Node-RED and Watson Visual Recognition

Hands-on Workshop

Urs Witzig
Yamini Rao

Overview

The IBM Watson Developer Cloud (WDC) offers a variety of services for developing cognitive applications. Each Watson service provides a Representational State Transfer (REST) Application Programming Interface (API) for interacting with the service. Some services, such as the Speech to Text service, provide additional interfaces.

This workshop will use the Visual Recognition service to show the integration into Watson Cloud services.

Node-RED is a visual tool for wiring the Internet of Things. It is easy to connect devices, data and API's (services). It can also be used for other types of applications to quickly assemble flows of services. Node-RED is available as open source and has been implemented by the IBM Emerging Technology organization. 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 IBM Cloud 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. You can also use it for example to implement simple REST APIs. You can find many other sample flows on the Node-RED website.

This app in this lab will be created and run on your IBM Cloud Account.

- In a first step, a IBM Cloud Node-RED environment will be created and setup. This will then host your Node-RED flows.

- Next you will create a simple Node-RED flow that allows you to past an Image URL from the Web and pass it to the IBM Watson Visual Recognition service you have created in an earlier lab.

Objectives

- Learn how to use the Node-Red environment on IBM Cloud

- Learn how to implement the Image-Analysis application using Node-Red

Prerequisites

- The prerequisite for this lab is an IBM Cloud account and a Watson Visual Recognition service

- Steps to create a Visual Recognition Service in the IBM Cloud console and retrieve the key from there (Service Credentials) is included below.

Section 1: *Creating services in IBM Cloud*

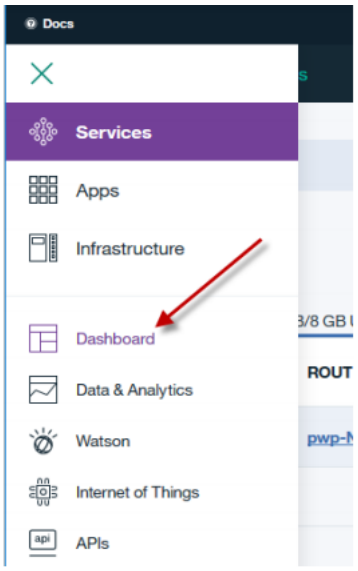
IBM Cloud offers services, or cloud extensions, that provide additional functionality that is ready to use by your application's running code.

You have two options for working with applications and services in IBM Cloud. You can use the IBM Cloud web user interface or the Cloud Foundry command-line in- terface (cf cli).

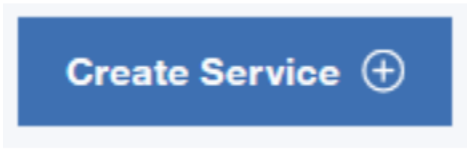
Step 1: **Create a Visual Recognition service** (On the IBM Cloud Console)

a) Open a Browser and navigate to your IBM Cloud Interface

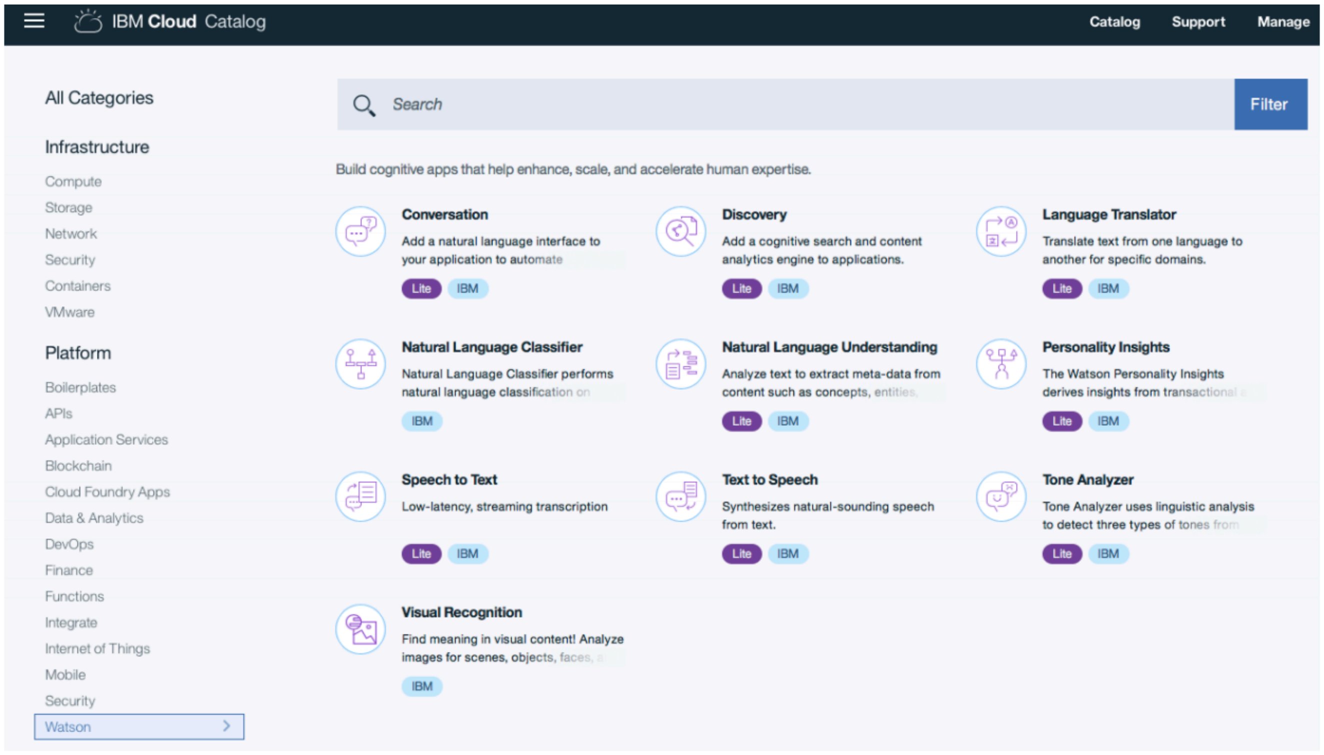
b) Click the menu icon and select Services and then Dashboard.



c) Click the Create Service button



d) Under Services select the category Watson



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

Conversation — Add a natural language interface to your application to automate — Lite IBM

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

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

Natural Language Classifier — Natural Language Classifier performs natural language classification on — IBM

Natural Language Understanding — Analyze text to extract meta-data from content such as concepts, entities, — Lite IBM

Personality Insights — The Watson Personality Insights derives insights from transactional — Lite IBM

