

CREATING A TWITTER DASHBOARD USING IBM CLOUD

Mandie Quartly, IBM
Latest update: 13/02/2019

TABLE OF CONTENTS

Pre-requisites	2
Overview	2
Step 1 – create your node-red application.....	3
Step 2 – initial set up for your Node-RED application.....	5
Step 3 – a quick overview of the Node-RED interface	6
Step 4 – create a Hello World flow.....	7
Step 5 – using Watson services for further analytics	8
Step 7 – creating a dashboard – prerequisites.....	13
Step 8 – creating the Twitter dashboard.....	14
Step 9 - Other flows to explore.....	18
useful links / further information	19

PRE-REQUISITES

- An IBM Cloud account
 - Upgraded to “Trial” level using IBM Academic Initiative
 - See this guide for details: <https://ibm.box.com/s/qn3ui3ydxogd6ro5ry8x1unsj8gutoih>
- Internet access
- A Twitter account & developer access (instructions included to create the latter if required)
 - To access the Twitter API and real time tweets

OVERVIEW

The aim of this session is to introduce you to IBM Cloud, the Node-RED application and to enable you to create a dashboard based on a stream of tweets in real time.

For reference, elements of this session have been adapted from:

- Github Node-RED page from John Walicki:
<https://github.com/johnwalicki/Node-RED-Twitter-Workshop>
- developerWorks “Node-RED: Basics to bots” course:
<https://developer.ibm.com/courses/all/node-red-basics-bots/>

USEFUL MATERIAL (OPTIONAL)

Node-RED introduction video (5 mins): <https://youtu.be/vYreeoCoQPI>

Useful links:

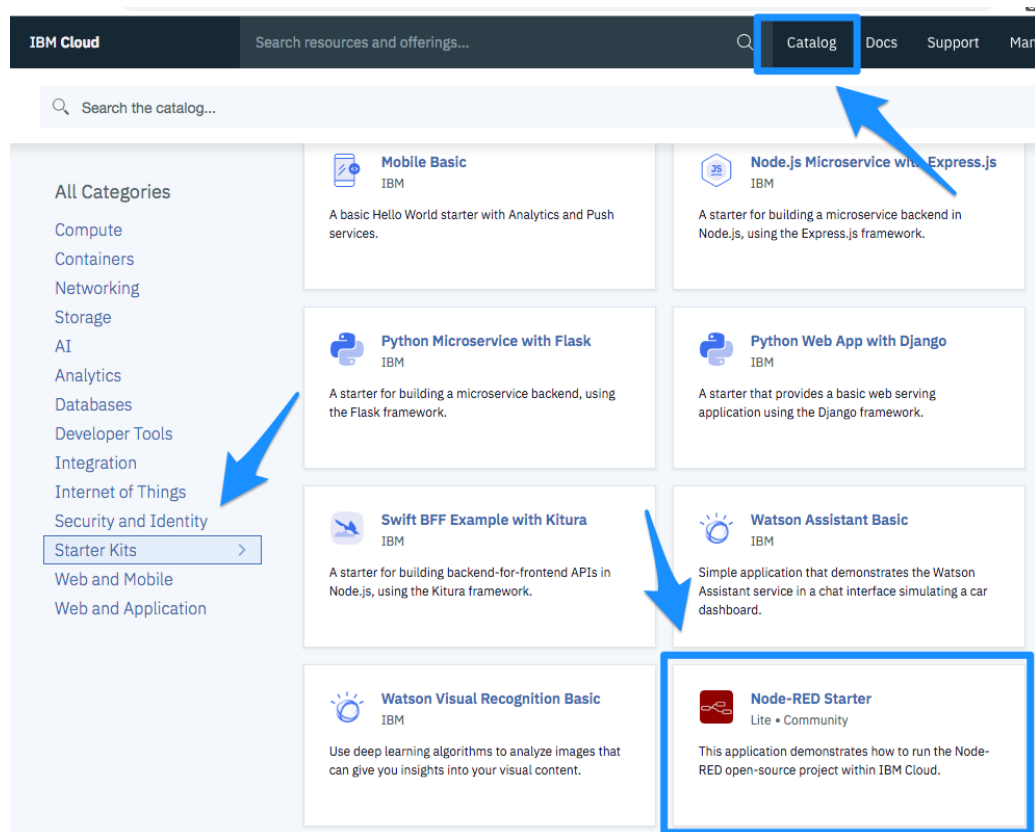
- Documentation <https://nodered.org/docs/>
- Node-RED Github <https://github.com/node-red>
- Watson Developer Cloud <https://github.com/watson-developer-cloud/>
- The Node-RED Community is very active and helpful, with further links to Slack / GitHub / Stack Overflow etc: <https://nodered.org/#community>

STEP 1 – CREATE YOUR NODE-RED APPLICATION

Log into IBM Cloud and go to your Dashboard: <https://cloud.ibm.com/>. This gives an overview of your resources / applications / Cloud statue. At this point, there will likely be nothing in here.

