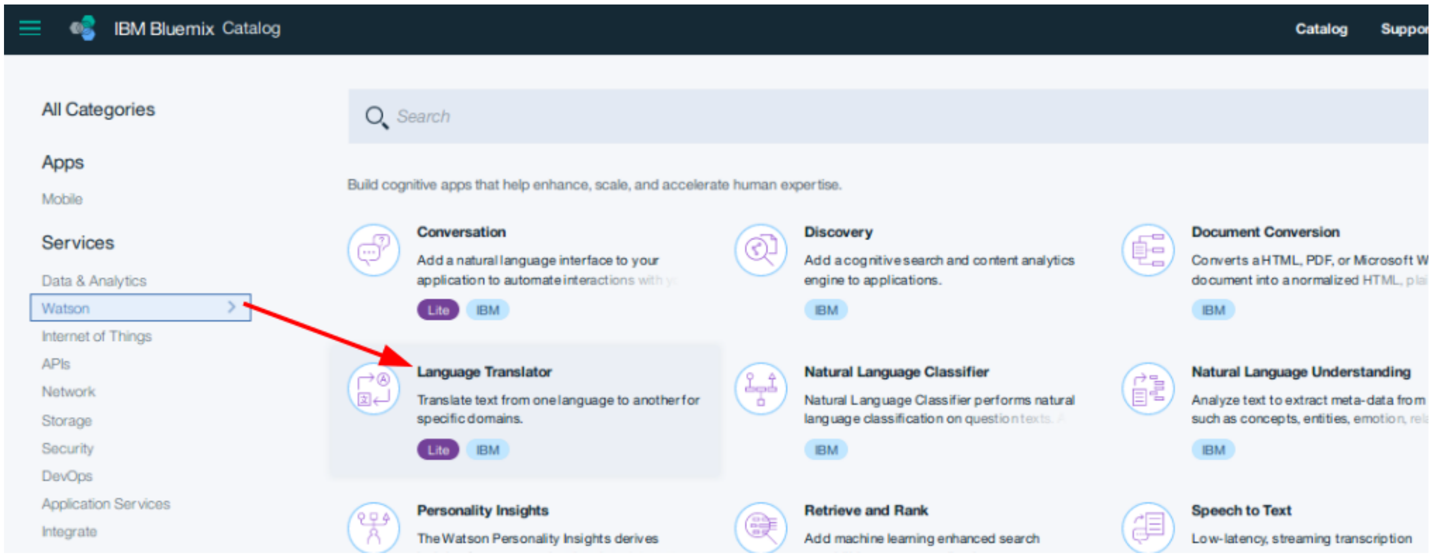
By clicking on route, you will be taken to another Node-RED flow editor screen.

However, we want to go to the Cloud Foundry App control panel instead.

Step 4: Click “Connect new” under Connections



Step 5: Select Language Translator



Build cognitive apps that help enhance, scale, and accelerate human expertise.

Conversation
Add a natural language interface to your application to automate interactions with y

Discovery
Add a cognitive search and content analytics engine to applications.

Document Conversion
Converts a HTML, PDF, or Microsoft W document into a normalized HTML, pl

Language Translator
Translate text from one language to another for specific domains.

Natural Language Classifier
Natural Language Classifier performs natural language classification on question texts. A