Speech to Text — Low-latency, streaming transcription — Lite IBM

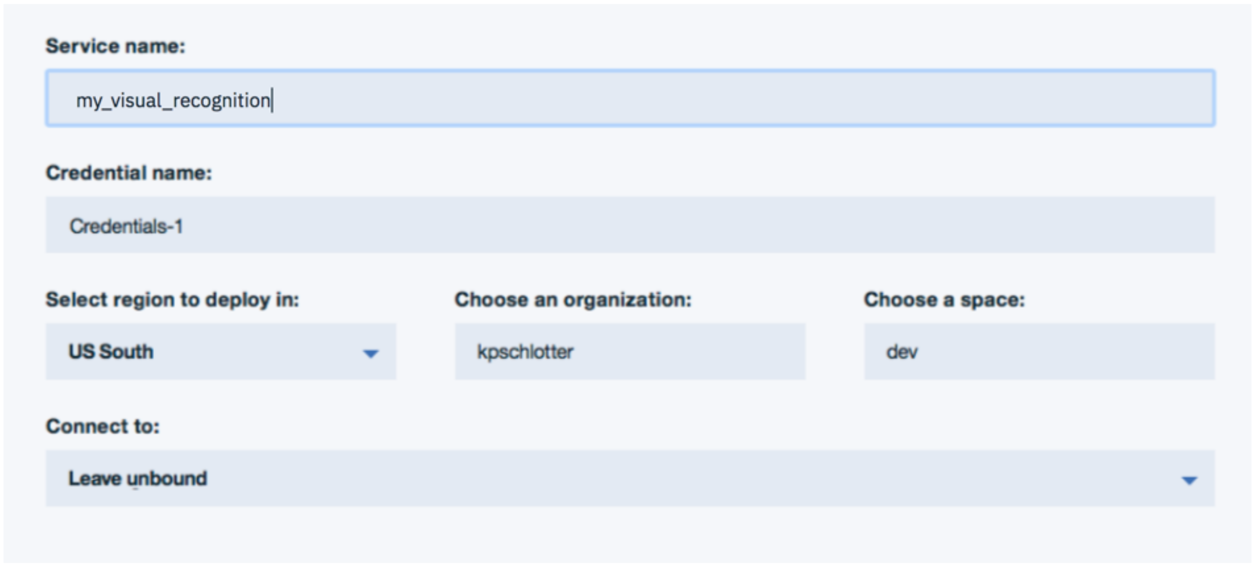
Text to Speech — Synthesizes natural-sounding speech from text. — Lite IBM

Tone Analyzer — Tone Analyzer uses linguistic analysis to detect three types of tones from — Lite IBM

Visual Recognition — Find meaning in visual content! Analyze images for scenes, objects, faces, — IBM

e) Click the **Visual Recognition** service and as *Service name* specify **my-visual-recognition**

You can leave the *Credentials name* as the default.



Service name:

my_visual_recognition

Credential name:

Credentials-1

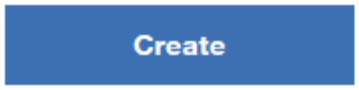
Select region to deploy in: US South

Choose an organization: kpschlotter

Choose a space: dev

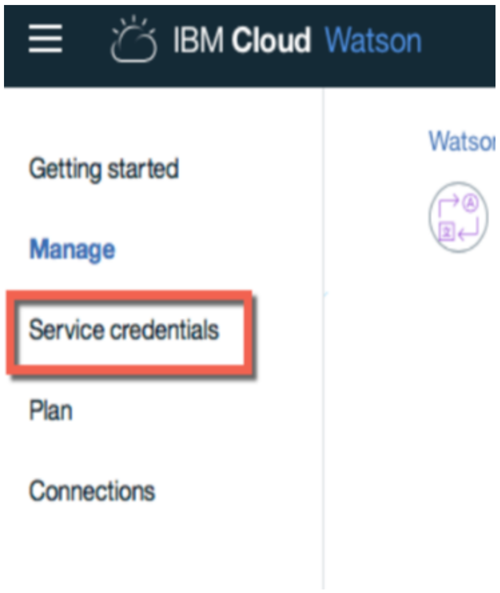
Connect to: Leave unbound

f) Click the **Create** button



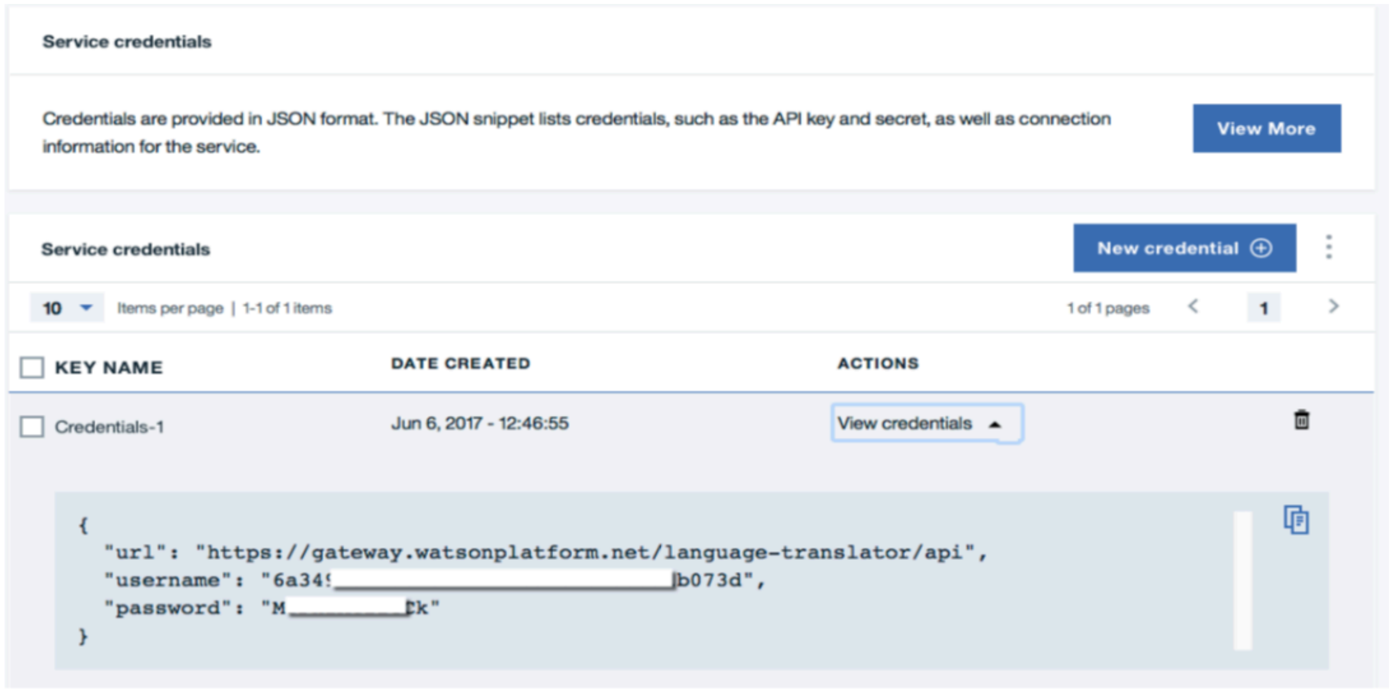
g) After the service is created you will see the **Manage** page of the service.

Click the **Service Credentials** to display them



IBM.

h) In the list of credentials select the Action **View credentials** to display the “Credentials-1”.



Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

Service credentials

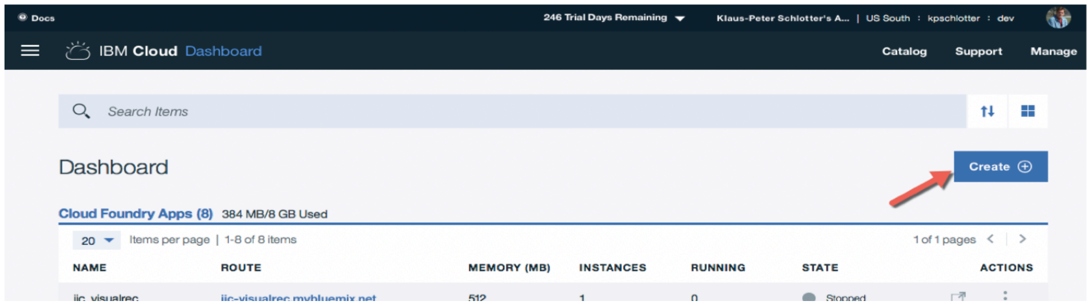
KEY NAME	DATE CREATED	ACTIONS
Credentials-1	Jun 6, 2017 - 12:46:55	View credentials

```
{
  "url": "https://gateway.watsonplatform.net/language-translator/api",
  "username": "6a34[redacted]b073d",
  "password": "M[redacted]tk"
}
```

i) Copy the username and the password of the service for later use into a file on your workstation or you can always go to the IBM Cloud console to copy the credentials from your list of services when you need them.

