Visual Recognition Hands-on Lab using Node Red

Introduction

IBM Watson offers a broad set of services to enable developers to build cognitive solutions. Cognitive Computing Systems are defined as systems that **understand**, **reason**, and **learn** to assist humans in making better decisions by penetrating the complexity of Big Data.

In addition to text and speech, humans communicate using images and as such, it is critical for any cognitive system to be able to understand images and extract relevant information. To help with this task, IBM Watson offers Alchemy Vision and Visual Recognition services which employ deep learning technology to understand images' content and context.

Alchemy Vision API can analyze an image and return information about the objects, people, and text found within that image. Alchemy Vision leverages a pre-trained set of classifiers which it uses to extract and identify objects in an image. Alchemy Vision also does face detection, returns the gender and age range for the face in the image and if it is the face of a celebrity, returns the name of the celebrity as well.

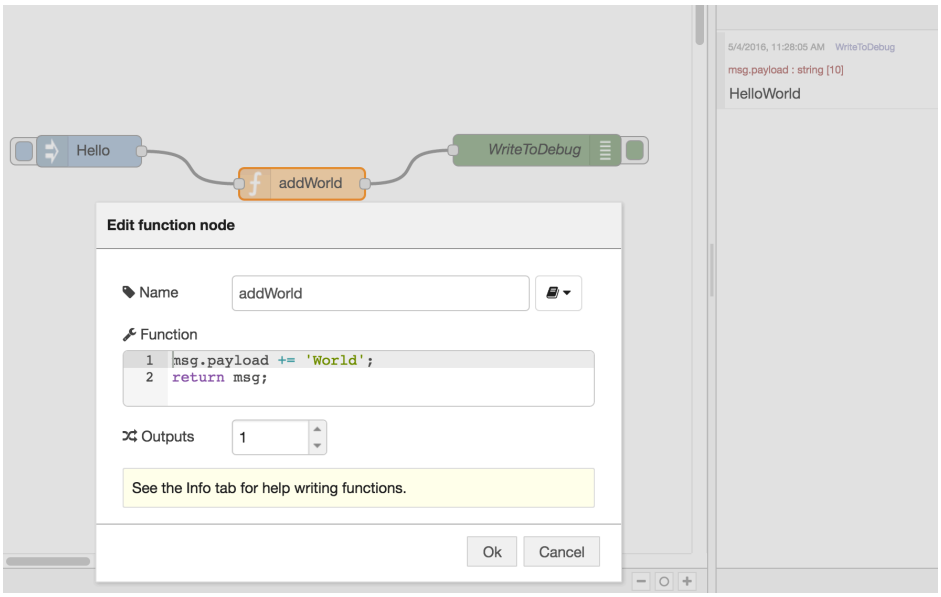
Visual Recognition adds to the Alchemy Vision capabilities by offering the ability to train custom classifiers. This is extremely valuable when trying to identify and extract unique information from an image that best suit your needs.

In this lab, we will step through the process of building an app using Node-RED to identify objects in an image using Alchemy Vision. We will also build custom classifiers using Visual Recognition service and leverage that in an app.

Node-RED

Node-RED is a visual tool designed by IBM Emerging Technology team with the objective of simplifying the task of connecting today's world of computers, devices, and online services (APIs). Node-RED is an open source tool based on Node.js which was originally designed for wiring the Internet of Things, but has since evolved to support various nodes that provide expanded functionality.

The following image shows a quick preview of how you'd write a Hello World program using Node-RED. This simple flow starts with an "inject" node that injects a string ("Hello"), then adds a "function" node which adds a string to msg.payload ("World) and sends the resulting payload to a "debug" node which prints out the results to the Debug tab in Node-RED.



In this lab we will leverage Node-RED to build flows using Watson cognitive services.