Let's create our base app using a Starter Kit (a template that includes one application and its associated environment and predefined services for a particular domain). Go to "Catalog" and choose "Starter Kits".

Choose the Node-RED Starter.



Choose a name for your app – this needs to be unique for the domain you are using, since this will form the first part of the URL to access your app. You should be able to leave everything else as the default values.

App name:

Host name: **Domain:**

Choose a region/location to deploy in: **Choose an organization:** **Choose a space:**

Tags: ⓘ

Selected Plan:

SDK for Node.js™ **Cloudant**

Then hit “Create”. This will take a few minutes to create and start your new Node-RED application.

Feel free to look at the details of your new Node-RED application via the links on the left hand side of the page. Don’t worry about following the instructions on the next page.

STEP 2 – INITIAL SET UP FOR YOUR NODE-RED APPLICATION

Go to your Resource list – which will include your newly created Node-RED application under Cloud Foundry Apps. It may take a few seconds for this list to populate.

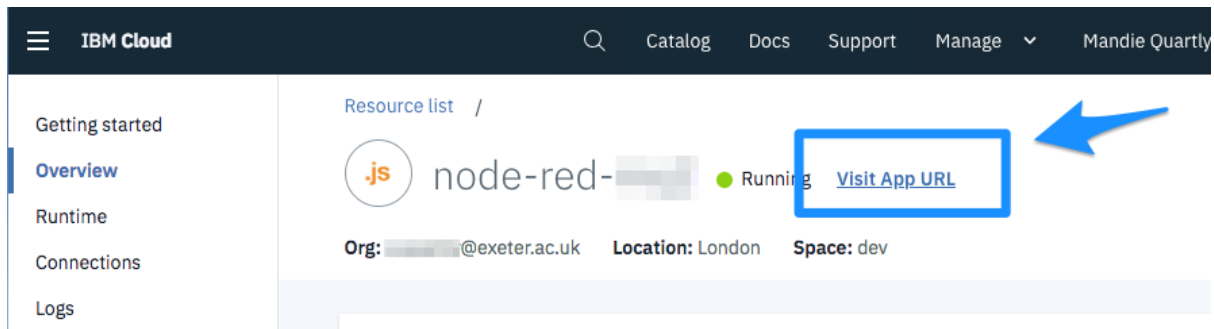
Resource list can be accessed via various routes:

- Via the navigation menu found via the “hamburger” in the top left hand corner next to the IBM Cloud logo.
- Via your Dashboard using “View all” in the Resource summary.
- Or via <https://cloud.ibm.com/resources>

After a delay (might be a few minutes) the state of the application will show as green / running. Open the details of your Node-RED application by clicking on its name in the resource list...



... and then “Visit App URL” on the detailed view.



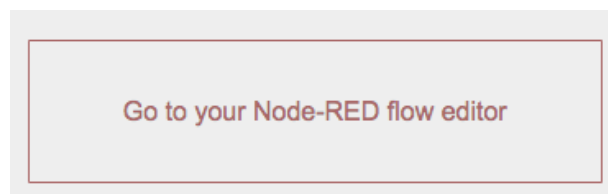
We need to do some one time set up to secure your access (since anyone with your URL can access this application currently).

Hit “Next” on the Welcome page and then create a username and password to secure your application. You’ll need these to log into your new app, make sure you take note of these!

Username	
Password	

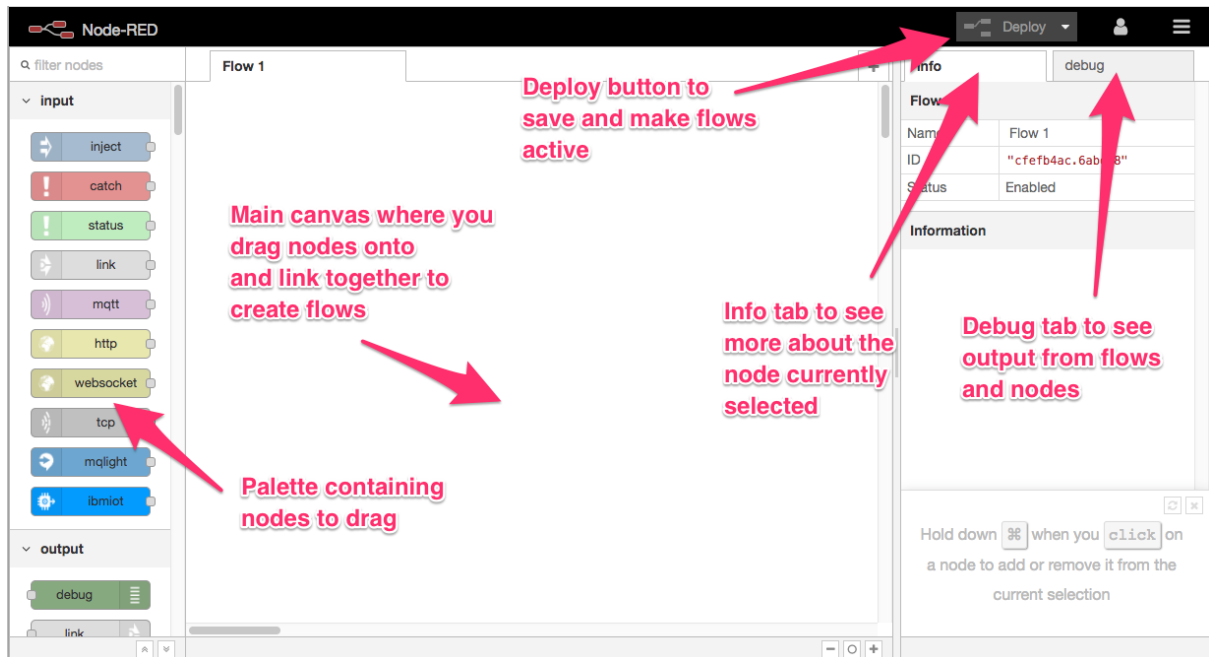
Hit “Next” to save the username and password. Have a browse of the available nodes on the next page – now you start to get an understanding of some of the add-ons available for Node-RED. Finally hit “Finish” on the last page. Your settings are applied and you get access to the main application.