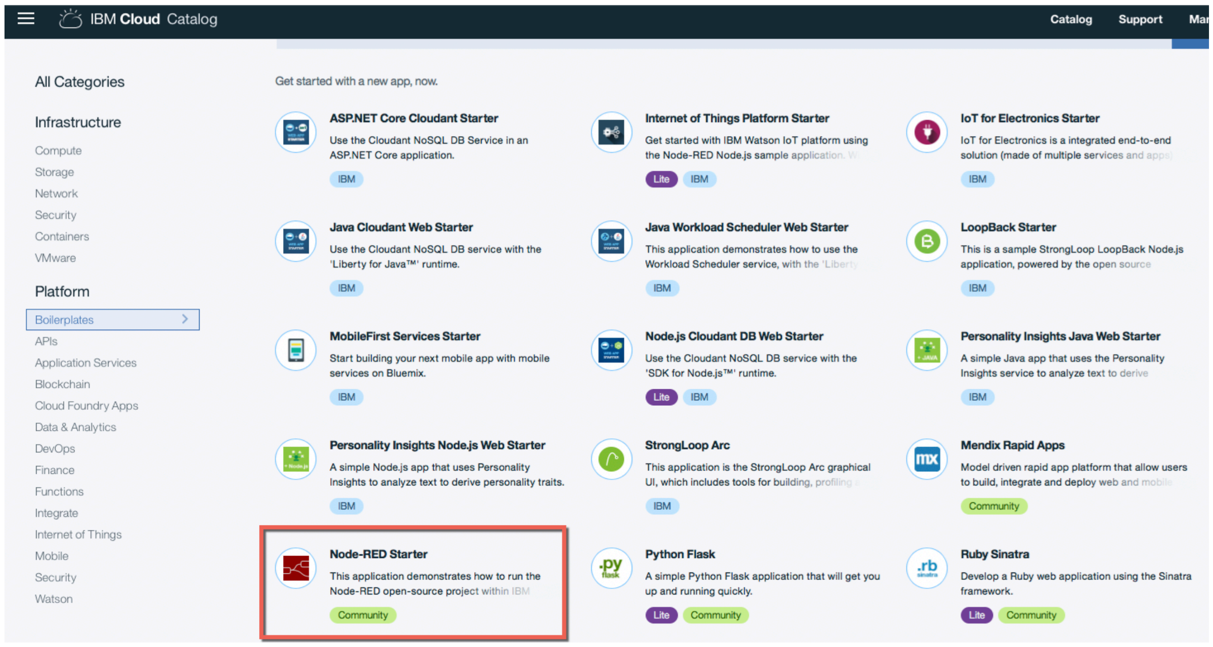
Section 2: Creating a Node-RED Visual Recognition Flow on IBM Cloud

Step 1 Log in to your IBM Cloud console and create a new application.


Docs
246 Trial Days Remaining
Klaus-Peter Schlotter's A... | US South : kpschlotter : dev
IBM Cloud Dashboard
Catalog
Support
Manage
Search Items
Dashboard
Create
Cloud Foundry Apps (8) 384 MB/8 GB Used
20 Items per page | 1-8 of 8 items
1 of 1 pages

NAME	ROUTE	MEMORY (MB)	INSTANCES	RUNNING	STATE	ACTIONS
iic_visualrec	iic-visualrec.mybluemix.net	512	1	0	Stopped	

Step 2 Under the Apps → Boilerplates category **click** the **Node-RED Starter**



Get started with a new app, now.

ASP.NET Core Cloudant Starter
Use the Cloudant NoSQL DB Service in an ASP.NET Core application.
IBM

Internet of Things Platform Starter
Get started with IBM Watson IoT platform using the Node-RED Node.js sample application. W
Lite IBM

IoT for Electronics Starter
IoT for Electronics is a integrated end-to-end solution (made of multiple services and apps
IBM

Java Cloudant Web Starter
Use the Cloudant NoSQL DB service with the 'Liberty for Java™' runtime.
IBM

Java Workload Scheduler Web Starter
This application demonstrates how to use the Workload Scheduler service, with the 'Liberty
IBM

LoopBack Starter
This is a sample StrongLoop LoopBack Node.js application, powered by the open source
IBM

MobileFirst Services Starter
Start building your next mobile app with mobile services on Bluemix.
IBM

Node.js Cloudant DB Web Starter
Use the Cloudant NoSQL DB service with the 'SDK for Node.js™' runtime.
Lite IBM

Personality Insights Java Web Starter
A simple Java app that uses the Personality Insights service to analyze text to derive
IBM

Personality Insights Node.js Web Starter
A simple Node.js app that uses Personality Insights to analyze text to derive personality traits.
IBM

StrongLoop Arc
This application is the StrongLoop Arc graphical UI, which includes tools for building, profiling
IBM

Mendix Rapid Apps
Model driven rapid app platform that allow users to build, integrate and deploy web and mobile
Community

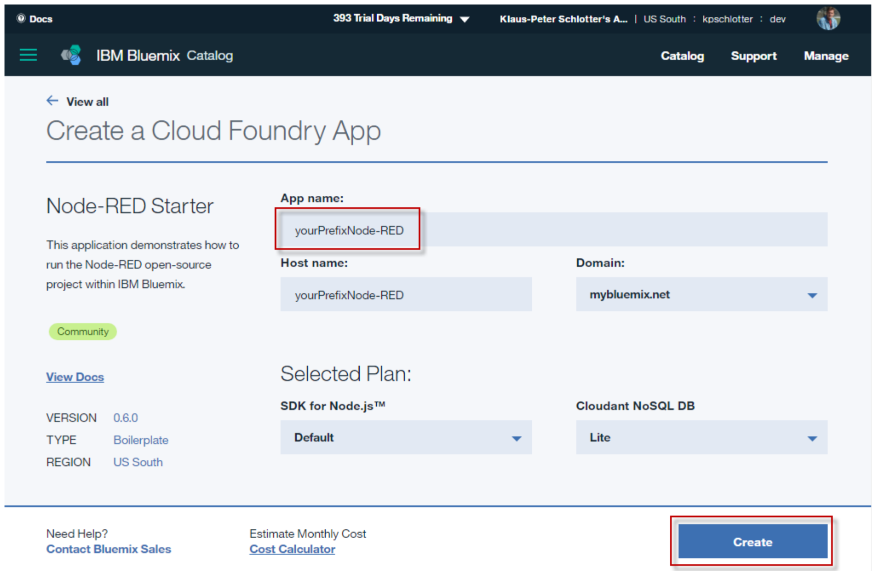
Node-RED Starter
This application demonstrates how to run the Node-RED open-source project within IBM
Community

Python Flask
A simple Python Flask application that will get you up and running quickly.
Lite Community

Ruby Sinatra
Develop a Ruby web application using the Sinatra framework.
Lite Community

Step 3

Give your App a unique name and click the **Create** button.