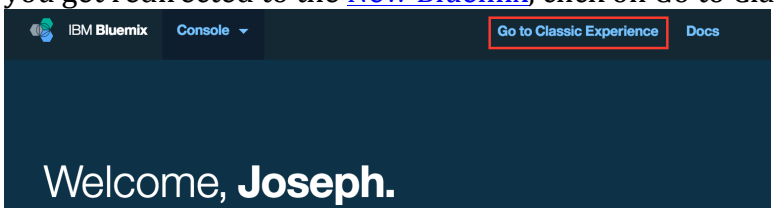
Creating Services on Bluemix

For this lab, we will need to get credentials for Alchemy Vision and Visual Recognition services.

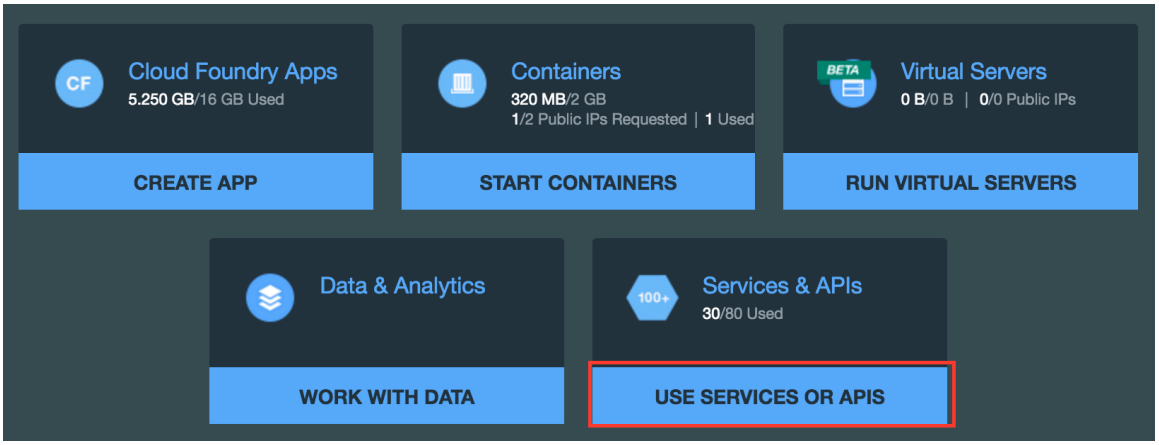
To leverage any of the Watson Developer Cloud services (including Alchemy Vision and Visual Recognition), you need to create a service instance in Bluemix which, upon success, returns credentials that you can use in accessing the service.

Alchemy Vision Service

1- Log in to Bluemix using your username and password.
➔ The instructions below apply for the Classic Experience view of Bluemix. If you get redirected to the New Bluemix, click on Go to Classic Experience link.



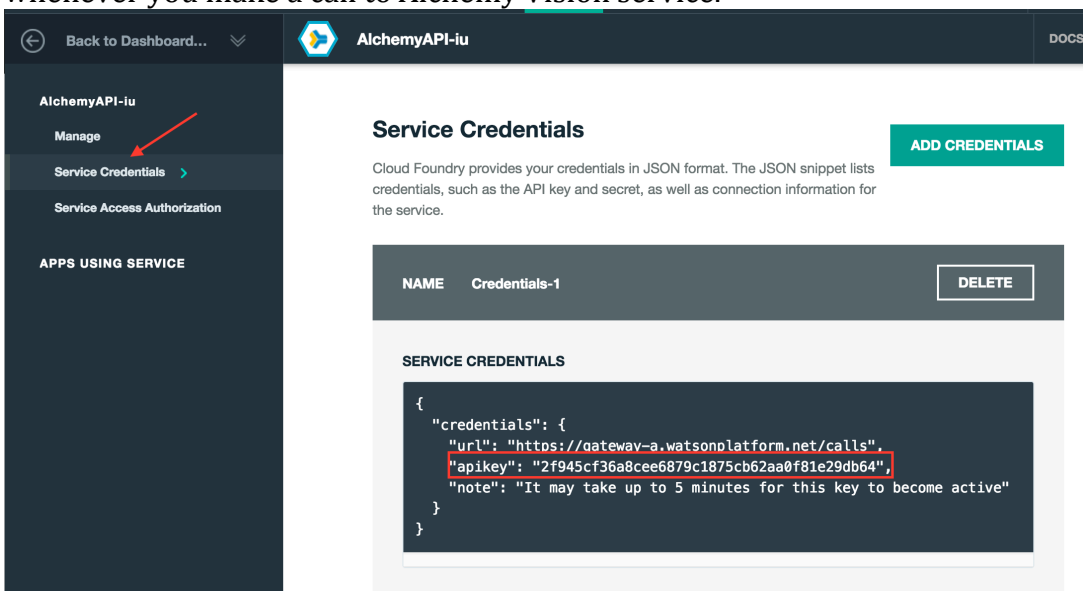
2- From your dashboard, click on USE SERVICES OR APIS.



