IoT Summer School

OpenWhisk

Serverless Computing Lab

Table of Contents

1 About OpenWhisk

Cloud is a game-changer for app development, and serverless computing is revolutionizing developers' access to some of the most powerful services cloud has to offer, including cognitive intelligence, data analytics and Internet of Things.

In traditional cloud computing models, developers often have to maintain their infrastructure and worry about when and how fast to scale, as well as potential resiliency issues if they are deploying apps and features in multiple regions. They are also charged for computing power even when an app is idling —a particularly painful point for startups and small companies with limited resources and early growth applications.

Serverless computing, which many are hailing as the next era of cloud computing, relieves many of these hassles by abstracting away infrastructure, running code and scaling on-demand. For developers, serverless platforms with a strong cognitive stack, such as IBM Bluemix OpenWhisk, gives them unprecedented access to powerful services such as Watson APIs, the Watson IoT Platform and weather intelligence.

As one of the few serverless platforms built on open standards, OpenWhisk acts as the invisible force within apps, binding together relevant events, actions and triggers. As data continues to grow and proliferate across all industries, serverless is certainly primed to become a standard for resourcefulness, scalability and connecting into the power of cognitive and IoT tools running on the cloud.

https://thenewstack.io/future-serverless-3-startups-using-serverless-cognitive-iot-transform-industries/

2 Background Knowledge

What is Serverless Computing?

Serverless computing refers to a model where the existence of servers is simply hidden from developers. I.e. that even though servers still exist developers are relieved from the need to care about their operation. They are relieved from the need to worry about low-level infrastructural and operational details such as scalability, high-availability, infrastructure-security, and so forth. Hence, serverless computing is essentially about reducing maintenance efforts to allow developers to quickly focus on developing value-adding code.

Serverless computing encourages and simplifies developing microservice-oriented solutions in order to decompose complex applications into small and independent modules that can be easily exchanged.

Serverless computing does not refer to a specific technology; instead if refers to the concepts underlying the model described prior. Nevertheless some promising solutions have recently emerged easing development approaches that follow the serverless model – such as OpenWhisk.

The term ‘Serverless’ is confusing since with such applications there are both server hardware and server processes running somewhere, but the difference to normal approaches is that the organization building and supporting a ‘Serverless’ application is not looking after the hardware or the processes - they are outsourcing this to a vendor.

https://developer.ibm.com/openwhisk/what-is-serverless-computing/

3 Prerequisites

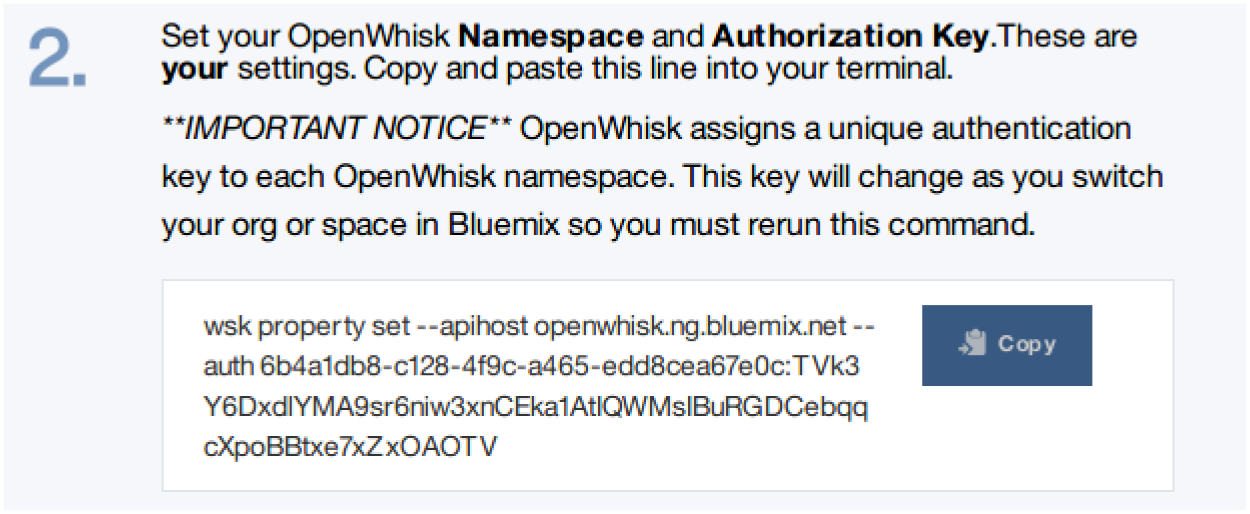
It is assumed you have completed the IoT Summer School Node-RED lab. At this stage you should have a Cloud Foundry Node-RED application in the United Kingdom domain.

Click the link provided below.

https://console.bluemix.net/openwhisk/learn/cli

Copy and paste the command "wsk propert set…."

Email this command to 16117743@studentmail.ul.ie



2. Set your OpenWhisk **Namespace** and **Authorization Key**.These are **your** settings. Copy and paste this line into your terminal.

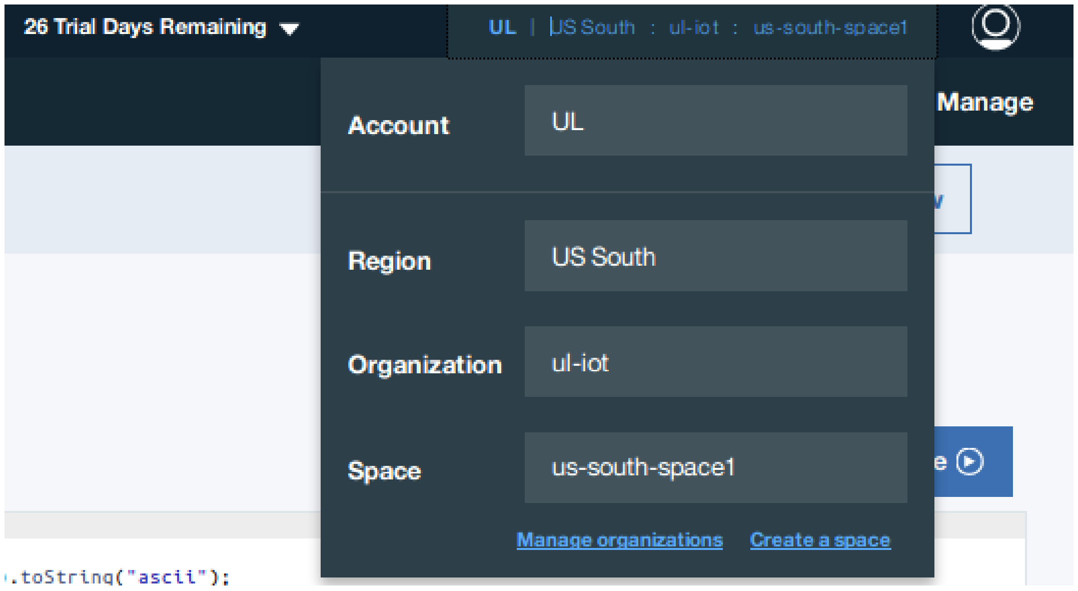
IMPORTANT NOTICE OpenWhisk assigns a unique authentication key to each OpenWhisk namespace. This key will change as you switch your org or space in Bluemix so you must rerun this command.

```
wsk property set --apihost openwhisk.ng.bluemix.net --
auth 6b4a1db8-c128-4f9c-a465-edd8cea67e0c:TVk3
Y6DxdlYMA9sr6niw3xnCEka1AtlQWMslBuRGDCebqq
cXpoBBtxe7xZxOAOTV
```

After receiving the email, the teaching assistant will use this command on their Ubuntu machine that has the OpenWhisk CLI installed. The command will generate the authorization code needed for your Node-RED flow to interact with your own OpenWhisk actions.