Create a Cloud Foundry App

Node-RED Starter

This application demonstrates how to run the Node-RED open-source project within IBM Bluemix.

Community

View Docs

VERSION	0.6.0
TYPE	Boilerplate
REGION	US South

App name: yourPrefixNode-RED

Host name: yourPrefixNode-RED

Domain: mybluemix.net

Selected Plan:

SDK for Node.js™: Default

Cloudant NoSQL DB: Lite

Need Help?
Contact Bluemix Sales

Estimate Monthly Cost
Cost Calculator

Create

It may take a while until a Node-RED environment is instantiated and started in your IBM Cloud environment

Step 4 In your App dashboard click the link for the running Node-RED application. You will first see the Welcome screen that lists some configuration steps. Click **Next**.

Step 5 You can secure your Node-RED editor by specifying a user id and password. Select the options that best suit your needs and click **Next**.

Step 6 On the *Browse Available IBM Cloud nodes* just click **Next**.

Step 7 On the *Finish the Install* screen click **Finish**.

Your Node-RED environment will be started.

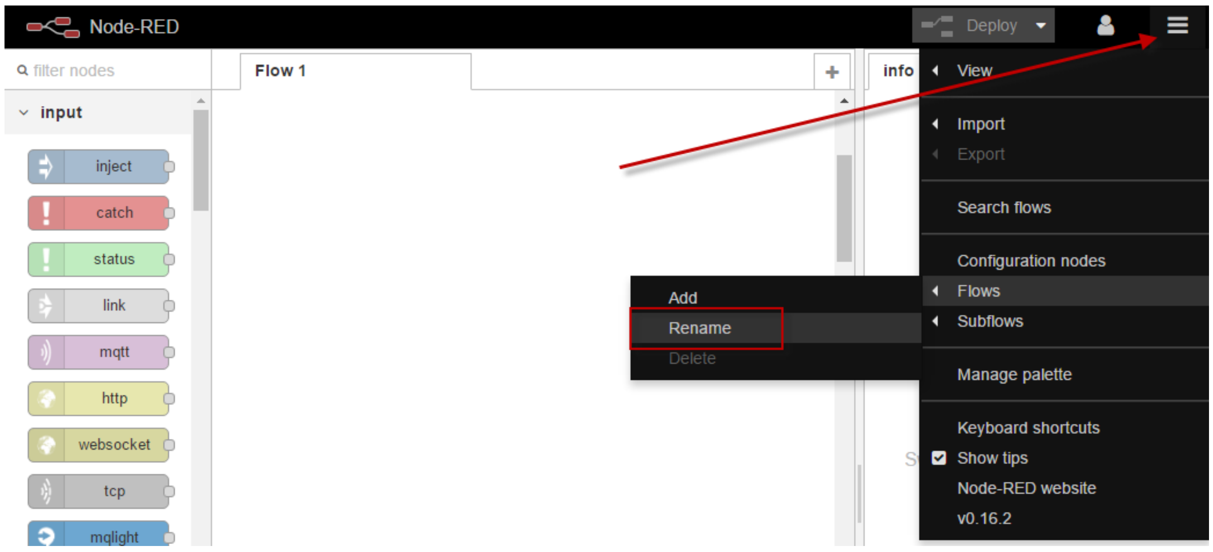
Step 8 On the Node-RED on IBM Cloud screen just click

Go to your Node-RED flow editor.

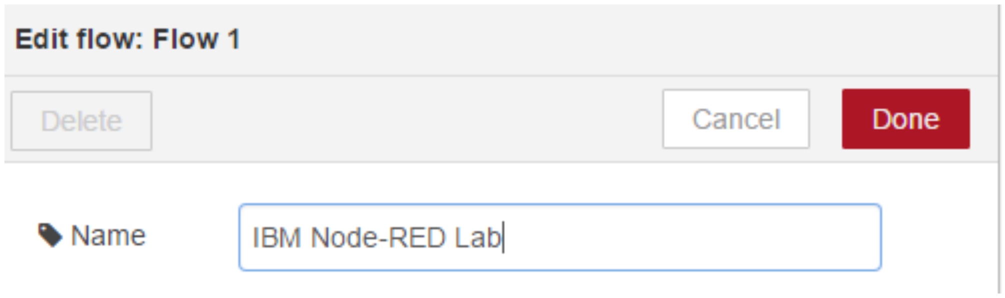
Step 9 Eventually enter the username and password created in Step 5 and click **Login**.

Step 10 A Node-RED Flow editor opens with an empty panel named *Flow 1*.

Step 11 Rename your Flow via the menu to IBM Node-RED Lab.



Step 12 Click **Done**.



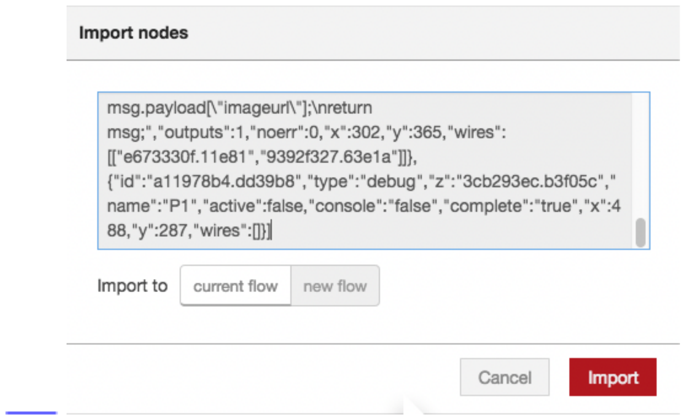
Name: IBM Node-RED Lab

Step 13 **(Optional:)** If you have problems creating the flow described in the following steps, you can also import the flow from here.

a) Open the file in a text editor.

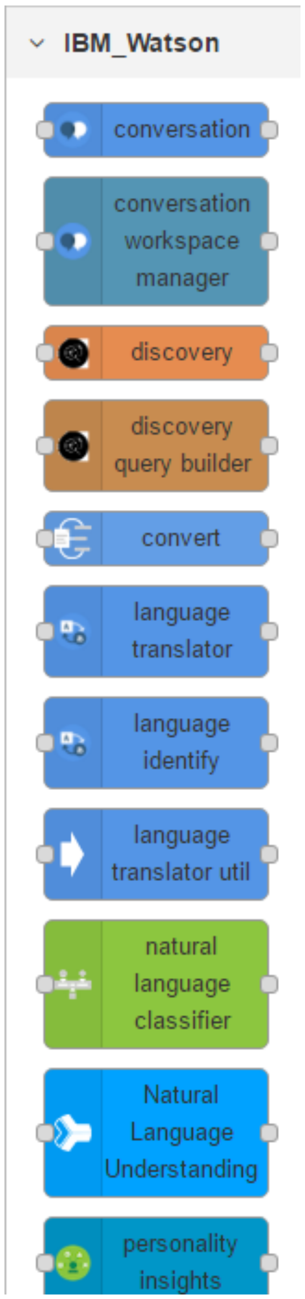
b) Copy all content to the clipboard

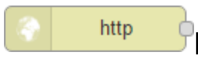
c) From the Node-Red editor menu select **Import → Clipboard**, paste the code, select **import to current flow** and click **Import**.