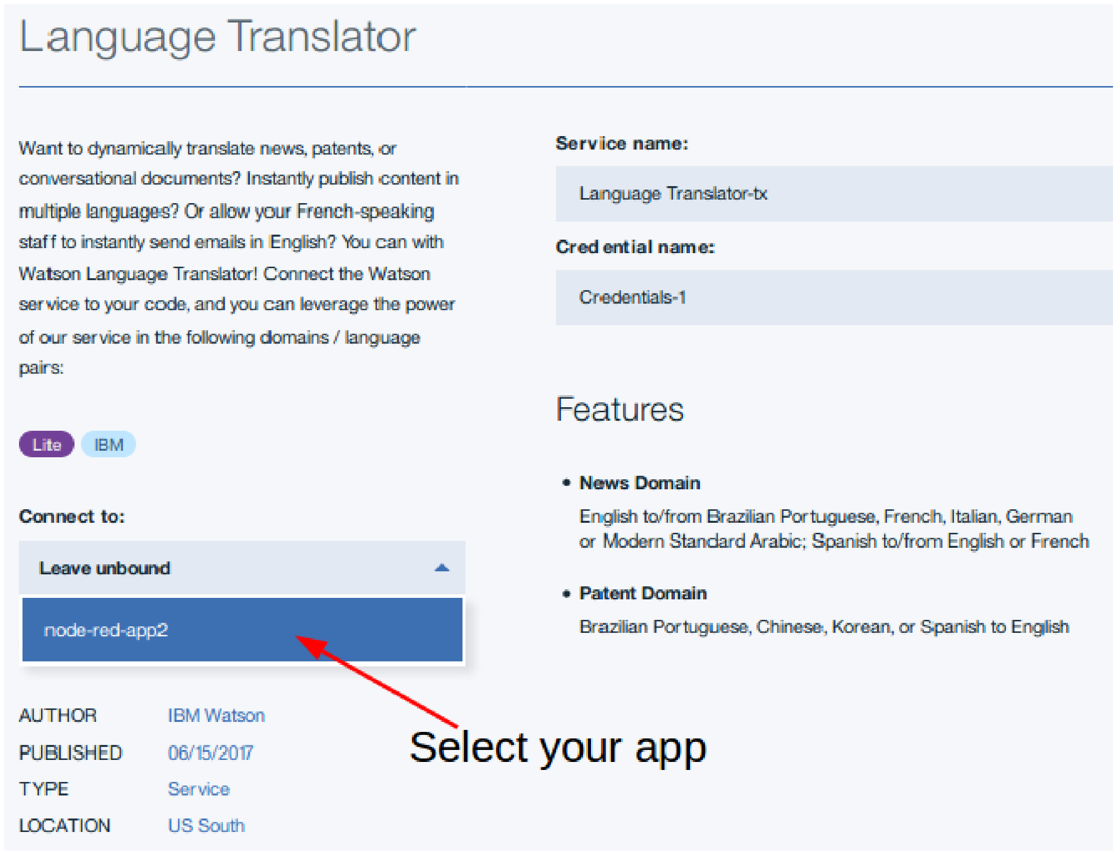
Natural Language Understanding
Analyze text to extract meta-data from such as concepts, entities, emotion, rel

Personality Insights
The Watson Personality Insights derives

Retrieve and Rank
Add machine learning enhanced search

Speech to Text
Low-latency, streaming transcription

Step 6: Connect translator service to Cloud Foundry App



Want to dynamically translate news, patents, or conversational documents? Instantly publish content in multiple languages? Or allow your French-speaking staff to instantly send emails in English? You can with Watson Language Translator! Connect the Watson service to your code, and you can leverage the power of our service in the following domains / language pairs:

Service name: Language Translator-tx

Credential name: Credentials-1

Connect to: Leave unbound

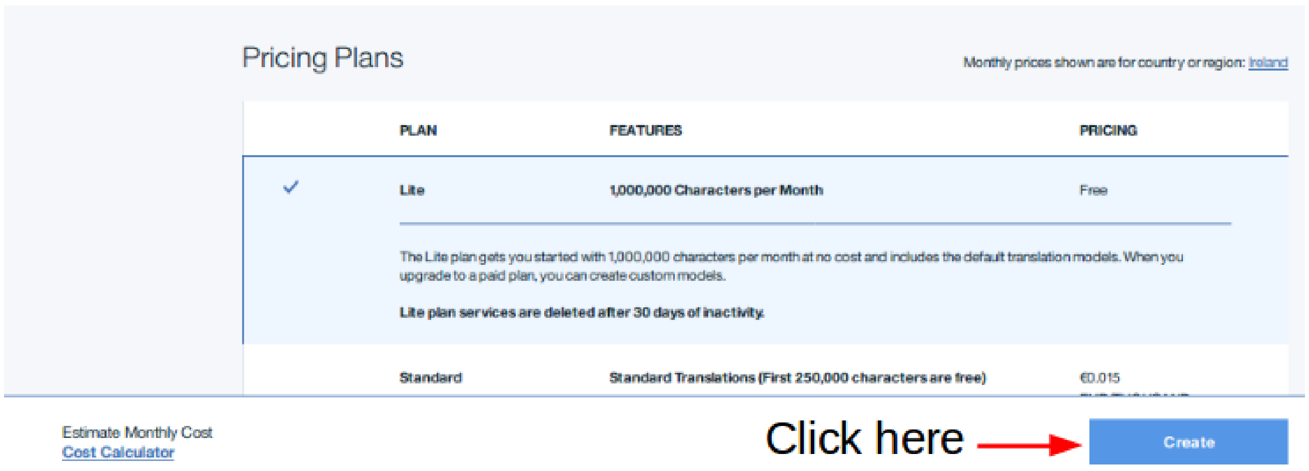
node-red-app2

- **News Domain**
  English to/from Brazilian Portuguese, French, Italian, German or Modern Standard Arabic; Spanish to/from English or French
- **Patent Domain**
  Brazilian Portuguese, Chinese, Korean, or Spanish to English

AUTHOR	IBM Watson
PUBLISHED	06/15/2017
TYPE	Service
LOCATION	US South

Step 7: Create translator service

In the bottom right hand side of the page. Click on “create” to create the service connected to your app.



Pricing Plans

Monthly prices shown are for country or region: Ireland

	PLAN	FEATURES	PRICING
✓	Lite	1,000,000 Characters per Month	Free
	Standard	Standard Translations (First 250,000 characters are free)	€0.015

The Lite plan gets you started with 1,000,000 characters per month at no cost and includes the default translation models. When you upgrade to a paid plan, you can create custom models.

Lite plan services are deleted after 30 days of inactivity.

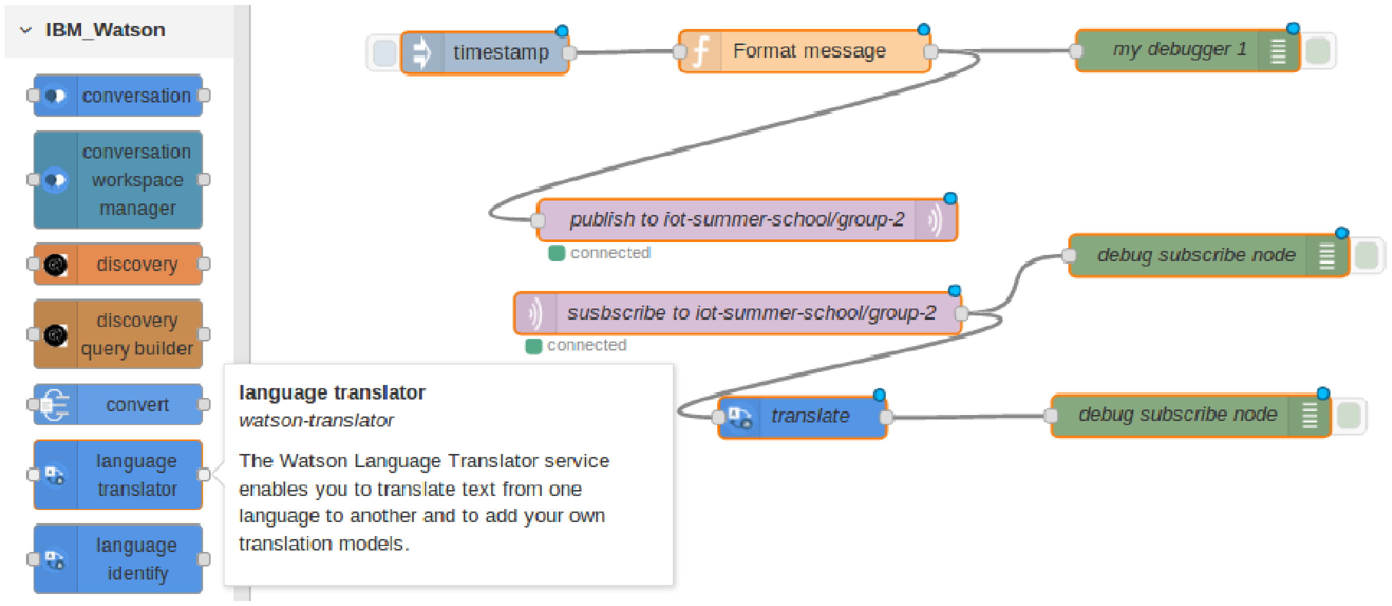
Estimate Monthly Cost
Cost Calculator

Step 8: Restart Cloud Foundry application

After creating a service you should get a prompt asking you to restart the application. Select yes.