4 OpenWhisk Lab

OpenWhisk is currently only available in the US South region. During this lab you will need to use the Bluemix dashboard in order to switch between your OpenWhisk actions in the US South region and your Node-RED application in the United Kingdom region.



Account: UL
Region: US South
Organization: ul-iot
Space: us-south-space1

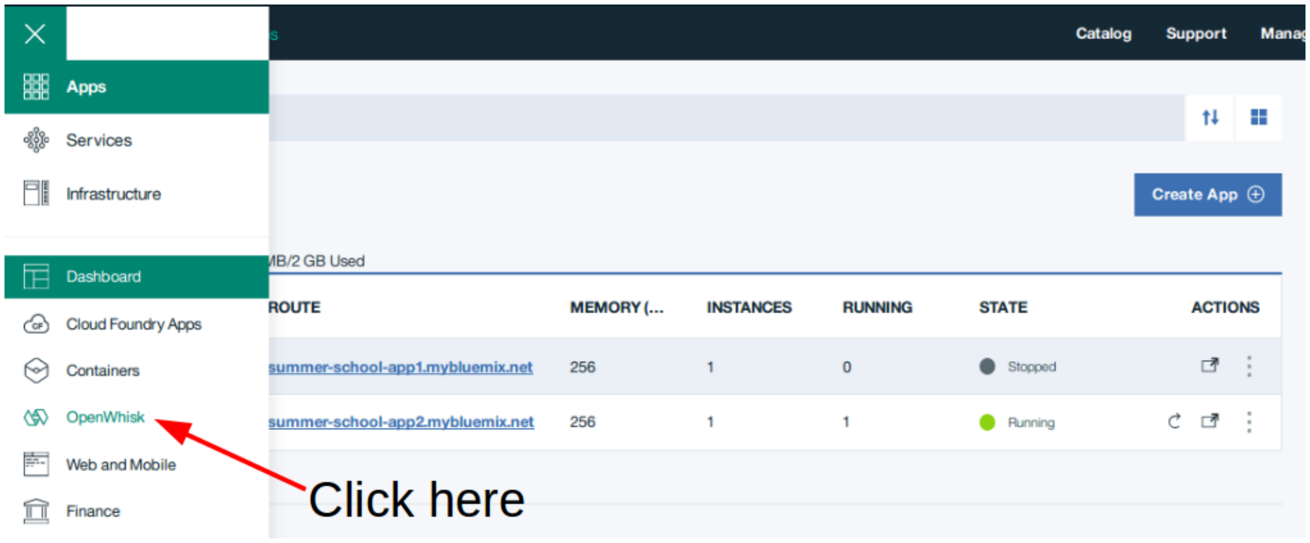
Should I just put my Node-RED application in the US South domain?

No. Your Node-RED application will be networking with a broker in the United Kingdom domain. The application would have significant latency and poor quality of service if deployed in the US South domain.

Does this mean there will be additional latency between my application and OpenWhisk actions, considering they are in two different regions?

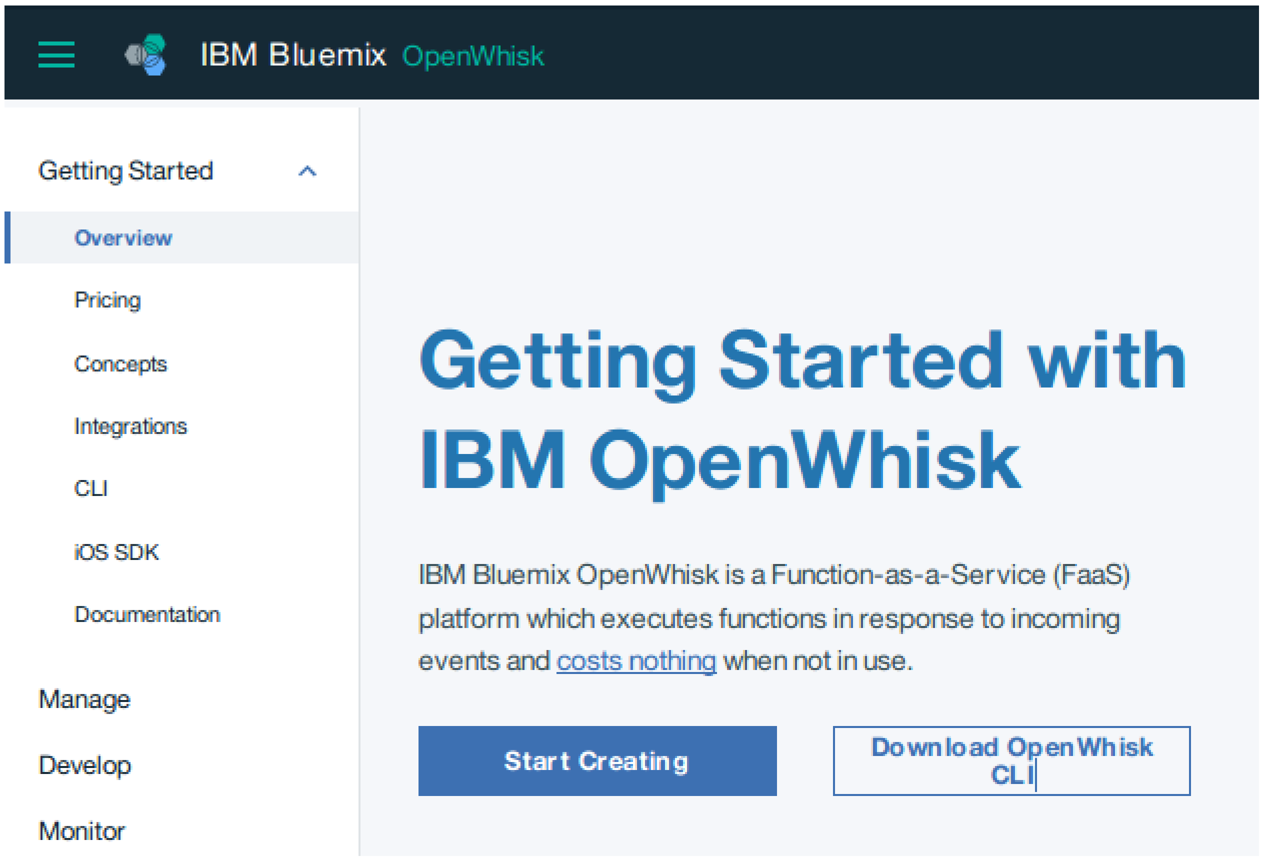
Yes. If your application has strict real time analytic constraints, consider measuring the latency of the OpenWhisk action against an analytics instance running in the United Kingdom domain. Based upon these results make a decision on your implementation.