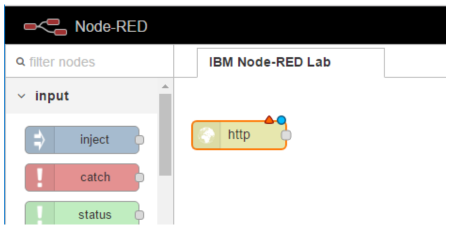
```
msg.payload[\"imageurl\"];\nreturn
msg;","outputs":1,"noerr":0,"x":302,"y":365,"wires":
[["e673330f.11e81","9392f327.63e1a"]]},
{"id":"a11978b4.dd39b8","type":"debug","z":"3cb293ec.b3f05c","
name":"P1","active":false,"console":"false","complete":"true","x":4
88,"y":287,"wires":[]}]
```

Step 14 In the left side bar of the Editor you see a lot of standard Node-RED nodes. At the end of the list there are certain categories already customized by IBM, such as **IBM_Watson**.

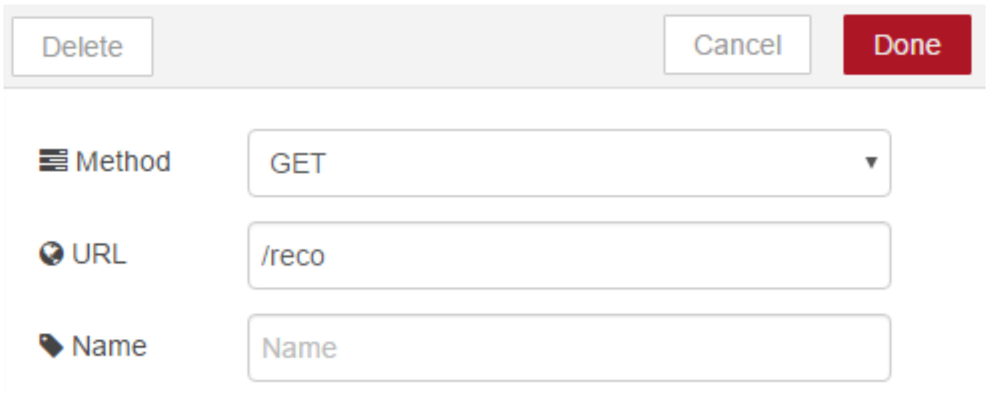


Step 15 From the input category of nodes drag the  node and drop it on the editor pane. The Info pane on the right side bar give you some information about the node.

IBM®



Step 16 Double click the **http** input node and specify the http **GET** request and an url of **/reco** and click **Done**.

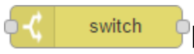


Method: GET

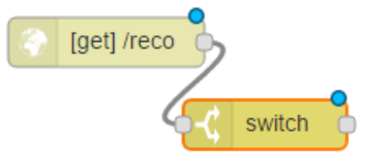
URL: /reco

Name: Name

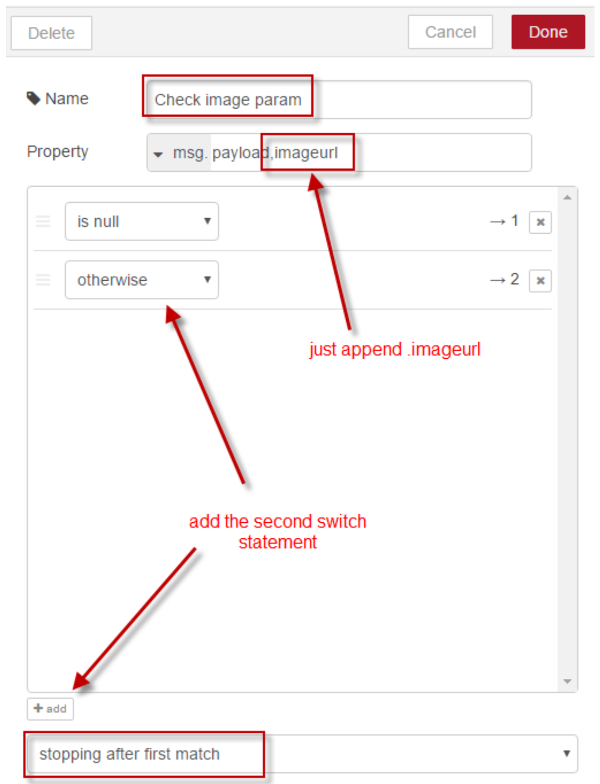
This is the URL appended to your Bluemix Node-RED application to initial- ize the new flow **IBM Node-RED lab**.

Step 17 In the nodes function section drag a  node and drop it on the editor pane.

Step 18 Select the *output* of the /reco node and drop it on the *input* of the **switch** node.



Step 19 Double click the **switch** node and enter the info shown on the following image and click **Done**.



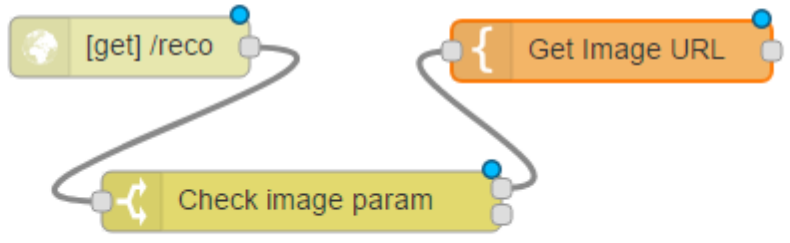
Name: Check image param

Property: msg. payload.imageurl

is null	→ 1
otherwise	→ 2

stopping after first match

Step 20 From the *function* section of nodes drag a node and drop it on the editor pane and connect the "is null" **output** of the switch to the **input** of the template.



Step 21 Double click the **template** and specify the following information:

a) Name: Get Image URL

b) Leave property as msg.payload

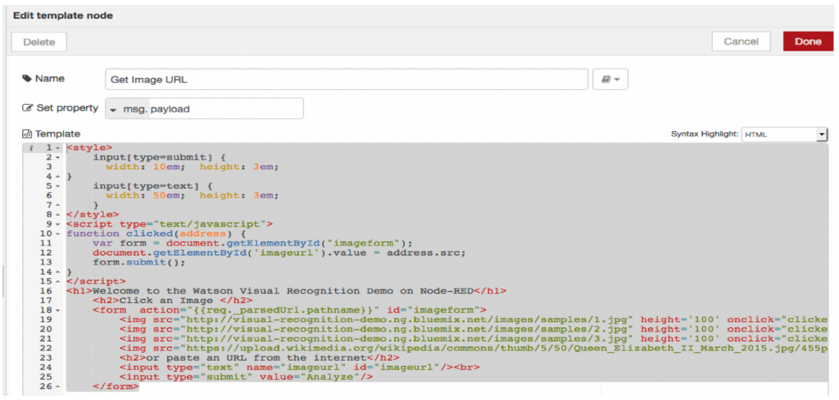
c) Syntax Highlight: HTML

d) Format: Moustache template Template:

e) Template:

```
<style>
input[type=submit]{
    background-color: rgb(85,150,230);
    border: none;
    color: white;
    padding: 16px 32px;
    text-decoration: none;
    margin: 4px 2px;
    cursor: pointer;
    font-size: 15px;
    border-radius: 10px;
)
input[type=text] {
    width: 50em;  height: 3em;
}
</style>
<script type="text/javascript">
function clicked(address) {
 var form = document.getElementById("imageform");
 document.getElementById('imageurl').value = address.src;
 form.submit();
}
</script>
<h1>Welcome to the Watson Visual Recognition Demo on Node-RED</h1>
 <h2>Click an Image </h2>
 <form  action="{{req._parsedUrl.pathname}}" id="imageform">
 <img src="http://visual-recognition-
demo.ng.bluemix.net/images/samples/1.jpg" height='100'
onclick="clicked(this)"/>
 <img src="http://visual-recognition-
demo.ng.bluemix.net/images/samples/2.jpg" height='100'
onclick="clicked(this)"/>
 <img src="http://visual-recognition-
demo.ng.bluemix.net/images/samples/3.jpg" height='100'
onclick="clicked(this)"/>
 <img
src="https://upload.wikimedia.org/wikipedia/commons/thumb/5/50/Queen_
Elizabeth_II_March_2015.jpg/4
55px-Queen_Elizabeth_II_March_2015.jpg" height='100'
onclick="clicked(this)"/>
 <h2>or paste an URL from the internet</h2>
        <input type="text" name="imageurl" id="imageurl"/><br>
        <input type="submit" value="Analyze"/>
</form>
```

f) Click **Done**.



Name: Get Image URL

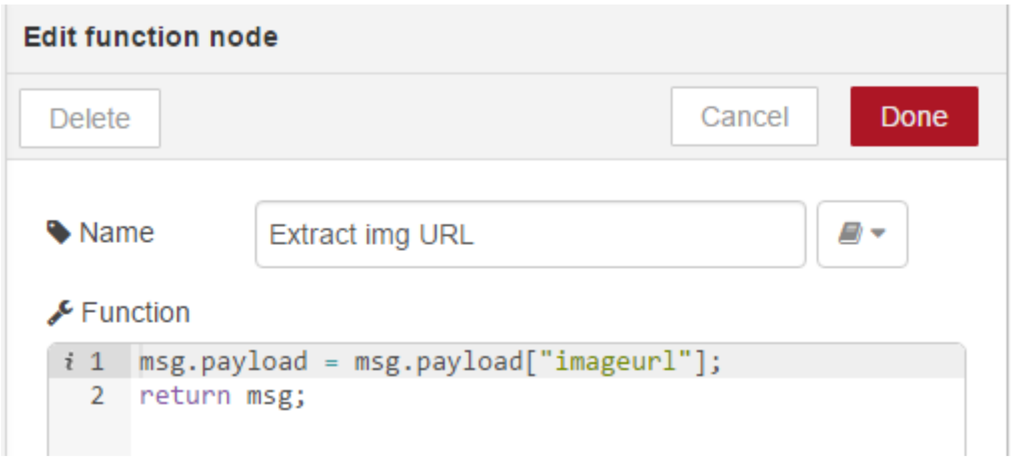
Set property: msg. payload

Template

Syntax Highlight: HTML

```
<style>
    input[type=submit] {
      width: 10em;  height: 3em;
}
    input[type=text] {
      width: 50em;  height: 3em;
    }
</style>
<script type="text/javascript">
function clicked(address) {
    var form = document.getElementById("imageform");
    document.getElementById('imageurl').value = address.src;
    form.submit();
}
</script>
<h1>Welcome to the Watson Visual Recognition Demo on Node-RED</h1>
    <h2>Click an Image </h2>
    <form  action="{{req._parsedUrl.pathname}}" id="imageform">
        <img src="http://visual-recognition-demo.ng.bluemix.net/images/samples/1.jpg" height='100' onclick="clicke
        <img src="http://visual-recognition-demo.ng.bluemix.net/images/samples/2.jpg" height='100' onclick="clicke
        <img src="http://visual-recognition-demo.ng.bluemix.net/images/samples/3.jpg" height='100' onclick="clicke
        <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/5/50/Queen_Elizabeth_II_March_2015.jpg/455p
        <h2>or paste an URL from the internet</h2>
        <input type="text" name="imageurl" id="imageurl"/><br>
        <input type="submit" value="Analyze"/>
    </form>
```

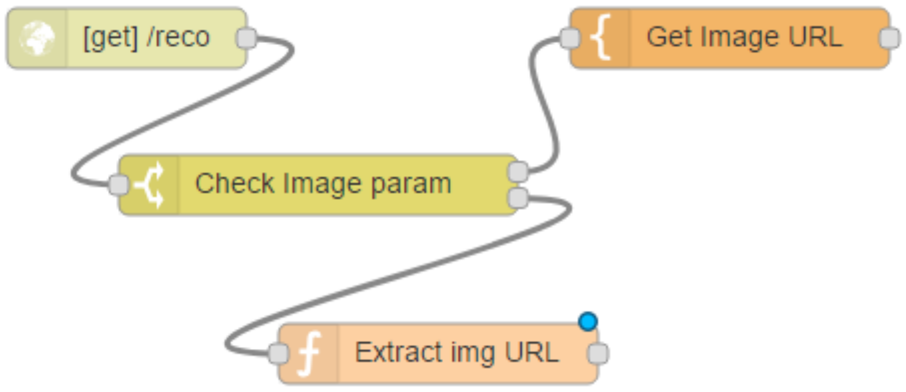
Step 22 Add a function node (named *Extract img URL* here) to convert the *imageurl* JSON object to a string and assign it to the payload to be provided as input to the Visual Recognition node:

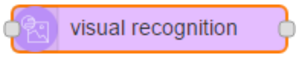


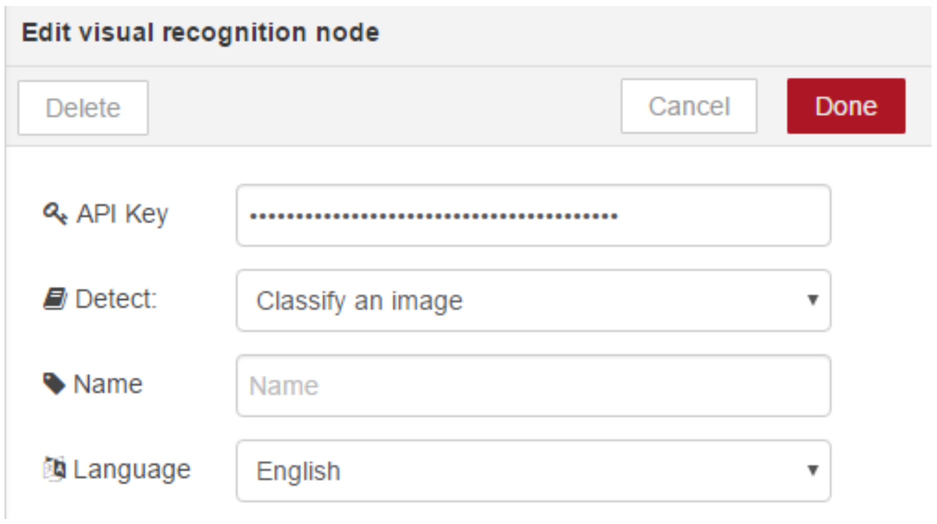
Name: Extract img URL

Function

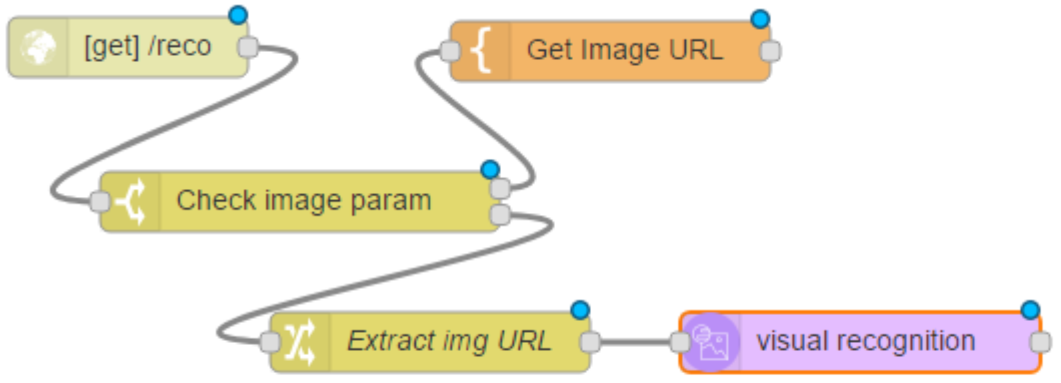
```
msg.payload = msg.payload["imageurl"];
return msg;
```



