

# Hands-on Lab WCS-3077 Getting Started with Node-RED and IBM Watson

Steven Chamberlin, IBM WDP  
Miguel Gonzalez, IBM WDP

© Copyright IBM Corporation 2017

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

This document is current as of the initial date of publication and may be changed by IBM at any time.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

## Part 1: Create Your First Node-RED Flow

## Overview

This part shows how to launch your own instance Node-RED by using the IBM Bluemix boilerplate. You can deploy Node-RED as a stand-alone application, but using the Bluemix platform as a service means that you don't need to worry about installing Node-RED prerequisites. You will then proceed to create your first flow, which uses the Watson Language translator service to translate incoming text into the target specified language.

## Prerequisites

You need an [IBM Bluemix account](#).

**Important:** You can use only 12 free services in IBM Bluemix. In this course, you will need 2 services. Please ensure that you have enough available service slots to complete the lab.

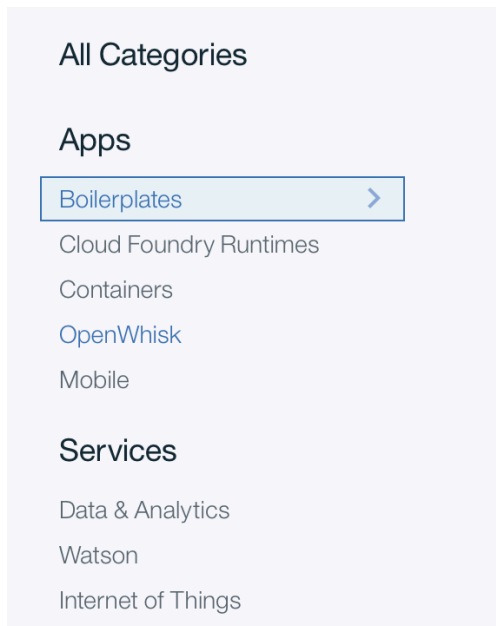
In this lab, you will use the following services:

- Cloudant (used by Node-RED to store metadata)
- Watson Language Translator

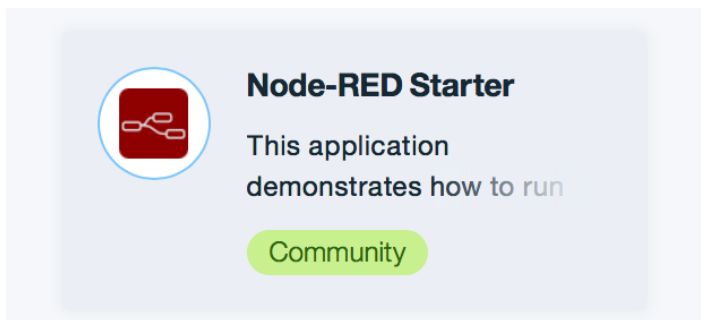
## Step 1. Create a Node-RED instance

In this step, you will create an instance of Node-RED running on IBM Bluemix

1. Open the Chrome browser. Log in to Bluemix. For the US, use <https://console.ng.bluemix.net/>. Then, click **Catalog**.



2. Click **Boilerplates**.
3. From the boilerplates list, click **Node-RED Starter**.



4. Give your Node-RED starter application a unique name and host name. The host name does not need to be the same as the app name, but it can be the same.

## Create a Cloud Foundry Application

---

### Node-RED Starter

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

Community

[View Docs](#)

VERSION	0.5.0
TYPE	Boilerplate
REGION	US South

App name:

devworks-course-app

Host name:

devworks-course

Domain:

mybluemix.net

Selected Plan:


SDK for Node.js™

Default





Cloudant NoSQL DB

Lite

5. Select **Create**. Wait for your new application to start

 devworks-course-app

Status: ● Your app is running

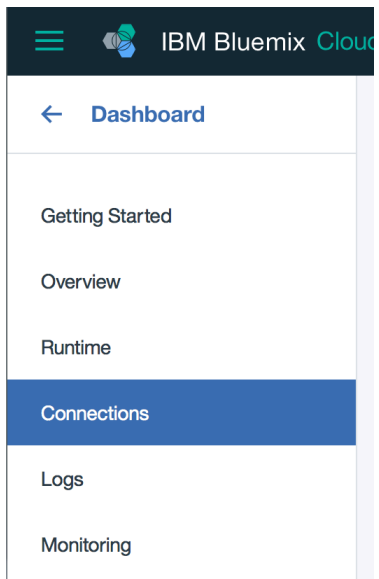
[View App](#)    