Step 1: Click on OpenWhisk



ROUTE	MEMORY (...	INSTANCES	RUNNING	STATE	ACTIONS
summer-school-app1.mybluemix.net	256	1	0	● Stopped	
summer-school-app2.mybluemix.net	256	1	1	● Running	

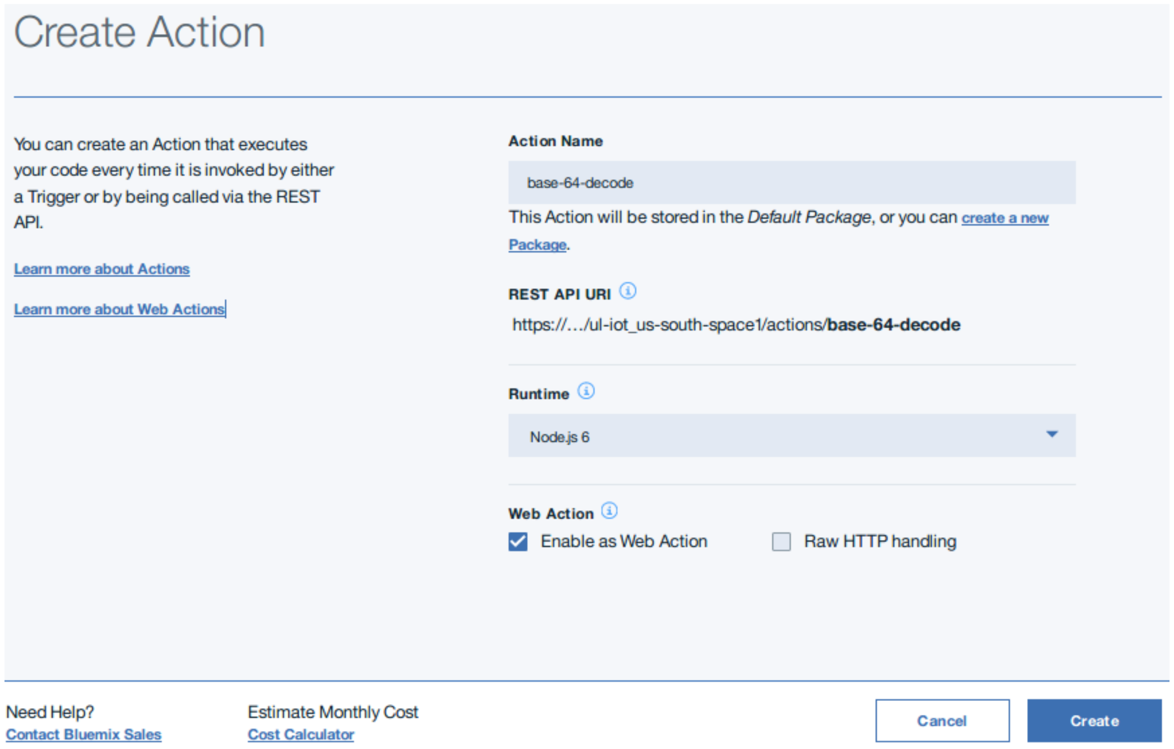
Step 2: Click on “Start Creating”



Getting Started with IBM OpenWhisk

IBM Bluemix OpenWhisk is a Function-as-a-Service (FaaS) platform which executes functions in response to incoming events and costs nothing when not in use.

Step 3: Create Action



Create Action

You can create an Action that executes your code every time it is invoked by either a Trigger or by being called via the REST API.

Learn more about Actions

Learn more about Web Actions

Action Name

base-64-decode

This Action will be stored in the *Default Package*, or you can create a new Package.

REST API URI

https://…/ul-iot_us-south-space1/actions/**base-64-decode**

Runtime

Node.js 6

Web Action

☑ Enable as Web Action ☐ Raw HTTP handling

Need Help?
Contact Bluemix Sales

Estimate Monthly Cost
Cost Calculator

Step 4: Enter base64 Node.js code

Type and do not copy the below code

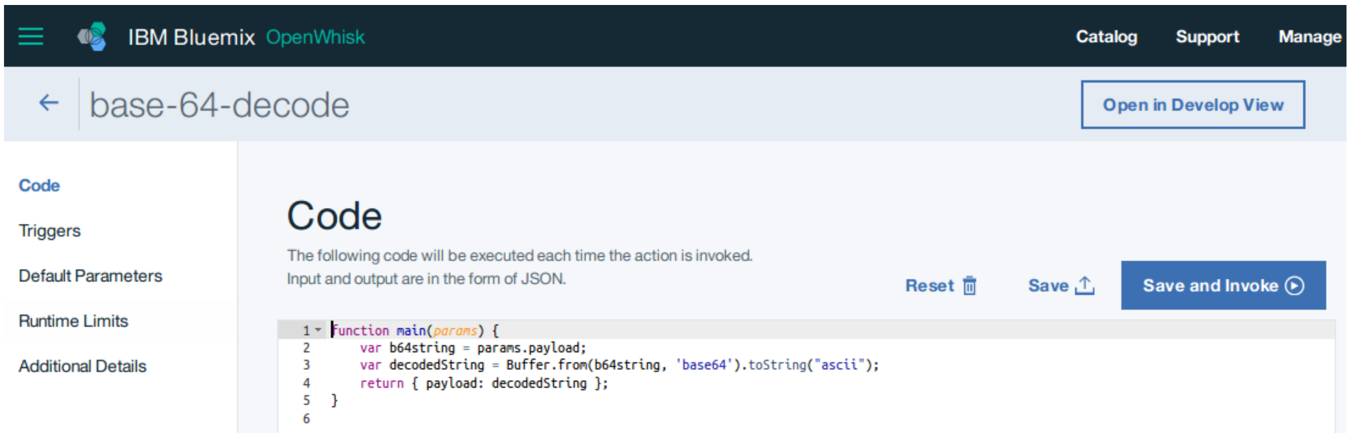
```
function main(params) {

        var b64string = params.name;

        var decodedString = Buffer.from(b64string, 'base64').toString("ascii");

        return { payload: decodedString };

}
```