Choose to go to the Node-RED editor on the following page, and use the username and password you previously defined to log into Node-RED.



STEP 3 – A QUICK OVERVIEW OF THE NODE-RED INTERFACE

Log into your Node-RED editor and have a look at the interface. See notes on the screenshot below.



Have a look at the variety of nodes which are available by default on the left palette. Note that you can filter them using the search bar at the top of the palette.

STEP 4 – CREATE A HELLO WORLD FLOW

This program is a very simple flow that outputs the message 'Hello World' on the debug tab. To build this 'Hello World' flow take the following steps:

1. Drag an 'Inject node' from the left-hand palette to the canvas.
2. Double click this node to see the options. Use the drop-down for Payload, select string and type 'Hello'. This will inject hello into the flow as the Payload (data passed to the next node) when the inject node is clicked. Click on Done, to save and close this node.
3. Add a 'Function node' to the canvas, open it and place the following text on the first line of the main Function box (leave Name blank, or add a descriptive name of your choice):

```
msg.payload += " World"
```

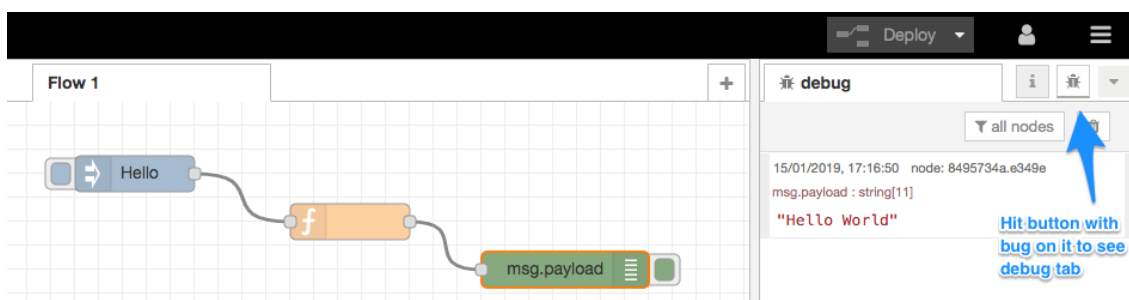
This will add the string “World” to the payload. The complete function should look like this:

```
Function
1 msg.payload += " World"
2 return msg;
```

Hit Done to save and close the function node.

4. Add a 'Debug node' to the canvas. The debug node writes to the debug tab, which helps you monitor the flow through your application. If you double click you'll see the default output is msg.payload – which we will keep. Hit Done to close this node.
5. Wire the 'Inject node' to the 'Function node' and the function node to the 'Debug node'. Most nodes have a grey circle on their left side, which is their input port, and on their right side, which is their output port. Left clicking and dragging the output to the input port of the next node connects the two together.
6. Finally press 'Deploy'.
7. Now click the button on the left-hand side of the Hello injection node – you should see the output in the debug tab on the right-hand side. Congratulations, you made your first Node-RED flow!

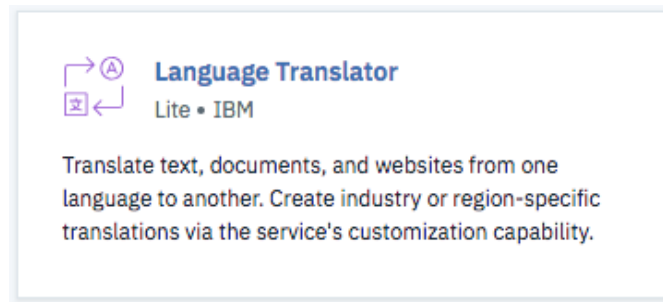
This flow has created a payload containing the string “Hello”, then added the string “ World” to the payload, then output the resultant Payload to the debug output.