## Step 2. Bind Watson services

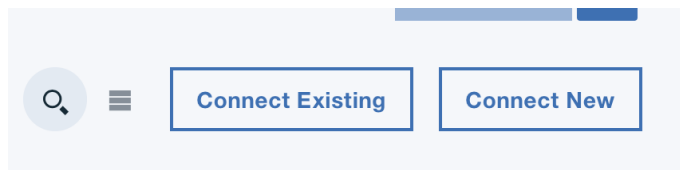
In this section, you'll bind the following Watson Language service.

To bind or connect the Watson service to your Node-RED application, you can either create a new Watson service or use an existing one.

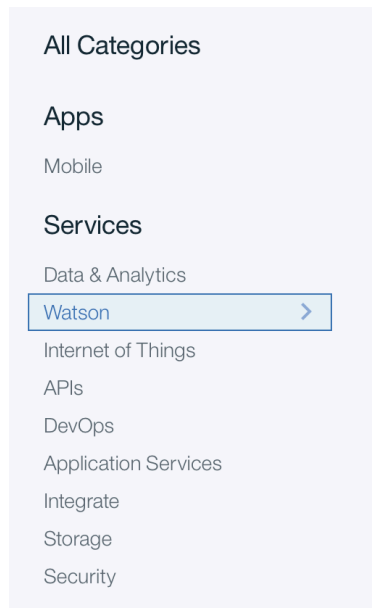
1. Click **Connections**.



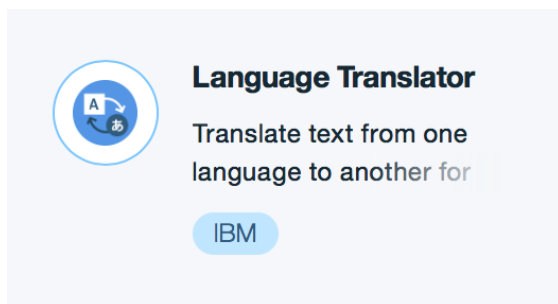
2. Click **Connect New**.



3. In the left pane, under **Services**, select **Watson**.



4. Click **Language Translator**.





5. Make sure that it is connecting to your Node-RED instance. You should see the name of your application in the **Connect to** field.

## Language Translator

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:

IBM

Connect to:

devworks-course-app ▼

6. If you want to be able to create custom translations, select the **Advanced** pricing plan. You will not need the Advanced plan for the labs in this course.

### Pricing Plans

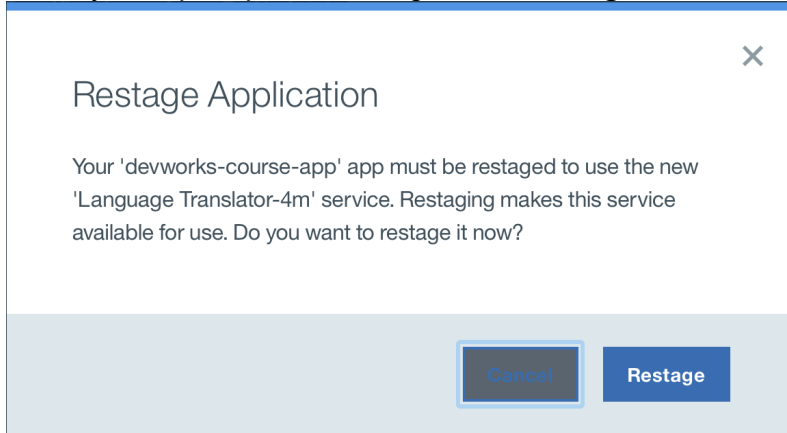
Monthly prices shown are for country or region: [United Kingdom](#)