base-64-decode

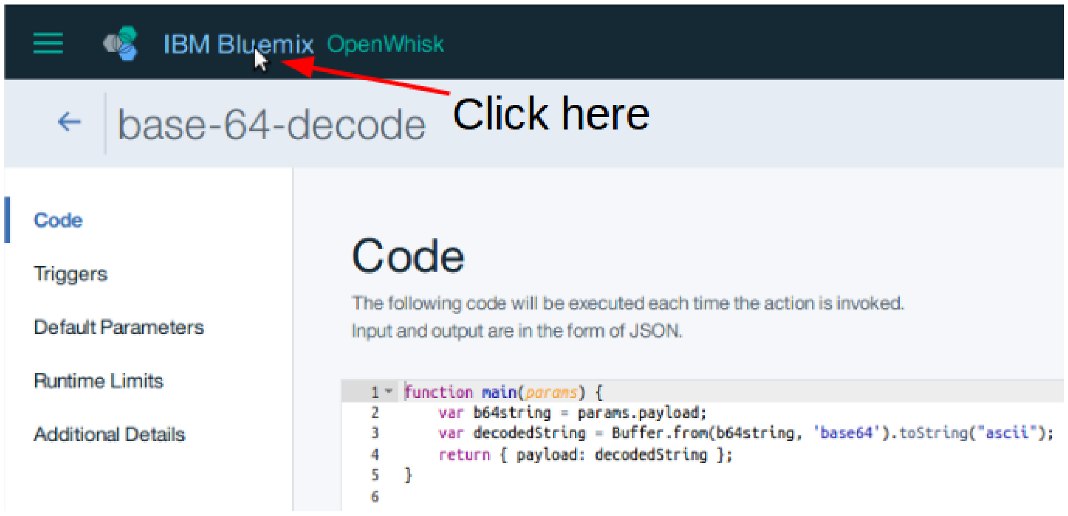
Code

The following code will be executed each time the action is invoked.
Input and output are in the form of JSON.

```
function main(params) {
    var b64string = params.payload;
    var decodedString = Buffer.from(b64string, 'base64').toString("ascii");
    return { payload: decodedString };
}
```

Step 5: Go back to Bluemix main page

After clicking on save in the previous step, click on Bluemix as shown in the figure below.



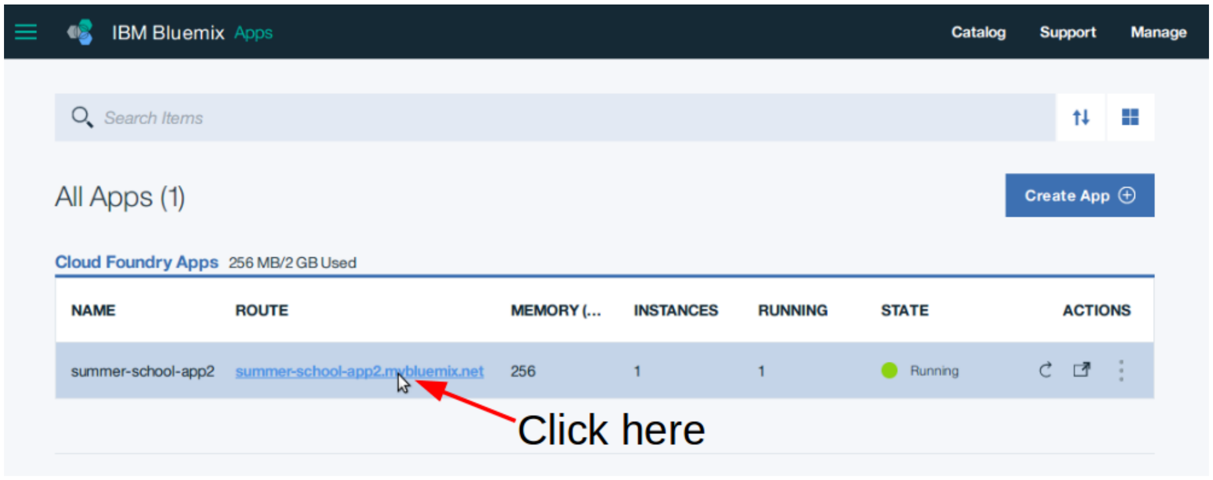
Code

The following code will be executed each time the action is invoked.
Input and output are in the form of JSON.

```
function main(params) {
    var b64string = params.payload;
    var decodedString = Buffer.from(b64string, 'base64').toString("ascii");
    return { payload: decodedString };
}
```

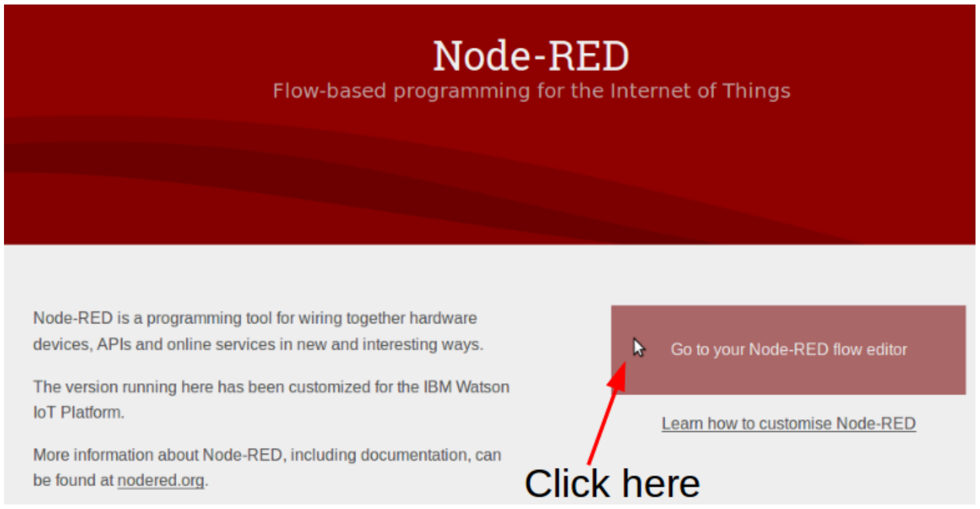
Step 6: Switch region and click on Cloud Foundry App Route

If you don't see your Cloud Foundry application please check to see if you are in the United Kingdom region.