Warning: If you do not restart your application after adding a service, you will not see the service in the list of available nodes in the Node-RED flow editor screen.

Step 9: Add Watson “language translator” node

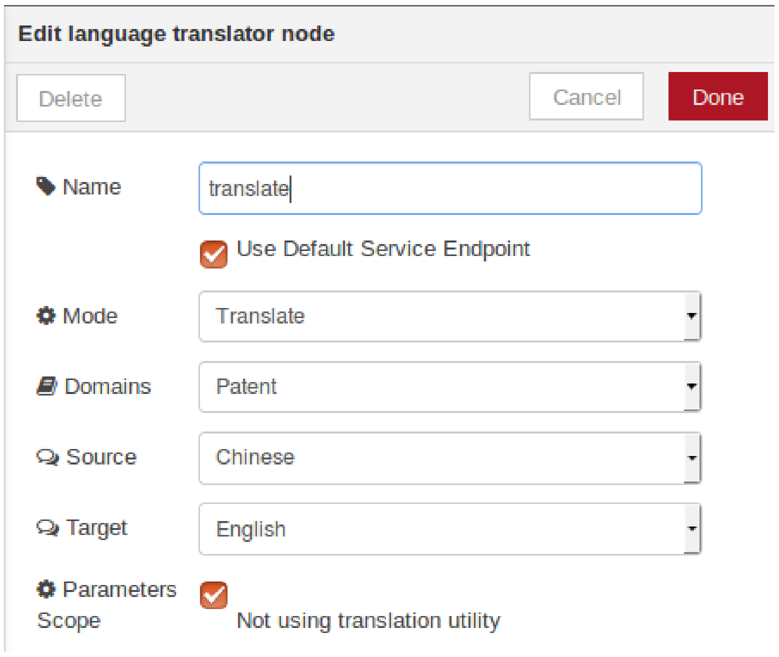


language translator
watson-translator

The Watson Language Translator service enables you to translate text from one language to another and to add your own translation models.

Add another debug node and wire it up to the output of the translate node (as shown in the above figure).