STEP 5 – USING WATSON SERVICES FOR FURTHER ANALYTICS

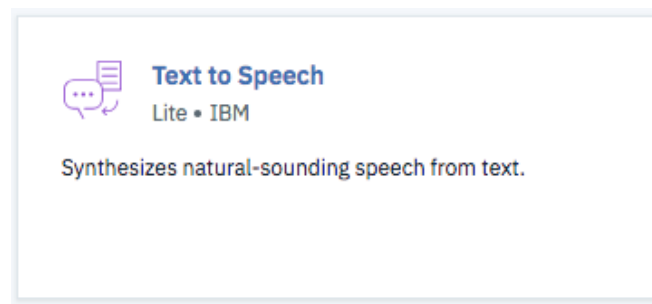
In this section, you'll create and bind some Watson services to your Node-RED instance, for language translation and text to speech. This action creates new credentials that allow you to use the service. By connecting to your Node-RED instance, these credentials are available for your application to use.

1. Go to the Resource list (see previous instructions if you can't remember how to get there) and click on Create Resource.
2. From the left-hand menu, choose AI and then Language Translator.



3. Leave the options as default and choose "Create"

Go through steps 1-3 above, but also for the Text to Speech Watson Service.



Finally, you need to connect both services to your Node-RED instance.

From your Resource List, open the detailed view of your Node-RED application by clicking on the name.

On the Overview page, click on Create Connection. From the next page, you need to connect **both** your new Watson services to your Node-RED instance.

10 ▾

Items per page | 1-4 of 4 items

1 of 1 pages < 1 >

SERVICES	RESOURCE GROUP	PLAN	SERVICE OFFERING
cloud-object-storage-eq	Default	Lite	Cloud Object Storage
Language Translator-i1	--	Lite	Language Translator

Connect

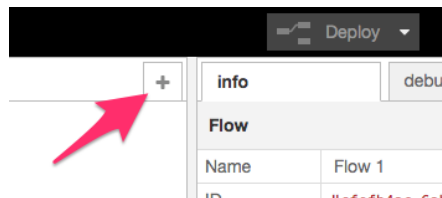
Firstly, choose Connect for the Language Translator service. Leave the options as default in the "Connect IAM-Enabled Service" window. When you're prompted to restage (which is equivalent to restarting), click Cancel because you want to connect to both services before you restage.

Next choose to Connect for the Text to Speech service via the Create Connection button. This time choose to restart when both services are connected.

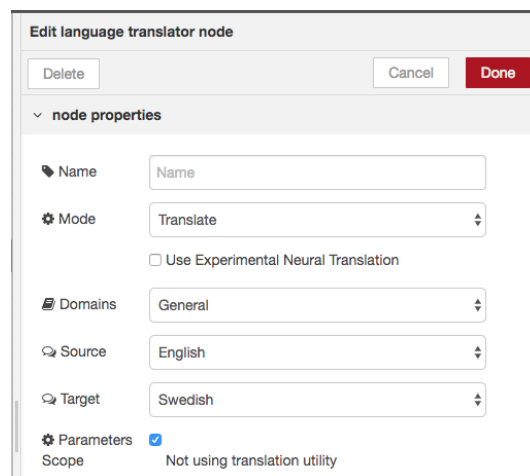
Once your Node-RED application has restarted, re-open the editor using the URL from your dashboard.

We will now create a Node-RED flow that takes some input text, translates it to another language and then converts the output text to speech.

1. Create a new Flow using the Plus symbol at the top right of the main canvas.



2. Select an input inject node and drag it onto the new canvas. Double-click to open and for the Payload field, select string. Enter a string, such as *Hello, this is my first Node-RED application.* In the Name field, enter a name for this node, such as *Hello World inject.* Click Done.
3. In the filter nodes search field, enter “translator” to find the language translator node. Drag the node onto the canvas. You can move the nodes to make more space if required.
4. Double-click the language translator node. Select to translate from English to another language of your choice. Select the General domain. Click Done to save your changes.



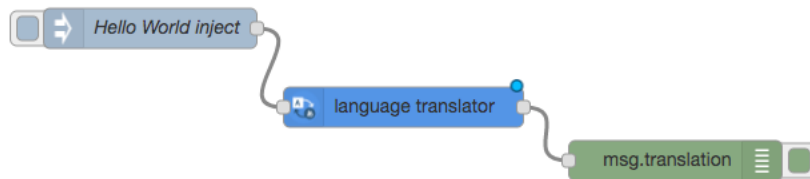
5. Select the language translator node and click the info tab. Notice that the node puts its translated text into msg.payload, but also a full response in msg.translation. msg is a reserved object that Node-RED uses to allow individual nodes to communicate with each other. Think of msg as an envelope into which one node places information that allows another node to read it. The language translator node is expecting to find a payload that is already in the msg envelope, and it will insert a translation into the msg envelope.
6. Select an output debug node and drag it onto the canvas. Open the debug node and change the output to msg.translation. Enter the word translation after msg.