NAME	ROUTE	MEMORY (...	INSTANCES	RUNNING	STATE	ACTIONS
summer-school-app2	summer-school-app2.mybluemix.net	256	1	1	● Running	

Step 7: Open Node-RED Flow editor in a separate tab


Node-RED
Flow-based programming for the Internet of Things
Go to your Node-RED flow editor
Learn how to customise Node-RED
Click here

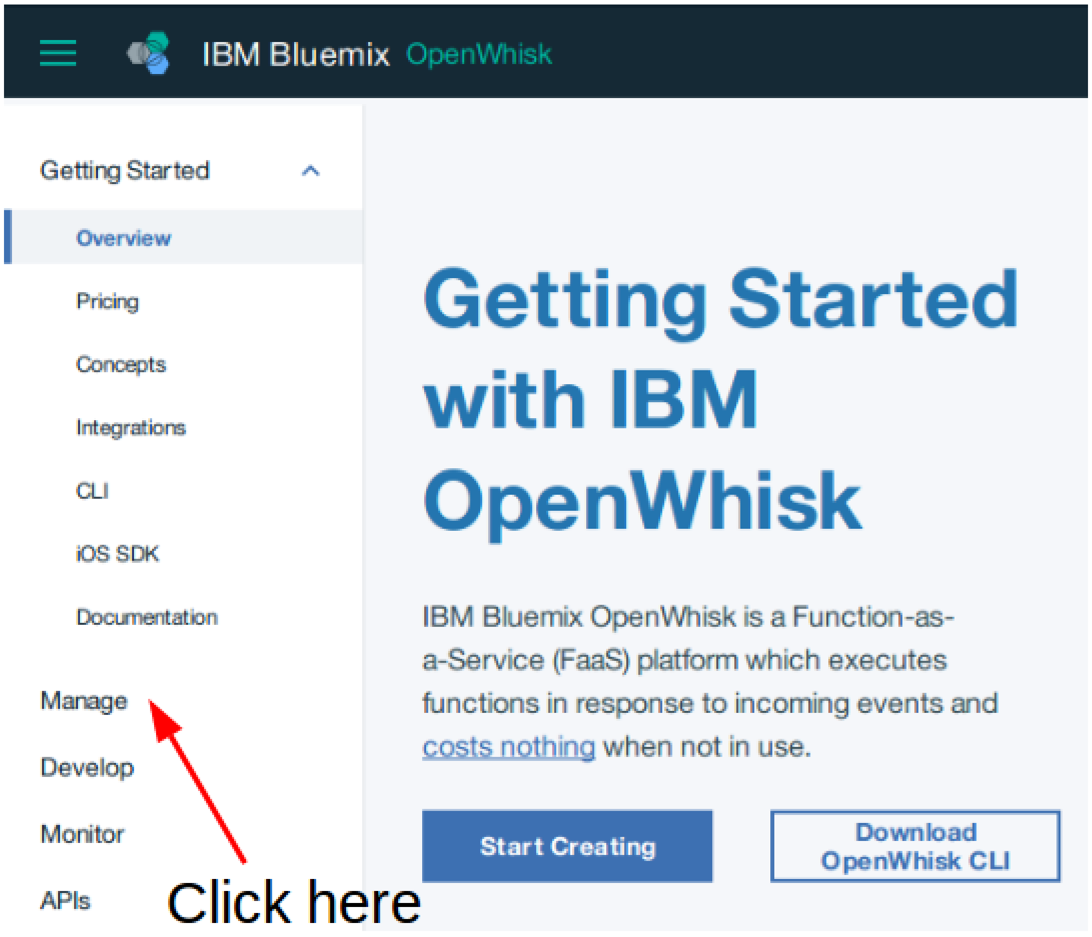
Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

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.

Step 8: Repeat step 1 to get back to the OpenWhisk screen

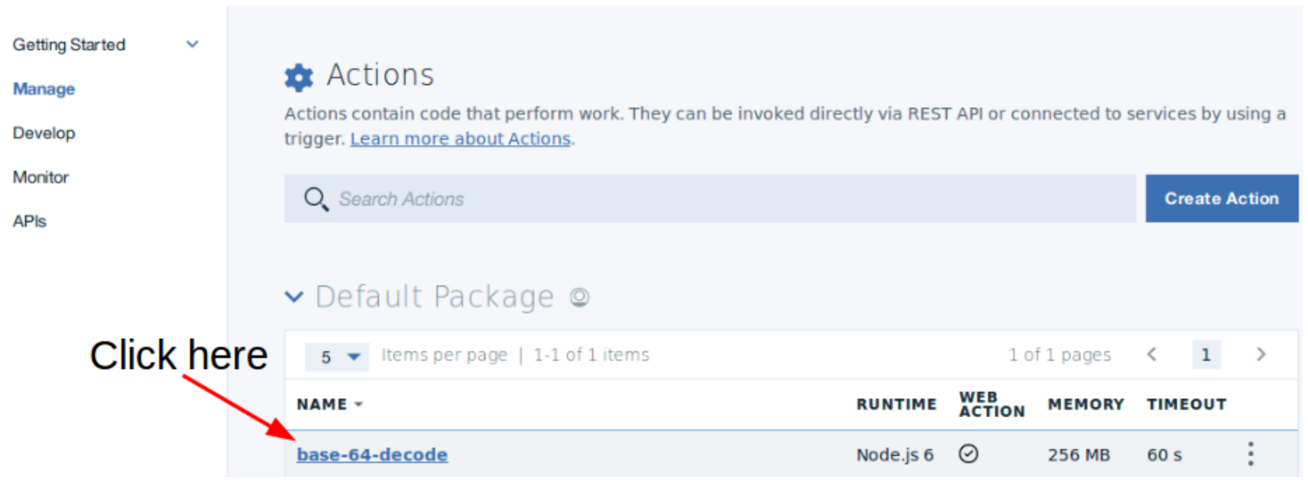
We want to be able to see our OpenWhisk Actions so we can implement changes. Do this by clicking on "manage". If you do not see OpenWhisk in the dashboard, check to see if you are in the US South region.



Getting Started with IBM OpenWhisk

IBM Bluemix OpenWhisk is a Function-as-a-Service (FaaS) platform which executes functions in response to incoming events and costs nothing when not in use.

Step 9: Click on base-64-decode action



Actions

Actions contain code that perform work. They can be invoked directly via REST API or connected to services by using a trigger. Learn more about Actions.