3- From the catalog of Watson services, select AlchemyAPI service.

4- Accept all the default fields and hit CREATE.
Space: dev
App: Leave unbound
Service name: AlchemyAPI-iu (by default the system creates a unique service name)
Credential name: Credentials-1 (by default, the system creates a unique credential name).
Selected Plan: free
➔ When complete, an AlchemyAPI service is created.
NOTE that if you had already created a free AlchemyAPI service, you will get an error (no need to create a new service, you can use you existing credentials):
Service broker error:
{"description"=>"Only one free key is allowed per account in a 24-hour period."}

5- Click the Service Credentials in the left navigation bar and capture the ***apikey*** which we will need in subsequent steps in the lab. This ***apikey*** is needed whenever you make a call to Alchemy Vision service.



After obtaining the AlchemyAPI credentials, run the following commands using curl (or your preferred REST client) to get an idea

apikey=YOUR_API_KEY ➔ your AlchemyAPI credentials
url=http://www.les-felins.com/wp-content/uploads/sites/791/2015/06/leopards_transparent.png

Image Tagging (objects in image)

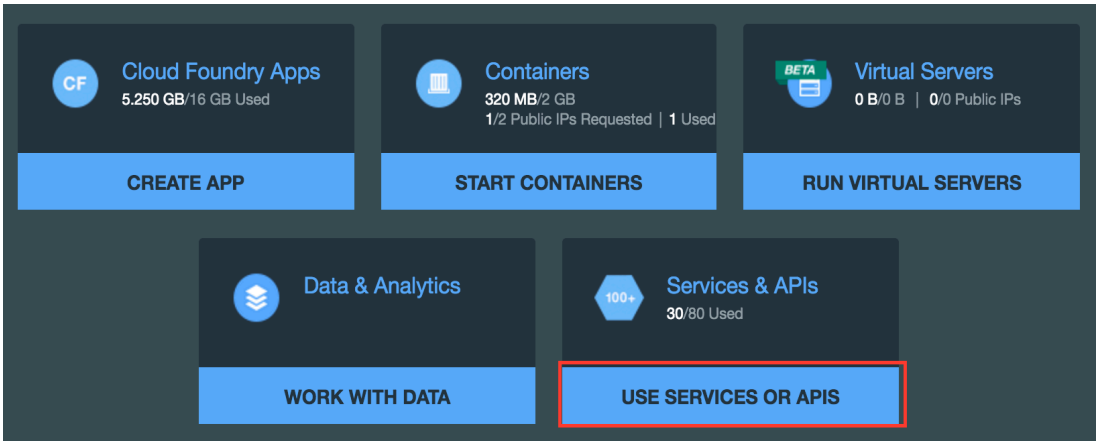
curl "http://gateway-a.watsonplatform.net/calls/url/URLGetRankedImageKeywords?apikey=${apikey}&url=${url}&outputMode=json"

➔ Review the response

Visual Recognition Service

Repeat the steps above to create a Visual Recognition service and capture the returned credentials.