PLAN	FEATURES	PRICING
Standard	Standard Translations (First 250,000 characters are free)	£0.0121 GBP/THOUSAND CHAR
✓ Advanced	Standard Translations Custom Translations Custom Model Maintenance (Pro-Rated Daily)	£0.0121 GBP/THOUSAND CHAR £0.0605 GBP/THOUSAND CHAR £9.07 GBP/INSTANCE MONTH

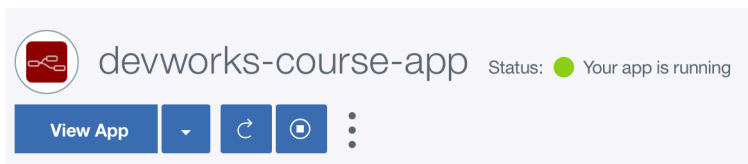
Base models are charged at the standard rate, usage of custom models incur additional charges.

Terms

7. Click **Create** to create the Service. This action creates new credentials that allow you to use the service. By connecting to your Node-RED instance, these credentials are available for the application to use.
8. When you're prompted to restage, click **Restage**

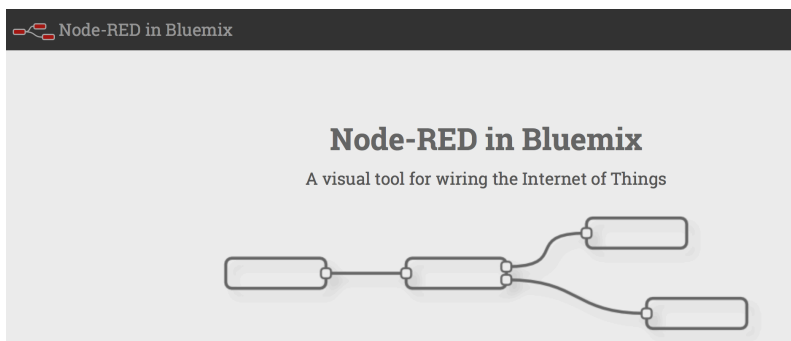


9. Wait for the application to restart. When your application has restarted, you will see the



status reported as “Your app is running.”

10. Click **View App** to open your running Node-RED application to see the landing page of your Node-RED instance. (Landing page is ‘/index.html’)