Default Package

5 Items per page | 1-1 of 1 items   1 of 1 pages

NAME	RUNTIME	WEB ACTION	MEMORY	TIMEOUT
base-64-decode	Node.js 6	✓	256 MB	60 s

Step 10: Import Node-RED flow

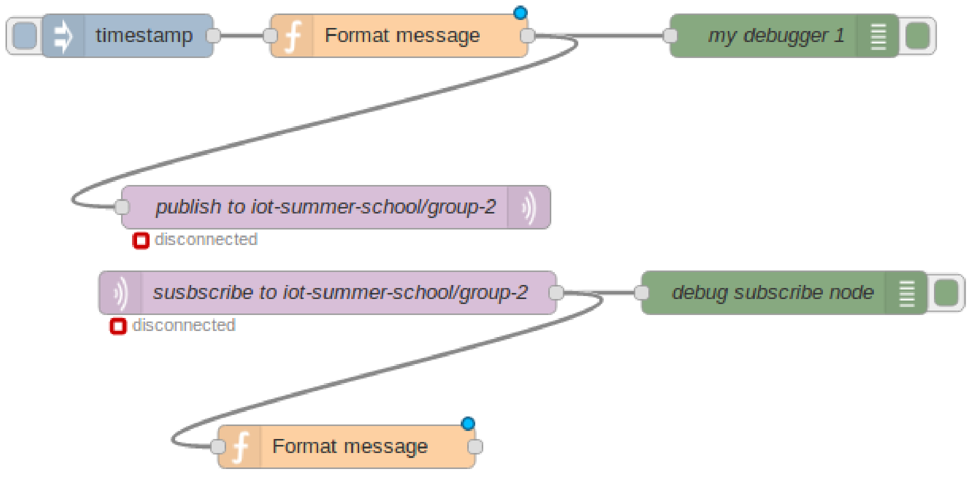
Switch to the tab that has the Node-RED flow.

Import the flow below.

[{"id":"8db20fe8.6ffa28","type":"tab","label":"Flow 3"},
{"id":"678eb277.25345c","type":"inject","z":"8db20fe8.6ffa28","name":"","topic":"","payload":"","payloadT
ype":"date","repeat":"","crontab":"","once":false,"x":102.88333129882812,"y":104.88333129882812,"wires"
:[["c3533458.8458c"]]},{"id":"c3533458.8458c","type":"function","z":"8db20fe8.6ffa28","name":"Format
message","func":"msg.payload = \"aGVsbG8gd29ybGQ=\";\n\nreturn
msg;","outputs":1,"noerr":0,"x":304.0777587890625,"y":104.76107788085938,"wires":
[["1575cb96.78f8bc","5ccb9ffb.4eb058"]]},
{"id":"1575cb96.78f8bc","type":"debug","z":"8db20fe8.6ffa28","name":"my debugger
1","active":true,"console":"false","complete":"true","x":580,"y":100,"wires":[]},
{"id":"5ccb9ffb.4eb058","type":"mqtt out","z":"8db20fe8.6ffa28","name":"publish to iot-summer-
school/group-2","topic":"iot-summer-school/group-
2","qos":"0","retain":"false","broker":"e639cd1b.be689","x":487.7165832519531,"y":203.58334350585938,"
wires":[]},{"id":"732958c3.79252","type":"mqtt in","z":"8db20fe8.6ffa28","name":"susbscribe to iot-
summer-school/group-2","topic":"iot-summer-school/group-
2","qos":"0","broker":"e639cd1b.be689","x":477.04998779296875,"y":266.24993896484375,"wires":
[["6e8658da.a96bf"]]},{"id":"6e8658da.a96bf","type":"function","z":"8db20fe8.6ffa28","name":"Format
message","func":"msg.payload = {\n   \"name\": msg.payload\n};\n\nreturn
msg;","outputs":1,"noerr":0,"x":470,"y":340,"wires":[["bc105a8.c4ed1a8"]]},
{"id":"bc105a8.c4ed1a8","type":"openwhisk-
action","z":"8db20fe8.6ffa28","name":"","func":"","namespace":"ul-iot_us-south-space1","action":"b64-
decode","params":[{"disabled":true}],"service":"937c50b4.19728","edit":false,"x":490,"y":420,"wires":
[["bd8e1c78.320a4"]]},
{"id":"bd8e1c78.320a4","type":"debug","z":"8db20fe8.6ffa28","name":"","active":true,"console":"false","co
mplete":"false","x":690,"y":420,"wires":[]},{"id":"e639cd1b.be689","type":"mqtt-
broker","z":"","broker":"IP-of-
Broker","port":"1883","clientid":"","usetls":false,"compatmode":true,"keepalive":"60","cleansession":true,"w
illTopic":"","willQos":"0","willPayload":"","birthTopic":"","birthQos":"0","birthPayload":""},
{"id":"937c50b4.19728","type":"openwhisk-service","z":"","name":"b64-
decode","api":"https://openwhisk.ng.bluemix.net/api/v1"}]

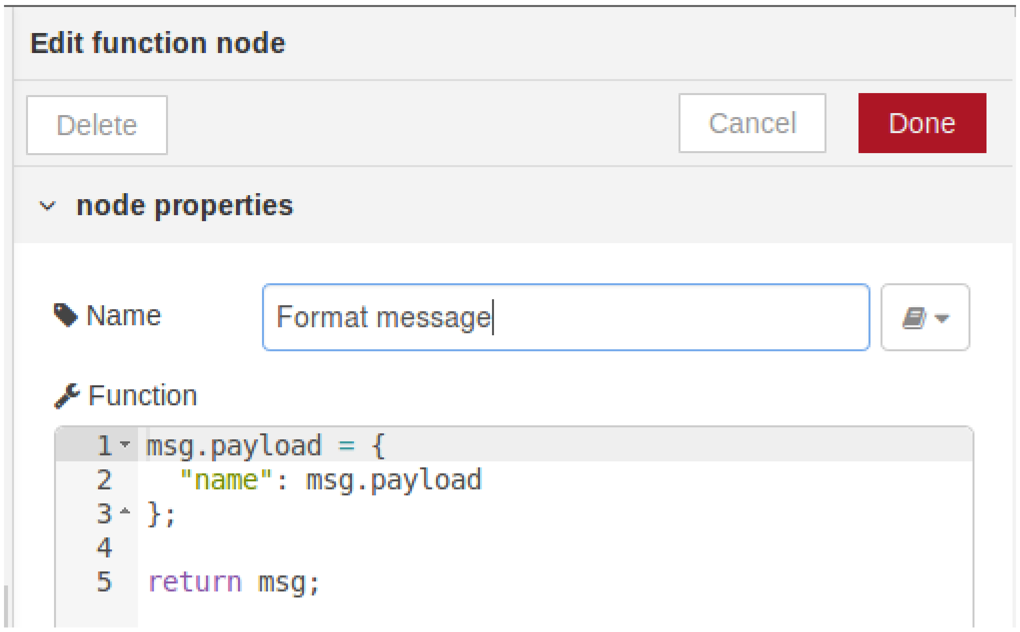
Step 11: Add a function node

Wire up the function node to the output of the mqtt subscribe node.



Step 12: Edit the function node

This function node changes the payload containing the base64 encoded value “hello world” into a key/value pair. This is required for the OpenWhisk action to be able to accept the arguments.