Output

to

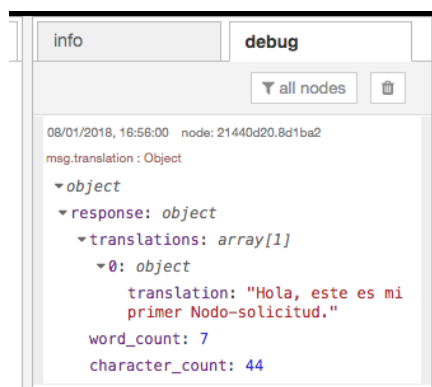
Name

- Click Done to save your changes.
- Link, or wire, the nodes together by clicking and dragging your cursor from one node to the other.

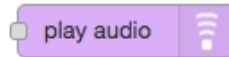


- The blue circles indicate that your flow has unsaved changes, which means that the application needs to be deployed. Click Deploy to deploy and save your changes.
- To initiate the flow, click the tab linked to the inject node. You now see the output on the debug tab.
- The application is translating the text that you entered in the Payload field of the inject node. You can view the translated text as part of the output object shown in the debug tab. Click on the arrow on the left of the titles to expand each section to show the various details included in this object, these include the translation, word count and character count.

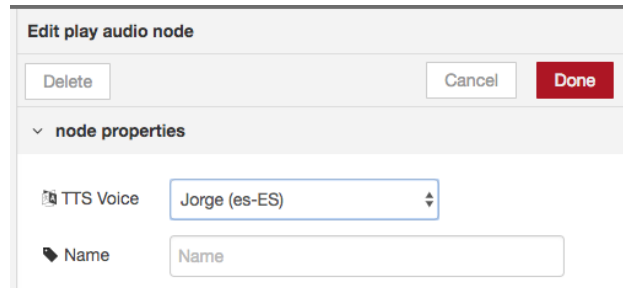
This kind of detailed output object starts to show the potential power of this toolset. For example, this data could be used for further detailed analysis, statistics or as part of a dashboard display.



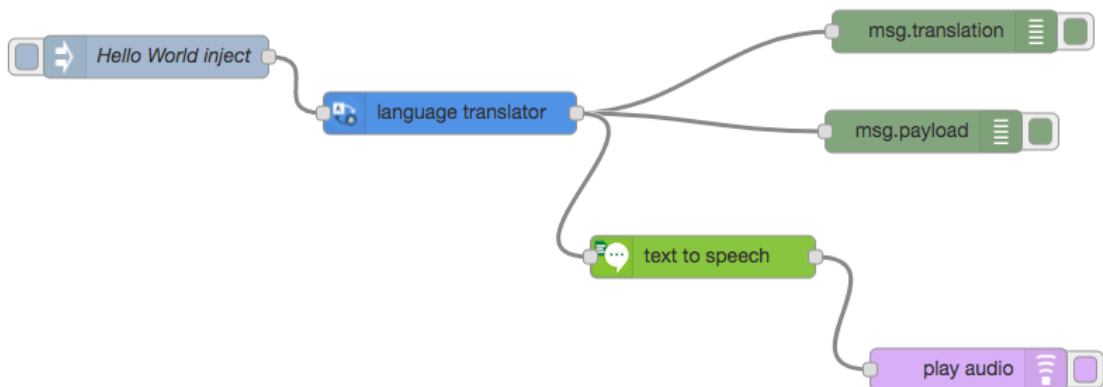
- Luckily for us, the translated text is also output as-is on the msg.payload. You can check for yourself by adding another debug node to see the msg.payload output as well. Deploy and re-run and you'll see both debug outputs in the debug tab.



17. If you double click the “play audio” node you will see the option to choose a TTS (Text to Speech) Voice for use when the payload is a string. Since we are using a .wav file as payload, you don’t need to change this option.



18. Finally, we need to wire all these nodes together and redeploy our flow. Create a link which takes the output from the translator node to the text to speech node. And then a link from this node onto the audio output node. Re-deploy your flow and then reinitiate your flow to hear the translation spoken out loud.



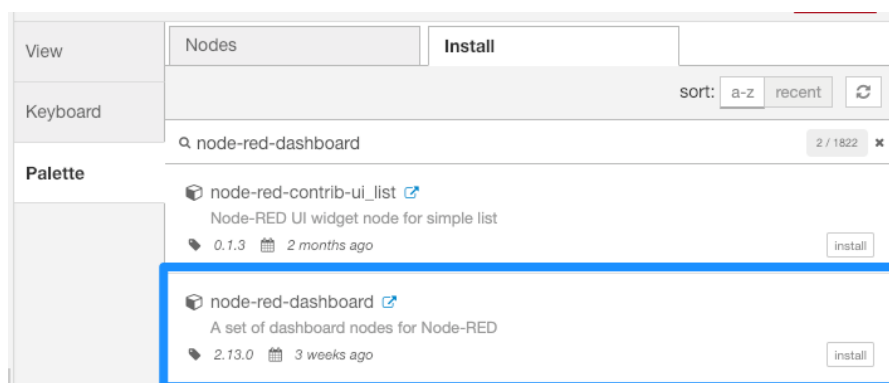
This is a very simple example. However, imagine the potential applications of this kind of technology, for example automated translation, transcription (using the other Watson service for speech to text), chatbots and many others. Each one using capabilities provided by an existing service which you can tap into as required.