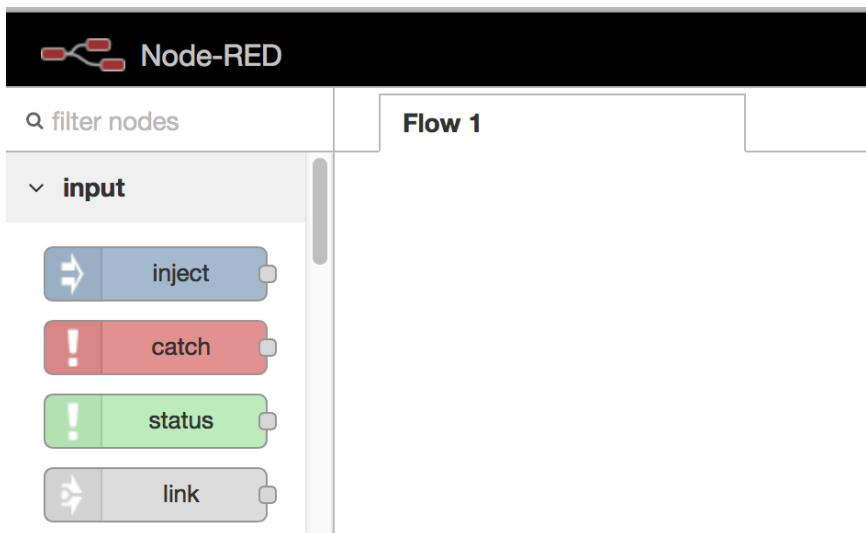


11. Click **Go to your Node-RED flow editor** to open the editor.

Go to your Node-RED flow editor

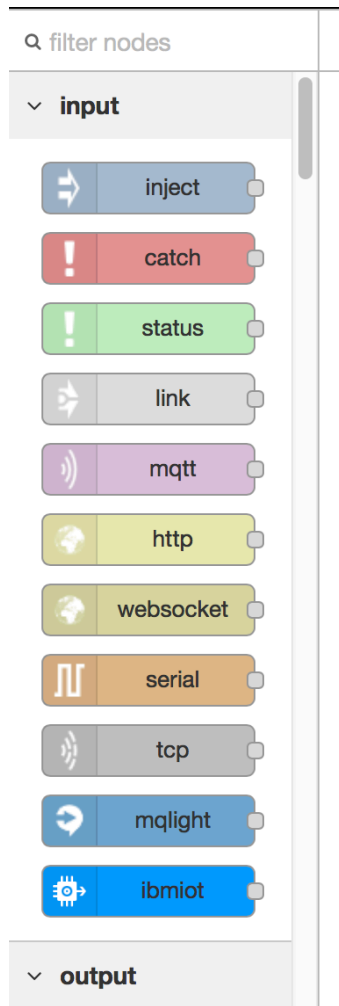
[Learn how to password-protect your instance](#)

[Learn how to customise Node-RED](#)



## Step 3. Create your first flows

In this section, you'll create your first flows in the Node-RED flow editor. An application in Node-RED is called a *flow*.



The palette in the left column shows you all the available nodes.

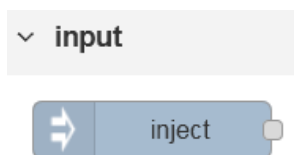
The nodes are grouped by category. The main categories of nodes are input, output, and function.

- Use input nodes to input data into a Node-RED application, or flow.
- Use output nodes to send data outside of a Node-RED flow.

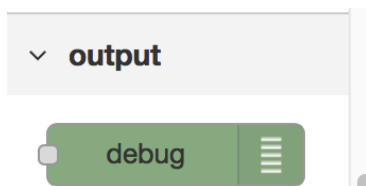
- Use function nodes to process data. You can use the function node to pass messages through a JavaScript function.



1. Select an input **inject** node and drag it onto the canvas.



2. Select an output **debug** node and drag it onto the canvas.



3. Link, or wire, the two nodes together by clicking and dragging your cursor from one node to the other. Note that the debug and inject nodes change their display names when you drag

them onto the canvas. This name change is expected and shows additional context for the node.



4. Double-click the **timestamp** node. For the **Payload** field, select **string**.

The screenshot shows the 'Edit inject node' dialog box. It has a title bar 'Edit inject node' and two buttons: 'Cancel' and 'Done'. The dialog contains several fields: 'Payload' (with a dropdown menu open), 'Topic', 'Repeat' (with a checkbox), and 'Name'. The dropdown menu for 'Payload' is open, showing options: 'a\_z', 'flow.', 'global.', 'a\_z string' (which is highlighted), '0\_9 number', 'boolean', '{} JSON', and 'timestamp'. Below the fields is a yellow note box that reads: 'Note: "interval" and "at a specific time" will use cron. See info box for details.'

5. Enter a string, such as `Hello, this is my first Node-RED application.`
6. In the **Name** field, enter a name for this node, such as `Hello World inject.`

Edit inject node

Cancel

Done

✉ Payload

▼ a\_z

Hello World, this is my first node red flow

📄 Topic

🔄 Repeat

none

⬆⬇⬆

☐ Inject once at start?

🔖 Name

Hello World Inject

Note: "interval between times" and "at a specific time" will use cron.

See info box for details.

- Click **Done**.

Cancel

Done

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.

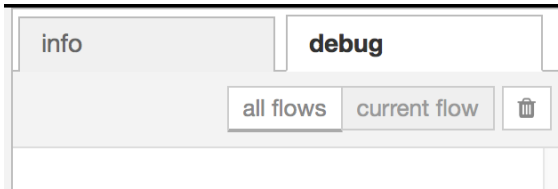
bluemix.net

Deploy

▼

debug

The **debug** node writes to the **debug** tab, which helps you monitor the flow through your application.

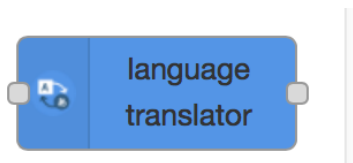


9. To initiate the flow, click the tab linked to the **inject** node.

You now see the output on the debug tab.