Name: Format message

Function

```
msg.payload = {
   "name": msg.payload
};

return msg;
```

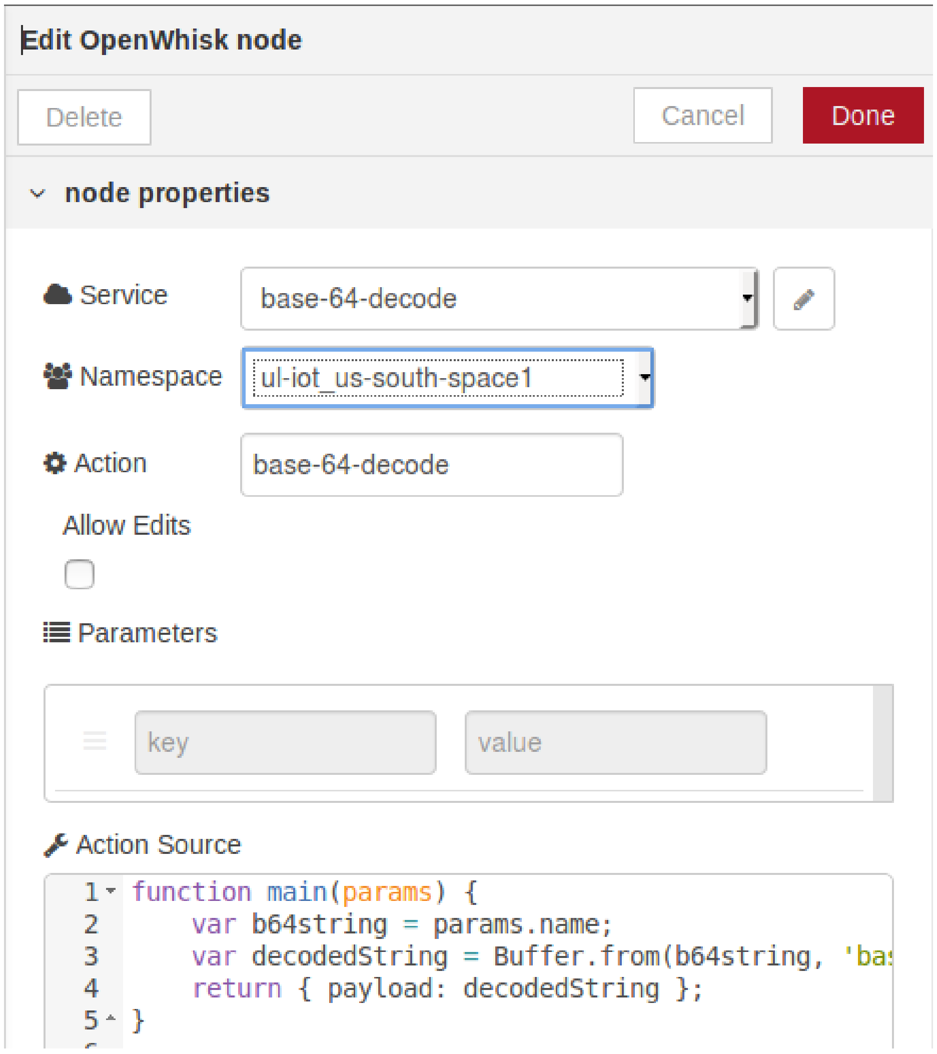
Step 13: Add OpenWhisk Action Node

The openwhisk-action node can be found under “function” node list. Drag the node onto the flow.

Connect the output of the newly added function node to the input of the openwhisk-action node.

Step 14: Edit OpenWhisk node

Click on the pencil icon to the right of the service drop-down menu.



Service: base-64-decode

Namespace: ul-iot_us-south-space1

Action: base-64-decode

Allow Edits

Parameters

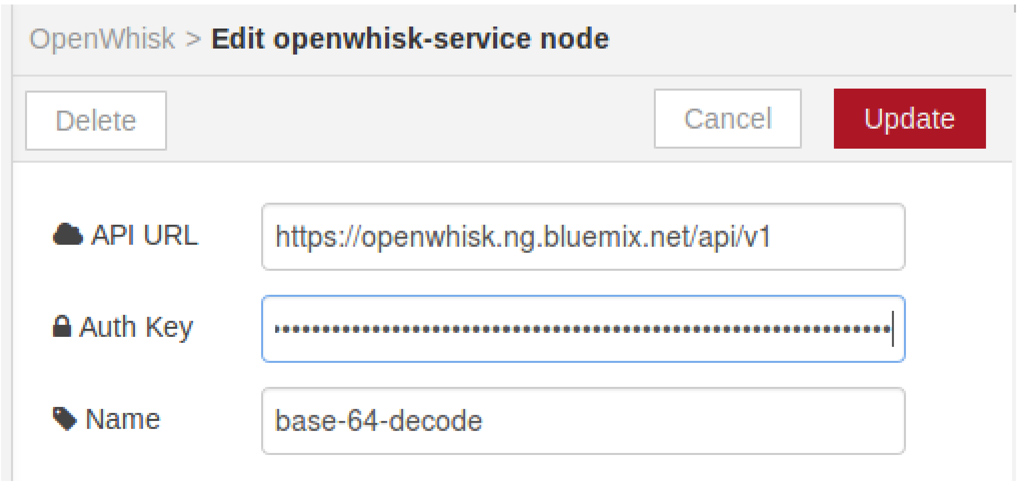
key | value

Action Source

```
function main(params) {
    var b64string = params.name;
    var decodedString = Buffer.from(b64string, 'ba
    return { payload: decodedString };
}
```

Step 15: Paste in the Auth Key obtained from the OpenWhisk CLI command

Make sure that the name field matches the name of your OpenWhisk action. If you are unsure about this step, please raise your hand and ask for assistance.



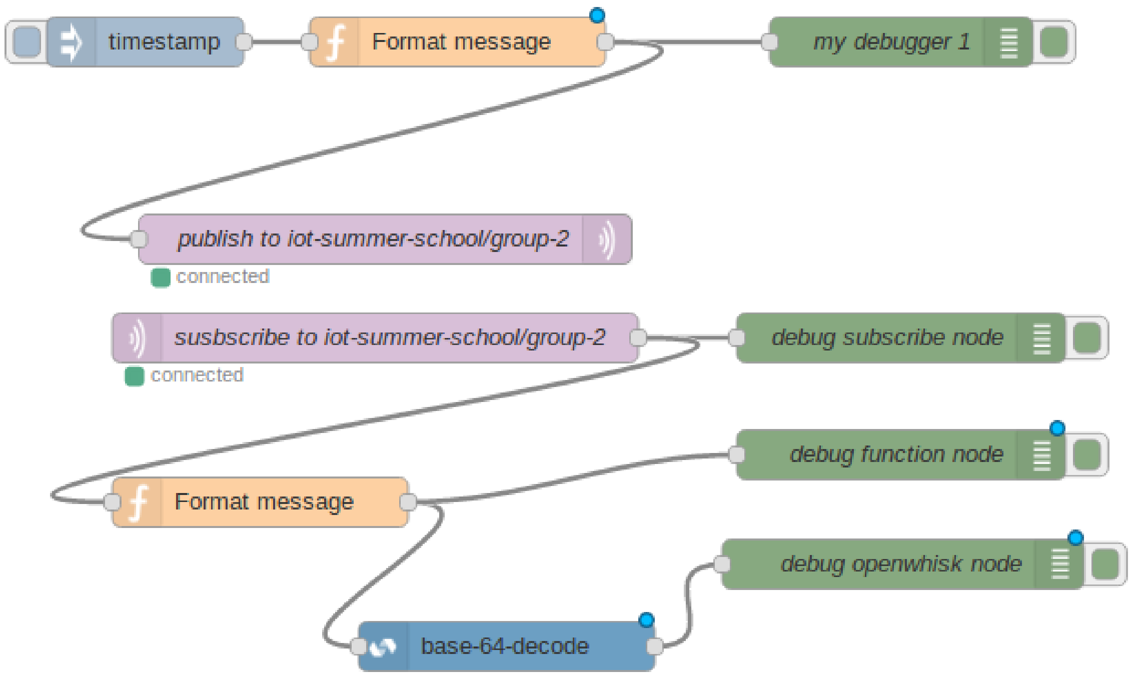
API URL: https://openwhisk.ng.bluemix.net/api/v1