STEP 7 – CREATING A DASHBOARD – PREREQUISITES

Now, getting started on our tweet dashboard.

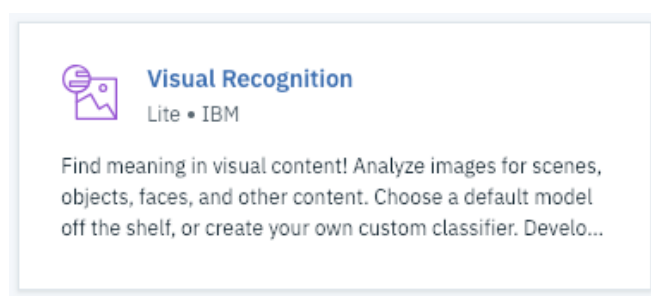
As we have seen in previous steps, one of the great things about Node-RED is that there are additional nodes and flows available to access via an online library at <https://flows.nodered.org/>. We want to add one of these nodes to allow us to create dashboard.

Log into your Node-RED editor, select the three-bar side menu in the top right hand corner and select "Manage palette". Then choose Palette > Install on the tabs and search for "node-red-dashboard" and install the node with this name.



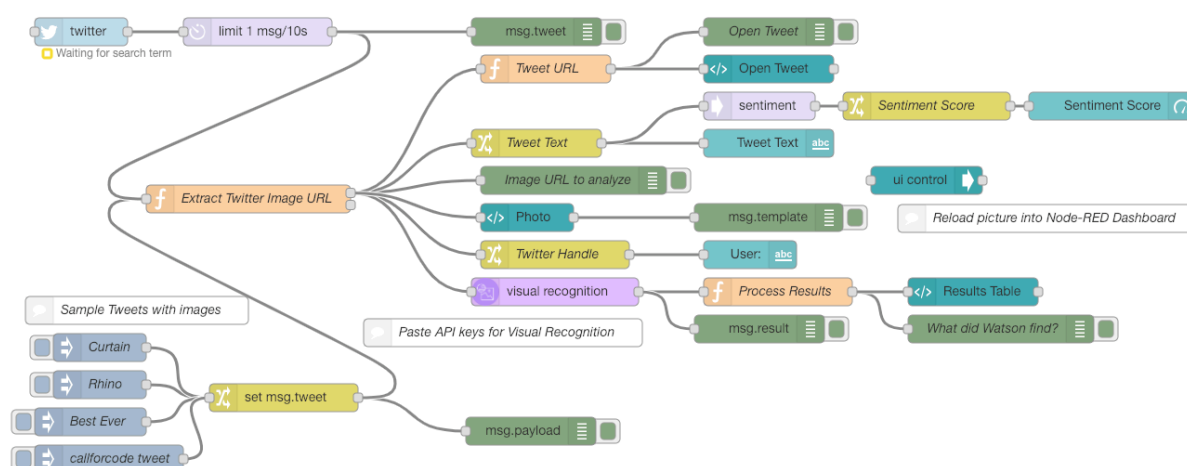
Click Close to close the palette manager and get back to the main editor.

We also plan to make use of another Watson API service for this dashboard. This time we'll add the Watson Visual recognition API. Go through the same steps as used previously in [Step 5](#) on page 8 of this document. Ensure that you connect this new service to your Node-RED instance and that you restage once the connection has been made.



STEP 8 – CREATING THE TWITTER DASHBOARD

You should ensure you look through the flow in detail to understand the overall process and the steps taken to get to a final dashboard.




- It takes input from two places: via the Twitter node in the top left, or via some customised “injection” nodes in the bottom left corner for testing.
- Both inputs go to the “Extract Twitter Image URL” node which takes the payload and pulls out key elements for processing.
- These elements are passed to a number of nodes which create the various content required for the final dashboard. These include the following:
 - URL of the tweet
 - Text from the tweet – which is given a sentiment score via the Watson service (how positive or negative is it?).
 - The URL of the picture contained in the tweet – which is passed to the Watson visual recognition service. This returns a table of things potentially found in the image, with a confidence score.
 - The handle of the person who tweeted
- Finally the dashboard elements are updated with the relevant information for the tweet.

Twitter Image Analysis

Tweet

User: Bodhihodi

RT @Jasper_Collie: I have fallen asleep watching the snow falling outside. Hopefully my Dad will let me play out in it again later. #uksnow...



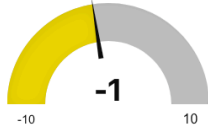
Results

Watson thinks this picture contains a ash grey color dog.