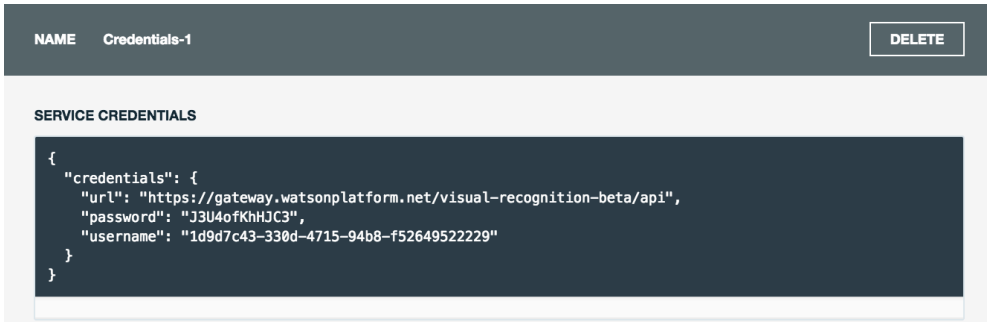
1- Log in to Bluemix using your username and password.
2- From your dashboard, click on USE SERVICES OR APIS.



3- From the catalog of Watson services, select Visual Recognition service.

4- Accept all the default fields and hit CREATE.
Space: dev
App: Leave unbound
Service name: Visual Recognition-kc (by default the system creates a unique service name)
Credential name: Credentials-1 (by default, the system creates a unique credential name).
Selected Plan: free
➔ When complete, a Visual Recognition service is created.

5- Click the Service Credentials in the left navigation bar and capture the ***username*** and ***password*** which we will need in subsequent steps in the lab.



Visual Recognition Custom Classifiers

After getting the visual recognition credentials, you will experiment with how to create and use custom classifiers. Visual Recognition service documentation includes details on how to create custom classifiers and what are the requirements for training data:
http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/visual-recognition/customizing.shtml

For training data, we will use the collection of leopard and tiger images referenced on visual recognition documentation.

1- Download these zip files to your local drive.

Tiger.zip: http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/visual-recognition/downloads/tiger.zip

Leopard.zip:
http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/visual-recognition/downloads/leopard.zip

2- Train leopard classifier

vrPosFile: Leopard.zip ➔ full path
vrNegFile: Tiger.zip ➔ full path
vrName: leopard

curl -v -X POST -u "***username*:*password***" -F "positive_examples=@${***vrPosFile***}" -F "negative_examples=@${***vrNegFile***}" -F "name=${***vrName***}" "https://gateway.watsonplatform.net/visual-recognition-beta/api/v2/classifiers?version=2015-12-02"

➔ capture the classifier id that gets returned

```
{"classifier_id":"leopard_669765398","name":"leopard"}
```

3- Train tiger classifier

vrPosFile: Tiger.zip ➔ full path
vrNegFile: Leopard.zip ➔ full path
vrName: tiger

curl -v -X POST -u "***username*:*password***" -F "positive_examples=@${***vrPosFile***}" -F "negative_examples=@${***vrNegFile***}" -F "name=${***vrName***}"

"https://gateway.watsonplatform.net/visual-recognition-beta/api/v2/classifiers?version=2015-12-02"

➔ capture the classifier id that gets returned

```
{"classifier_id":"tiger_1579683016","name":"tiger"},
```

4- Test classifiers using sample images:

-- download any image of a tiger or leopard to your local drive ➔ ***test.jpg***

-- Create a classifierlist.json file to include the classifier ids that were captured earlier:

```
{"classifier_ids": ["tiger_1579683016","leopard_669765398"]}
```

imgFile=test.jpg ➔ full path
classList=classifierlist.json ➔ full path

curl -v -X POST -u "***username***:***password***" -F "images_file=@${imgFile}" -F "classifier_ids=<${classList}" "https://gateway.watsonplatform.net/visual-recognition-beta/api/v2/classify?version=2015-12-02"

➔ review results of visual recognition classifiers, it should return the most relevant class and the confidence score