Step 23 From the **IBM_Watson** category select the  node and drop it behind the change node from previous step. Double click the **Visual Recognition** node and enter your **api key** you have created for the Visual Recognition service in Lab 01. Click **Done**.


Edit visual recognition node

API Key	••
Detect:	Classify an image
Name	Name
Language	English



Step 24 Add another behind the Visual Recognition node. Click **Done**.

a) Name: Report

b) property: message.payload

c) Syntax Highlight: mustache

d) Format: Mustache template

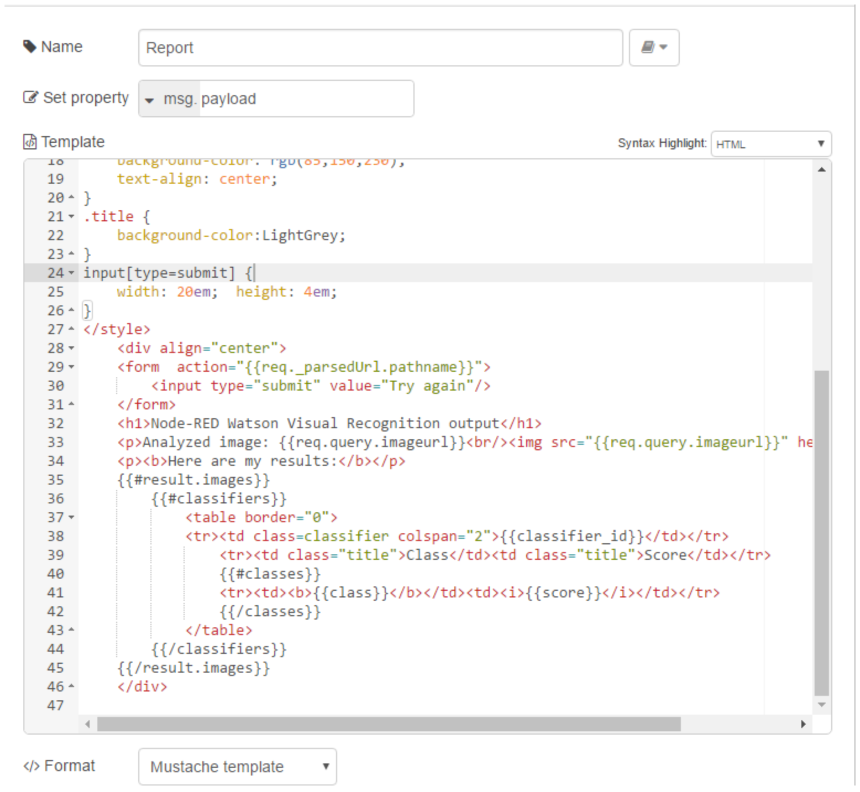
e) Template

```
<div align="center">
    <form  action="{{req._parsedUrl.pathname}}">
        <input type="submit" value="Try again"/>
    </form>
<h1>Node-RED Watson Visual Recognition output</h1> <p>Analyzed image:
{{req.query.imageurl}}<br/><img src="{{req.query.imageurl}}" height='200'/></p>
<p><b>Here are my results:</b></p> {{#result.images}}

        {{#classifiers}}
            <table border="0">
            <tr><td class=classifier
colspan="2">{{classifier_id}}</td></tr>
                <tr><td class="title">Class</td><td
class="title">Score</td></tr>
                {{#classes}}
```

```
<tr><td><b>{{class}}</b></td><td><i>{{score}}</i></td></tr>
                {{/classes}}
            </table>
        {{/classifiers}}
    {{/result.images}}
</div>
```

f) Optional Styling at the top of the template

```
<style> h4 {

    text-align: center;
    margin: 10px;
}
table {
    width: 480px;
    margin-top: 10px;
}
th, td {
    padding: 8px;
    text-align: left;
    border-bottom: 1px solid #ddd;
    background-color: #FFFFFF;
    width: 50%;
}
.classifier {
    background-color: rgb(85,150,230);
    text-align: center;
}
.title {
    background-color:LightGrey;
}
input[type=submit]{
    background-color: rgb(85,150,230);
    border: none;
    color: white;
    padding: 16px 32px;
    text-decoration: none;
    margin: 4px 2px;
    cursor: pointer;
    font-size: 15px;
    border-radius: 10px;
) </style>
```



Name: Report

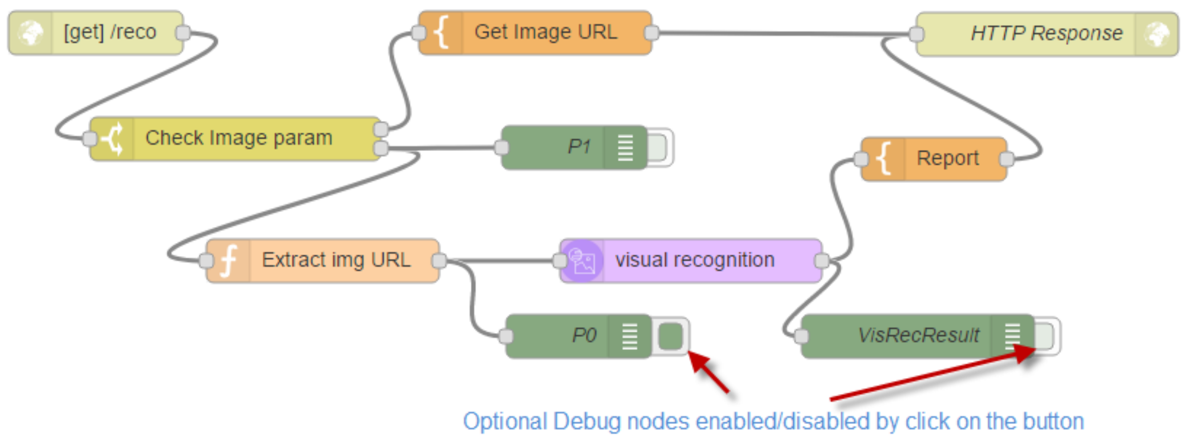
Set property: msg. payload

Template — Syntax Highlight: HTML

```
    background-color: rgb(83,150,230);
    text-align: center;
}
.title {
    background-color:LightGrey;
}
input[type=submit] {
    width: 20em;  height: 4em;
}
</style>
    <div align="center">
    <form  action="{{req._parsedUrl.pathname}}">
        <input type="submit" value="Try again"/>
    </form>
    <h1>Node-RED Watson Visual Recognition output</h1>
    <p>Analyzed image: {{req.query.imageurl}}<br/><img src="{{req.query.imageurl}}" he
    <p><b>Here are my results:</b></p>
    {{#result.images}}
        {{#classifiers}}
            <table border="0">
            <tr><td class=classifier colspan="2">{{classifier_id}}</td></tr>
                <tr><td class="title">Class</td><td class="title">Score</td></tr>
                {{#classes}}
                <tr><td><b>{{class}}</b></td><td><i>{{score}}</i></td></tr>
                {{/classes}}
            </table>
        {{/classifiers}}
    {{/result.images}}
    </div>