Class	Confidence
dog	0.898
canine	0.768
carnivore	0.768
mammal	0.769
animal	0.902
English setter dog	0.524
domestic animal	0.529
Animal Tending	0.794
ash grey color	0.988

[Open Tweet](#)

Sentiment Score

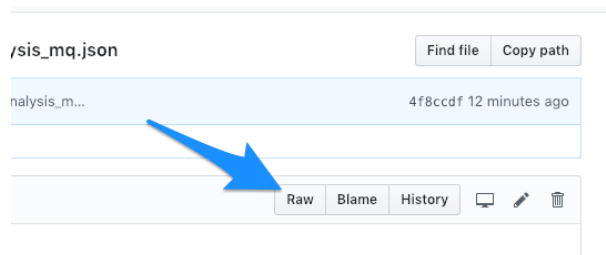


Let's import the flow and get it running. Use the following steps:

1. Open a new tab in your browser and navigate to the following link:

https://github.com/mandieq/node-red-exeter/blob/master/flows/twitter_food_analysis_mq.json

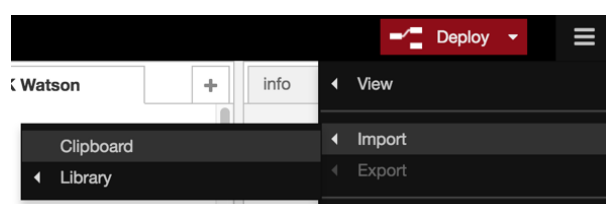
Download the JSON file which is shown, by right clicking on "Raw" and choosing "Save Link As..."



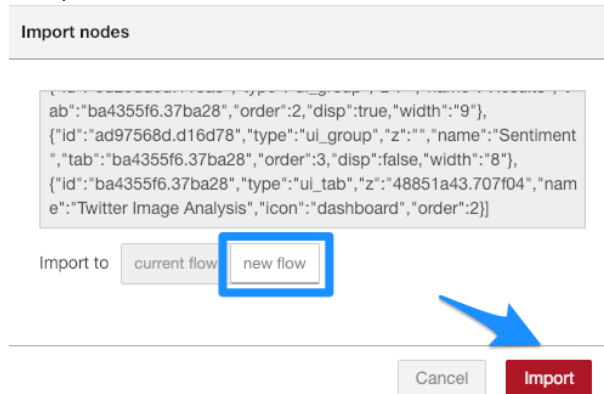
2. Open the downloaded JSON file with a text browser. Select all the text in the file, ensuring that you get the square brackets at the beginning and the end of the file. Copy this text so it is in your clipboard. You are now ready to import the flow into the flow editor.
3. Click the flow editor Menu.



4. Click Import > Clipboard.



5. Paste your flow (the text you copied from the JSON file) into the text box. Choose to import to “new flow”, then, click Import.



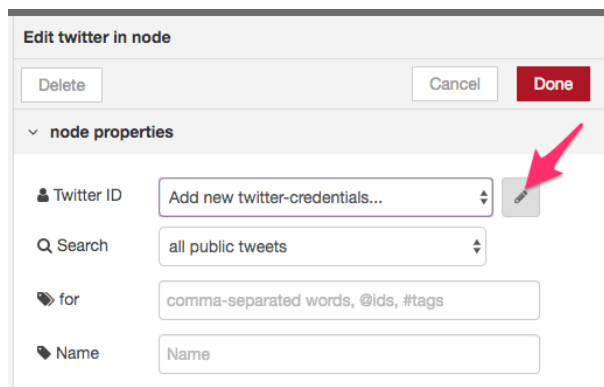
Import nodes

```
{  
  "id": "ba4355f6.37ba28", "order": 2, "disp": true, "width": "9",  
  {"id": "ad97568d.d16d78", "type": "ui_group", "z": "", "name": "Sentiment",  
    "tab": "ba4355f6.37ba28", "order": 3, "disp": false, "width": "8"},  
  {"id": "ba4355f6.37ba28", "type": "ui_tab", "z": "48851a43.707f04", "name": "Twitter Image Analysis", "icon": "dashboard", "order": 2}]
```

Import to

You can then use the Zoom buttons to show the whole flow which has been created.

6. Finally you need to set up your Twitter ID in the “twitter” node in the workflow. If you don’t have a Twitter account, a new one can be easily created via the main Twitter website: <https://twitter.com/>



Edit twitter in node

node properties

Twitter ID

Search

for

Name

Go through the steps required for this node. Unfortunately, Twitter has changed recently to require the creation of a developer account to access the Twitter APIs! I created a personal account using a twitter ID I created for this use. There is a section in this process which requires you to answer certain questions about your planned use of the service, feel free to use the following text to save you having to write your own 300 characters worth. ☺

1. I’m using Twitter’s APIs to use with MBA students for data analytics.
2. I plan to analyse Tweets filtered by a specific topic to perform sentiment analysis
3. Use case does not involve any of the above.
4. Tweets will be displayed on a map showing how positive or negative a tweet is on a particular topic.