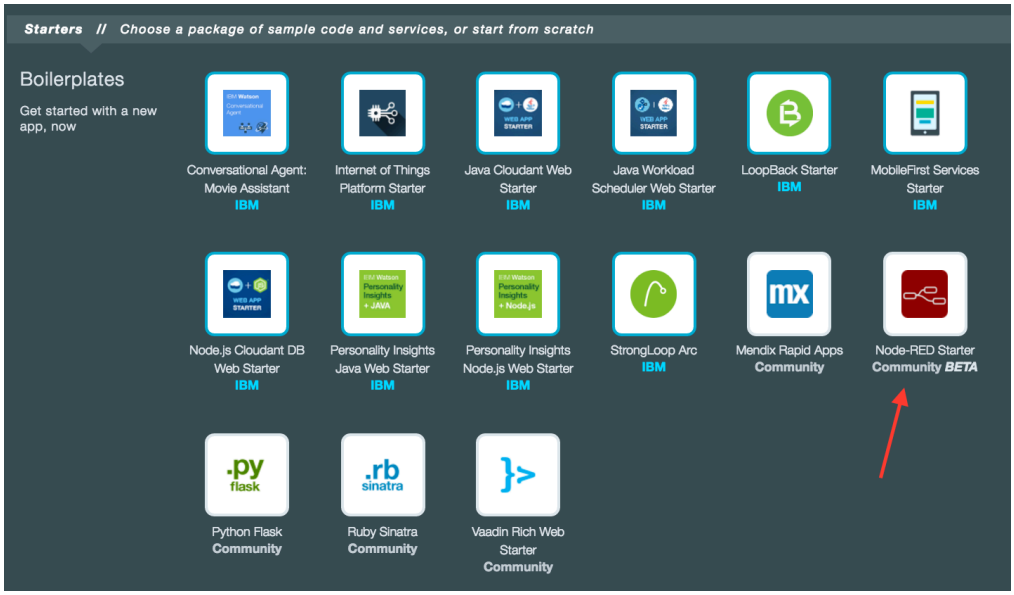
```
{"images":[
        {"image":"ihn-F2B0591EB9A71398E77E790A9A766244.jpg",
         "scores":[
                {"classifier_id":"tiger_1579683016","name":"tiger","score":0.760209}]}]
}
```

So far, we have created the required Watson services in Bluemix, obtained the credentials, and used those in simple curl commands to examine functionality. We also created two custom classifiers to differentiate between tigers and leopards. Next, we will explore how to leverage this data in Node-RED to build an app that would call the service and return visual identification results.

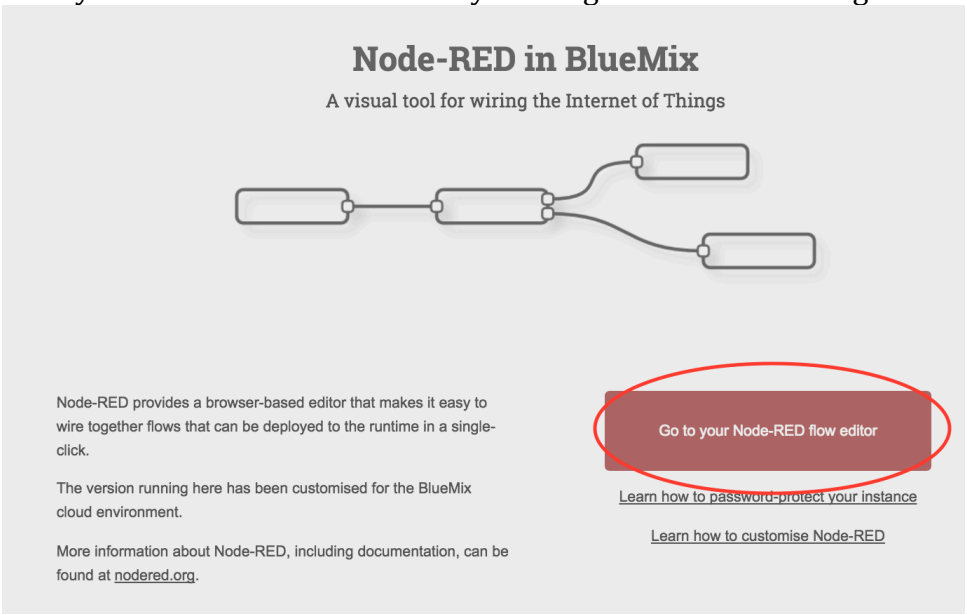
Node-RED App using Alchemy Vision

We will explore running Node-RED boilerplate from Bluemix as well as running Node-RED locally. To run it from Bluemix, follow these steps:

1- Log in to Bluemix using your username and password.
2- Go to Catalog tab and select Node-RED boilerplate.