Step 10: Edit language translator node



Edit language translator node

Delete | Cancel | Done

Name: translate

✓ Use Default Service Endpoint

Mode: Translate

Domains: Patent

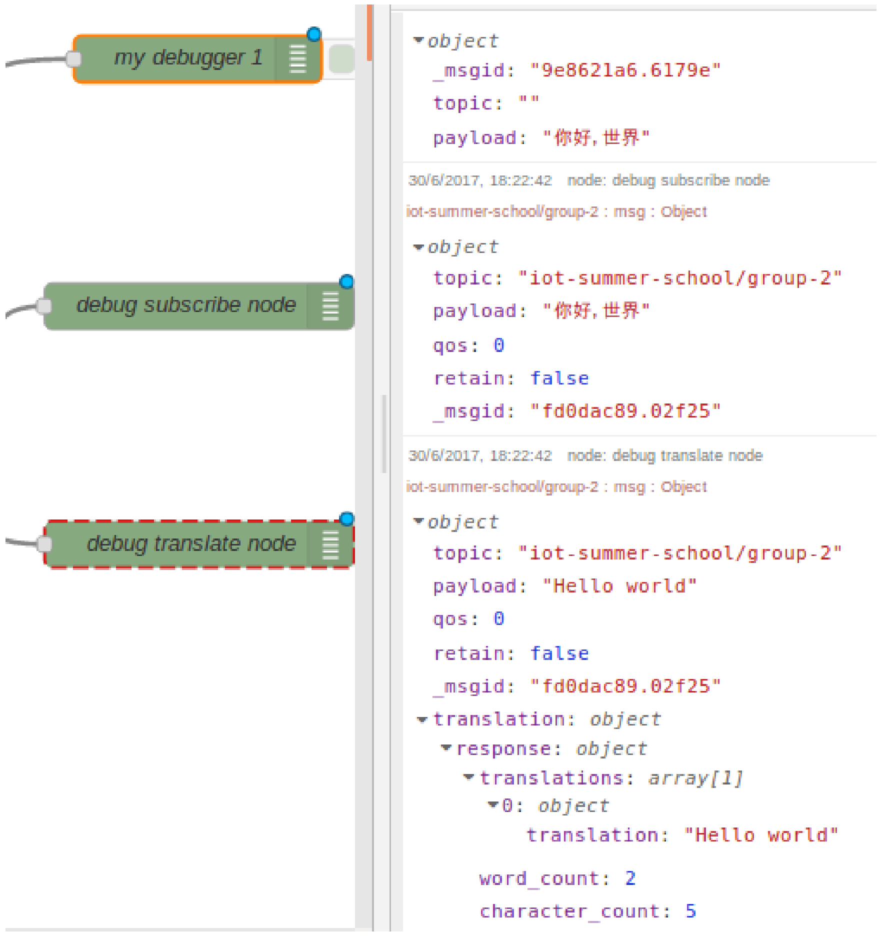
Source: Chinese

Target: English

Parameters Scope: ✓ Not using translation utility

Step 11: Deploy and test translate using debugger node

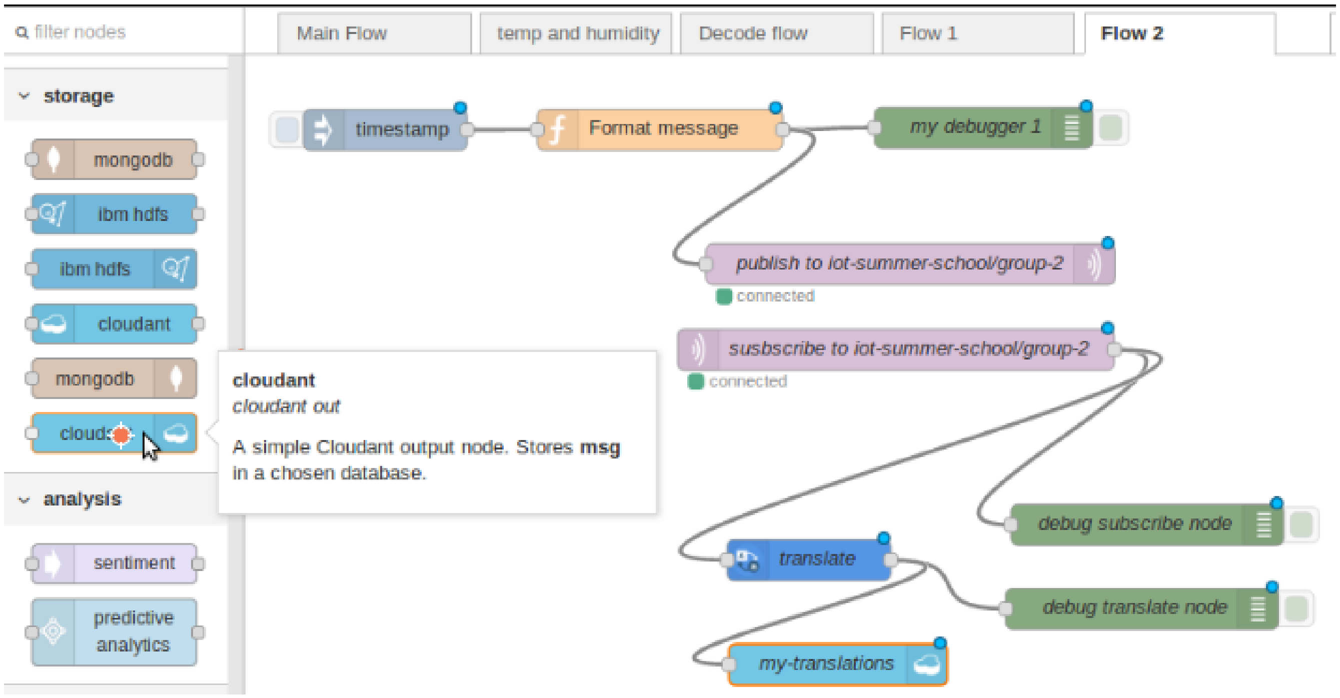
Remember to hit the deploy button after making a change. You should see the message “Hello world” as the result of inputting “你好，世界” into the Watson translate service.



```
▾object
  _msgid: "9e8621a6.6179e"
  topic: ""
  payload: "你好,世界"
30/6/2017, 18:22:42   node: debug subscribe node
iot-summer-school/group-2 : msg : Object
▾object
  topic: "iot-summer-school/group-2"
  payload: "你好,世界"
  qos: 0
  retain: false
  _msgid: "fd0dac89.02f25"
30/6/2017, 18:22:42   node: debug translate node
iot-summer-school/group-2 : msg : Object
▾object
  topic: "iot-summer-school/group-2"
  payload: "Hello world"
  qos: 0
  retain: false
  _msgid: "fd0dac89.02f25"
 ▾translation: object
   ▾response: object
     ▾translations: array[1]
       ▾0: object
           translation: "Hello world"

     word_count: 2
     character_count: 5
```