```

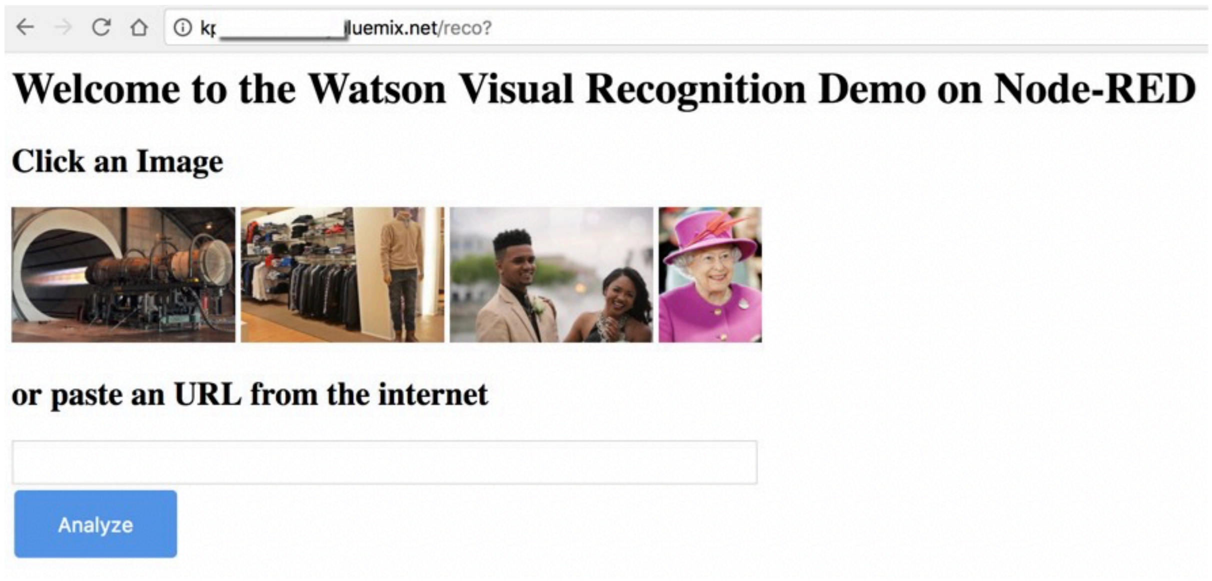
Format: Mustache template

Step 25 As the final node add a http response, name it *HTTP Response,* and connect it as shown in the following picture. Optional debug nodes display the output of the node in the right side bar on the debug tab.

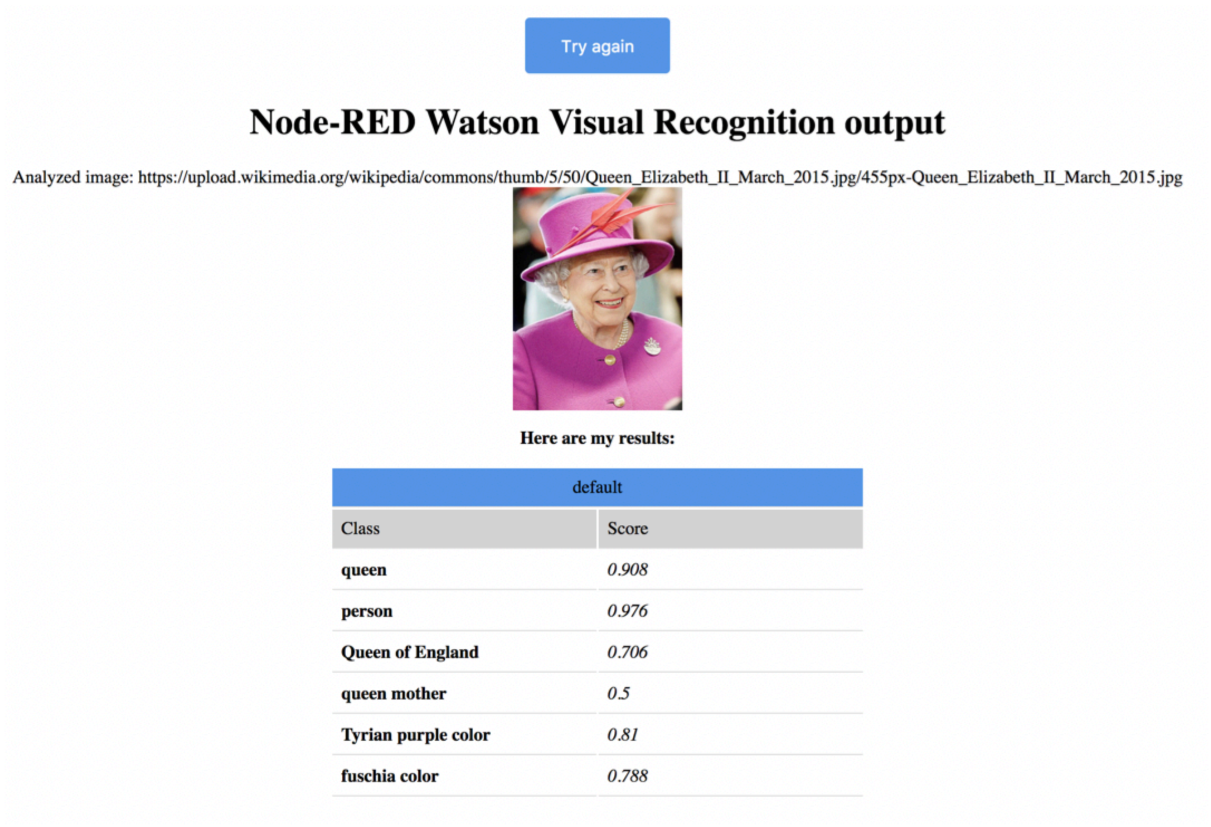


Step 26 Click the Deploy button at the top of the Node-RED application to de- ploy the flow in your environment.

Step 27 Test the flow by calling the url http://<yourNode-REDHost>.mybluemix.net/reco


kp[redacted]luemix.net/reco?
Welcome to the Watson Visual Recognition Demo on Node-RED
Click an Image
or paste an URL from the internet
Analyze

Step 28 **Click** on of the images or **Copy and Paste** any image URL from the internet and click **Analyse**



Node-RED Watson Visual Recognition output

Analyzed image: https://upload.wikimedia.org/wikipedia/commons/thumb/5/50/Queen_Elizabeth_II_March_2015.jpg/455px-Queen_Elizabeth_II_March_2015.jpg

Here are my results:

default	
Class	Score
queen	*0.908*
person	*0.976*
Queen of England	*0.706*
queen mother	*0.5*
Tyrian purple color	*0.81*
fuschia color	*0.788*

This completes Your Lab!