3- Provide a unique app and host name and hit Create.
➔ When complete, it starts your app.
4- Point your browser to https://{appname}.mybluemix.net
5- Start your Node-RED flow editor by clicking on the RED rectangle.



6- Now that you have your Node-RED flow editor up, you can follow the steps outlined here.
➔ Double click the Image Analysis node and add your Alchemy ***apikey***.

Node-RED App using Visual Recognition

In this section, we will install Node-RED locally and create a new node for Visual Recognition. The Visual Recognition node that is available on the Bluemix Node-RED

boilerplate doesn't support passing classifier list as a parameter yet so we'll create a new node in our local environment and add the required input parameters.

- To start with install node-red on your laptop:

```
sudo npm install -g --unsafe-perm node-red
```

- Start node-red locally:

```
node-red
```

  ==> this starts a server on port 1880 by default
- Open your node-red flow editor by pointing your browser to http://localhost:1880
  ➔ Notice that the look and feel is slightly different from the node-red flow editor you started on Bluemix. Mainly, the nodes available are different between the two. This is a reflection of which nodes are enabled and supported in the general release vs. specific installs.

- Go ahead and kill your node-red session (CTRL-C in the terminal window where you started it).

To create a new node, we need to define it in the following structure so it can fit with packaging requirements for node-red:

```
- package.json
- watsonVR
        |-watsonVR.html
        |-watsonVR.js
        \-icons
                \-watsonVR.png
- README.md
- LICENSE
```

- Download sample code from github

```
git clone https://github.com/joe4k/node-red-watsonVR.git
```

  ➔ Review the code in watsonVR.js and watsonVR.html

- Link newly created package for watsonVR node

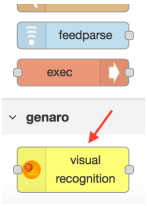
```
sudo npm link
```

➔ This makes the newly created node accessible from your local node-red install.

- Start node-red again

```
node-red
```

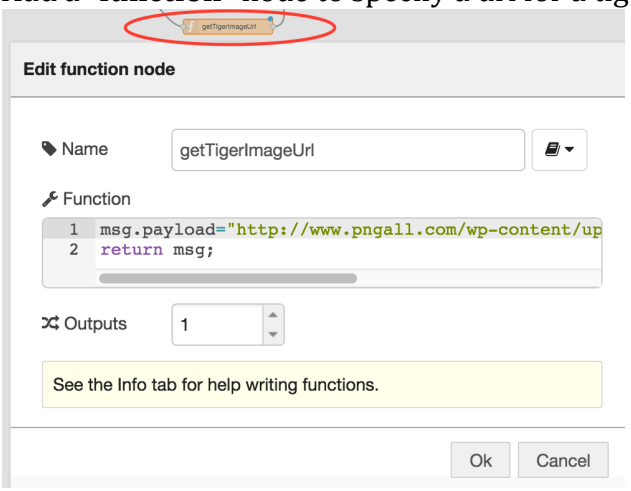
- Point your browswer to http://localhost:1880
  ➔ Notice in your flow editor the newly added node (visual recognition)



Now, we are ready to build a flow using Watson Visual Recognition service. We'll start with creating a simple test flow. Here are the steps:

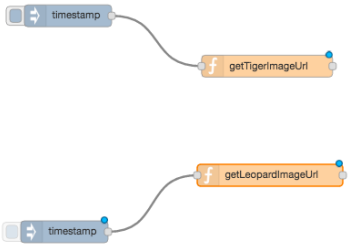
1- Add an "**inject**" node (this effectively starts the flow either with some initial data or with no data)

2- Add a "**function**" node to specify a url for a tiger image.