You then need to create an app with this developer account. Go to the following link and follow the steps required: <https://developer.twitter.com/en/apps>

Once this is created (note that you don’t need to fill in all the boxes), you can then access the keys and tokens required to use in your Twitter node!

Note the limitations of this node which are shown in the info tab when you click on it. Choose your search term wisely (i.e. something active enough to populate your map as you watch it, but perhaps not so busy as to completely swamp it) and redeploy your flow selectively (so as to not exceed the API deploy limit).

Note: this node is subject to the rate limiting restrictions of the Twitter API. It polls the API once a minute for updates. If you deploy frequently you may exceed the limit. The node will automatically delay polling until the end of the current rate limiting window.

7. Now deploy your flow. You can access the dashboard via <https://<your Node-RED URL>/ui> or via the relevant button on the Dashboard tab in the top right corner.



Note that I had issues with the picture not showing using Chrome on a Mac. It worked with Firefox.

THINGS TO NOTE / TRY

- You can watch your tweets arriving and being processed in the debug tab – check that you are actually finding something with the search term you’ve chosen. An alternative approach is to monitor a specific user / set of users and seed tweets yourself.
- Take this flow as a base and personalise. Change the dashboard set up / change the inputs.

STEP 9 - OTHER FLOWS TO EXPLORE

There are lots of existing flows available which you might want to explore if you have time. I'd recommend looking at the Twitter Nutrition Analysis flow from John Walicki's material on github:

<https://github.com/johnwalicki/Node-RED-Twitter-Workshop>

This dashboard takes a tweet, analyses the picture and then outputs the nutritional analysis of that food, along with a warning if it is high in sugar or fat.

The screenshot shows a web browser displaying the 'Twitter Nutrition Analysis' dashboard. The dashboard is divided into three main sections: Tweet, Results, and Nutrition.

Tweet Section: Shows a tweet from a user with a blurred profile picture. The tweet text is 'Breakfast time! https://t.co/vWLMXmbUHJ'. Below the text is a photograph of a hard-boiled egg on a white plate, with a spoon and a small bowl of egg yolk.

Results Section: Displays the analysis results. It states 'Watson thinks this picture contains boiled egg.' Below this is a table with two columns: 'Class' and 'Confidence'.

Class	Confidence
boiled egg	1
egg	0.5

Nutrition Section: Displays the nutritional information for 'HARD BOILED EGGS, UPC: 812324023081'. It includes a table of nutritional information and a warning.

Nutritional Information	Value
Energy	155kcal
Protein	12.62g
Total lipid (fat)	10.68g
Carbohydrate, by difference	0.97g
Fiber, total dietary	0.0g
Sugars, total	0.97g
Calcium, Ca	58mg
Iron, Fe	1.40mg
Sodium, Na	121mg
Vitamin C, total ascorbic acid	0.0mg
Vitamin A, IU	775IU

Warning: High in Fat

There are a number of other dashboards in this repository which you might want to try, or alternatively use as a base for a personal project.

Also consider browsing the flows available in the Node-RED library at: <https://flows.nodered.org/>

USEFUL LINKS / FURTHER INFORMATION

If you've finished early or want to explore further after this session, there are many other things you might want to try. See the following links for further details.

Watson starter kits <https://cloud.ibm.com/developer/watson/starter-kits>, with demos you can try from a web front end, including:

- Watson Assistant Basic
- Watson Speech to Text Basic
- Watson Text to Speech Basic
- Watson Natural Language Understanding Basic
- Watson News Intelligence
- Watson Visual Recognition Basic

developerWorks course, Node-RED: Basics to bots: <https://developer.ibm.com/courses/all/node-red-basics-bots/>

Watson Developer Cloud on Github: <https://github.com/watson-developer-cloud>, containing 3 key repositories for Node-RED work:

- node-red-node-watson – collection of nodes for the IBM Watson services
- node-red-bluemix-starter – Deploy to IBM Cloud enabled instance of Node-RED that can be forked, customised and reused
- node-red-labs – Node-RED labs including:
 - Introduction to Node-RED
 - Basic Labs using Watson nodes - show how to invoke nodes from a simple flow
 - Advanced Labs using multiple Watson nodes - show how to extend or combine where different nodes and services
 - Watson Contribution Nodes - how to add watson developer cloud contribution nodes to Node-RED
 - New Node-RED Starter Kits - prebuilt applications from which to start your own prototypes

Watson Developer Cloud main website: <https://www.ibm.com/watson/developer/> has all the formal materials for the Watson APIs, kits and tooling including:

- Documentation: <https://cloud.ibm.com/developer/watson/documentation>
- Starter kits and App Gallery: <https://www.ibm.com/watson/developercloud/starter-kits.html>
- Watson "What's New" blogs: <https://developer.ibm.com/watson/blog/>
- Watson Services Catalog: <https://www.ibm.com/watson/products-services/>
- Other Resources: <https://www.ibm.com/us-en/marketplace/cognitive-application-development/resources>

Watson Academy: <https://www.watson-academy.info/>