Auth Key: ••

Name: base-64-decode

Step 16: Add debug nodes

Wire up the debug nodes as shown in the figure below.



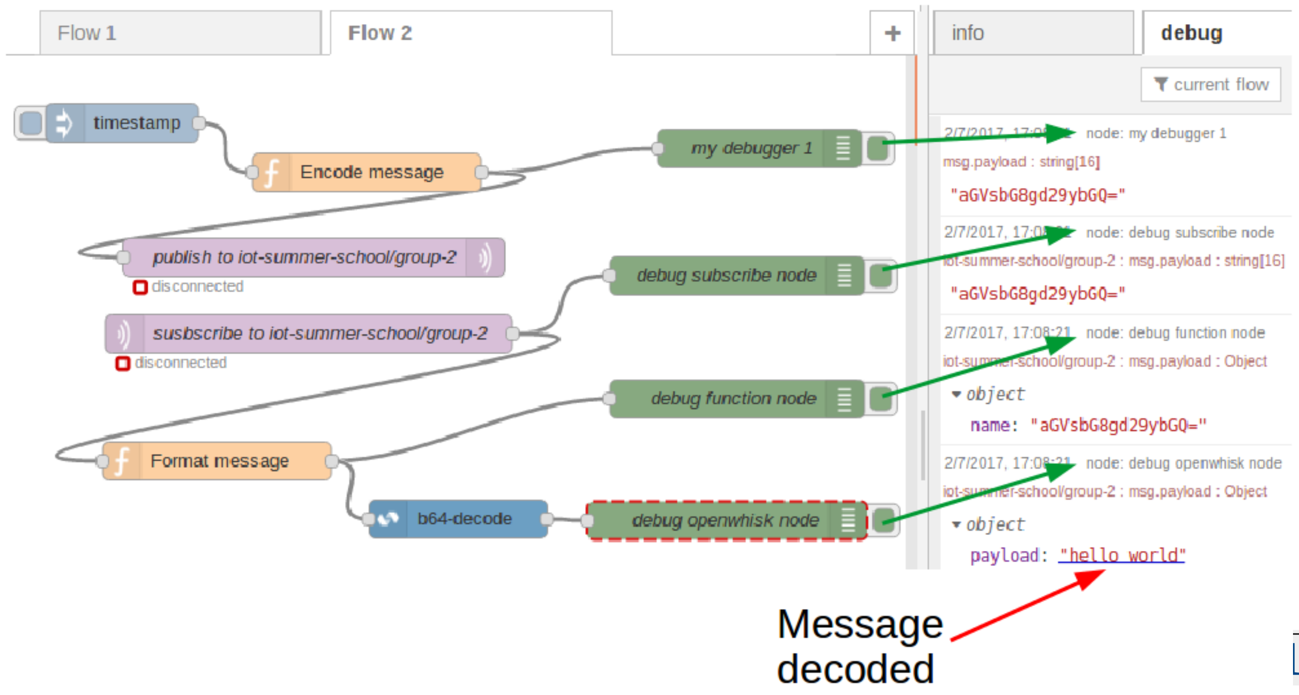
Step 17: Configure mqtt nodes

<u>Both</u> the mqtt nodes need to be configured. Change the topic and ip address field.

If you are unsure about this step, please raise your hand and ask for assistance.

Step 18: Deploy and test

Click the inject node to test the OpenWhisk action.



```
2/7/2017, 17:08:21  node: my debugger 1
msg.payload : string[16]
 "aGVsbG8gd29ybGQ="
2/7/2017, 17:08:21  node: debug subscribe node
iot-summer-school/group-2 : msg.payload : string[16]
 "aGVsbG8gd29ybGQ="
2/7/2017, 17:08:21  node: debug function node
iot-summer-school/group-2 : msg.payload : Object
 ▾ object
   name: "aGVsbG8gd29ybGQ="
2/7/2017, 17:08:21  node: debug openwhisk node
iot-summer-school/group-2 : msg.payload : Object
 ▾ object
   payload: "hello world"
```

5 FAQ

I am not understanding the encoding/decoding part of this lab?

In the first function node “Encode message” you are encoding the message “hello world” into base64.

The result of this is “aGVsbG8gd29ybGQ=”.

This value is assigned to msg.payload and is then sent to the mqtt broker using the “mqtt out node”.

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

This is the topic the “mqtt out node” published to.

The “mqtt in node” receives a message with a payload

msg.payload = “aGVsbG8gd29ybGQ=”

This message flows into the function node “format message” where the value is formatted into JSON.

```
msg.payload = {
        "name": msg.payload
};
return msg;
```

OpenWhisk accepts a single argument in JSON format. So we create a JSON key/value pair.

Where “name” is the key and “aGVsbG8gd29ybGQ=” is the value.

OpenWhisk can access this value using **params.name** as seen in the OpenWhisk action code below.

```
var b64string = params.name;
var decodedString = Buffer.from(b64string, 'base64').toString("ascii");
return { payload: decodedString };
```

The above code decodes the value “aGVsbG8gd29ybGQ=”, the result of which is “hello world”.

This value decodedString is then returned.

I am not seeing OpenWhisk in the dropdown menu?

Check the top right hand corner to see if you are in a US-south region. If you are in the US-South region and still cannot see an option for OpenWhisk, call over the teaching assistant.

How might I use OpenWhisk for my analytics application?

One use case would be to have a trigger attached to an mqtt feed. For example

Listening to topic **“iot-summer-school/temperature”**

Every time a message is published to this topic, the associated action is triggered by the “trigger node”.