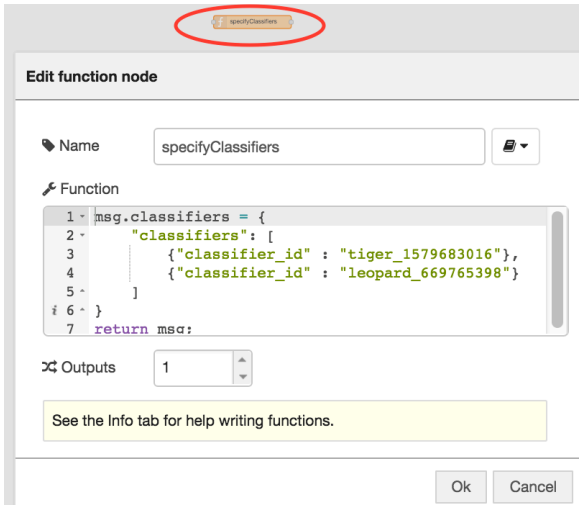



```
msg.payload="http://www.pngall.com/wp-content/uploads/2016/03/Tiger-Free-Download-PNG.png";
return msg;
```

3- Repeat steps 1 and 2 to insert another "**inject**" node and another "**function**" node but this time specify a url for a leopard image. The flow should look as follows. This allows us to pass either a tiger image or a leopard image.

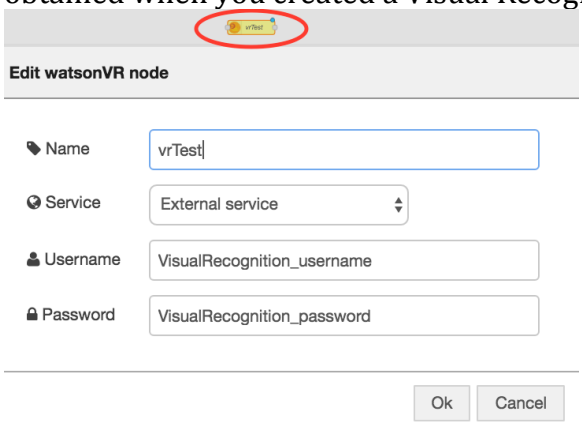


4- Add a "**function**" node to specify which classifiers to use. Note that these classifiers should have been created as described earlier in the Visual Recognition Custom Classifiers section, step 3.



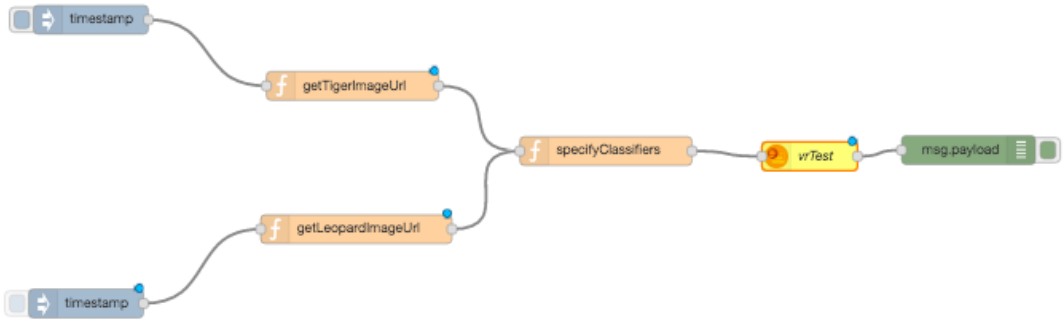

```
msg.classifiers = {
   "classifiers": [
      {"classifier_id" : "tiger_1579683016"},
      {"classifier_id" : "leopard_669765398"}
   ]
}
return msg;
```

5- Add "**visual-recognition**" node to pass the image url to Watson Visual Recognition service. For Username and Password, specify the credentials you obtained when you created a Visual Recognition service in Bluemix.



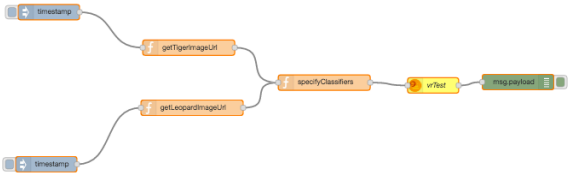
6- Add a "**debug**" node to report results to Debug tab.

7- Lastly connect the nodes together to get the following flow.