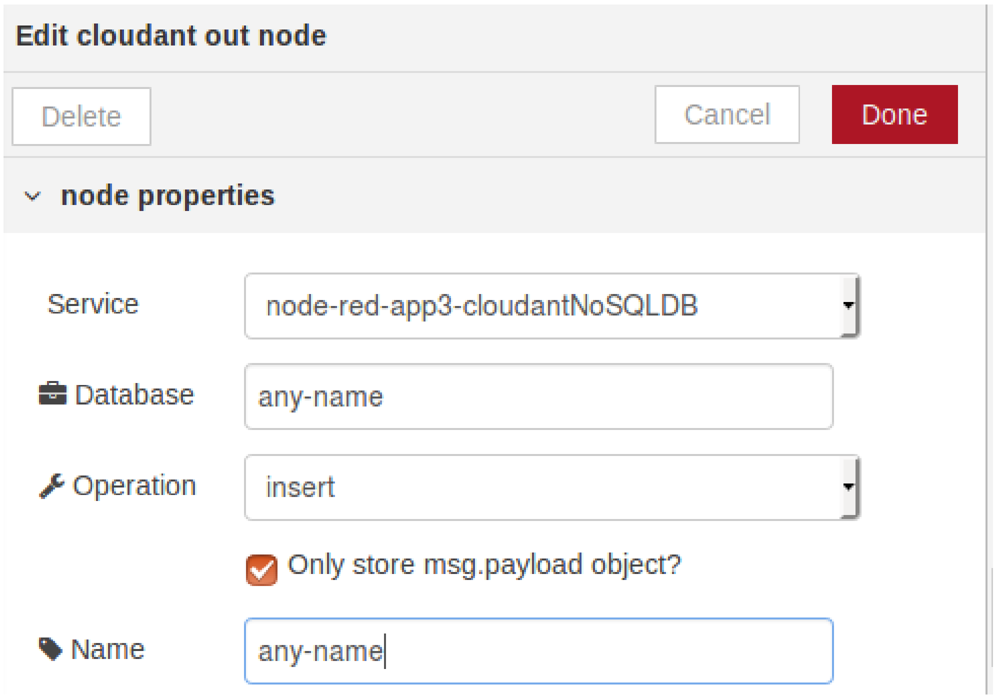
Step 12: Add Cloudant out node and connect to translate node



cloudant
cloudant out

A simple Cloudant output node. Stores **msg** in a chosen database.

Step 13: Edit Cloudant node



Edit cloudant out node

Service	node-red-app3-cloudantNoSQLDB
Database	any-name
Operation	insert
	✓ Only store msg.payload object?
Name	any-name

Step 14: Deploy and click the inject node

Click on the inject node to fire messages.

What is happening when I click inject now?

When you click on the inject node, a msg flows into the function node.

In this node the msg.payload value is being changed to “你好，世界”.

The message is then sent to an mqtt broker running on Bluemix.

An mqtt in node subscribes to the topic **“iot-summer-school/group<group number>/<your-name>”.**

Now the message is flowing from the “mqtt in node” to both the “debug subscribe node” and the “translate” node.

After translating msg.payload from Chinese to english, the msg.payload is then stored in cloudant.

9 Conclusion

After finishing each step in parts 1, 2 and 3 you have completed this lab.

In this lab you learned about:

-Creating your own flows

-Importing flows

-Mqtt pub/sub protocol

-Mqtt topic structure

-Adding services to Cloud Foundry Applications on Bluemix

-Analytic nodes such as language translate

-Cloudant nodes for storage

10 References

https://github.com/watson-developer-cloud/node-red-labs/blob/master/introduction_to_node_red/README.md

https://www.ibm.com/cloud-computing/bluemix/internet-of-things

https://docs.cloudant.com/json.html

http://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels

https://www.informatica.com/services-and-training/glossary-of-terms/extract-transform-load-definition.html

https://techcrunch.com/2010/09/03/cloudant/