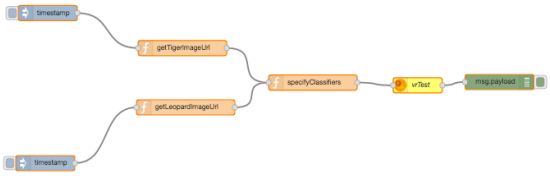


8- Deploy the flow.

9- Hit the top "**inject**" node (press the little rectangle to the left of the node) and note the output in the Debug tab.



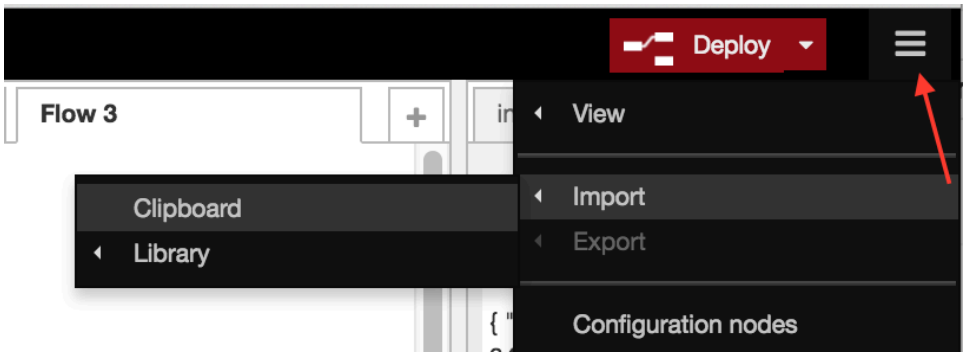
10- Try again with the bottom inject node.



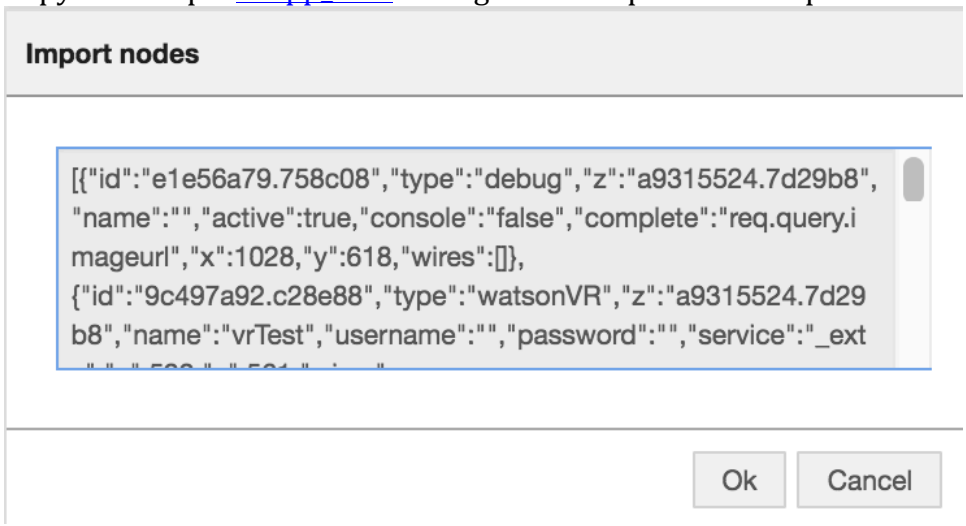
Now that we've tested our flow is functioning as expected with our added Watson Visual Recognition node, we can build an app using that. To do so, we will import a flow from github repository, edit it and test it.

1- In your Node-RED flow editor, start a new flow by hitting the + sign in the Flows window (middle one, top right).

2- Import this sample flow by clicking on the Menu→Import→Clipboard



3- Copy the sample [vrApp_flow] from github and paste into Clipboard.



4- This imports the flow into your editor.
5- Double click "**vrTest**" node to edit and add your credentials.
6- Edit the specifyClassifiers node to specify which classifiers you'd like to use for visual recognition.
7- Deploy the flow.
8- Point your browser to http://localhost:1880/vrapp
9- Provide image URL and hit Analyze.
➔ Review the results.