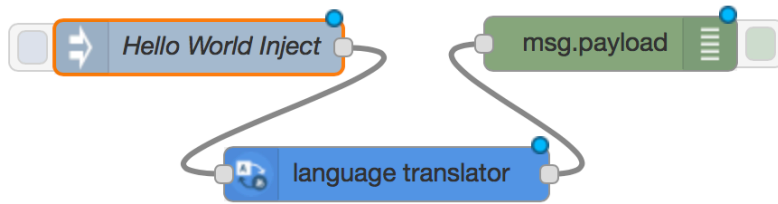
10. In the **filter nodes** search field, enter `translator` to find the **language translator** node.



11. Drag the node onto the canvas so that it lies in between the **inject** and **debug** nodes. You can move the nodes to make more space.



To remove an unwanted line, select the line and press Delete on your keyboard.



12. Double-click the **language translator** node. Select to translate from the inject English to another language. Select the **Conversational** domain.

**Edit language translator node**

Cancel Done

Name

Mode

Domains

Source

Target

Parameters ☒

Scope Use Local Parameters

**Note:** When using with the language translation utility, 'Use Local Parameters' must be unchecked.

13. Click **Done** to save your changes.
14. Select the **language translator** node and click the **info** tab.

info		debug
Node		
Type	watson-translator	
ID	2702aef8.d8fd52	

Notice that the node puts its translated output 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.

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

**Translation Mode.**

The text to translate should be passed in on `msg.payload`.

The translated text will be returned on `msg.payload`.

The full response from the service will be returned on `msg.translation`

Source and destination language parameters

15. Open the **debug** node and change the output to `msg.translation`. Enter the word `translation` **after** `msg`.

**Edit debug node**

Cancel Done

Output `msg.translation`

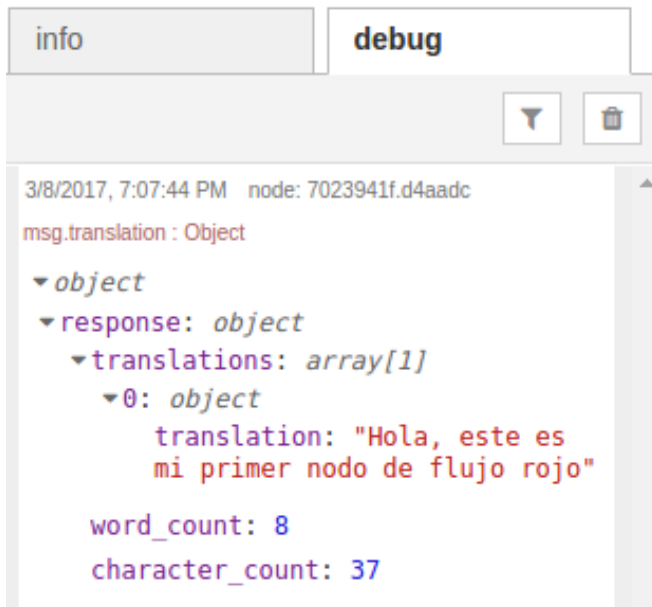
to `debug tab`

Name `Name`

16. Click **Done** to save your changes. Then, deploy your flow.
17. Initiate the flow by clicking the tab on the **inject** node.



18. View the translated text in the **debug** tab. The application is translating the text that you entered in the **Payload** field of the **inject** node and returning it in the msg.translation object. By expanding the tree you can see the details contained within that object, including the translation and other information like the word and character counts.



19. Modify the **language translator** node to use the News domain.

The screenshot shows the 'Edit language translator node' dialog box. It has a title bar and two buttons: 'Cancel' and 'Done'. The dialog contains several configuration fields:

- Name:** A text input field with the value 'Name'.
- Mode:** A dropdown menu with the value 'Translate'.
- Domains:** A dropdown menu with the value 'News'.
- Source:** A dropdown menu with the value 'English'.
- Target:** A dropdown menu with the value 'Spanish'.
- Parameters:** A checkbox labeled 'Parameters' which is checked.
- Scope:** A text input field with the value 'Use Local Parameters'.

At the bottom of the dialog, there is a yellow note box with the text: "Note: When using with the language translation utility, 'Use Local Parameters' must be unchecked."

20. Click **Done** and deploy your app.

21. Initiate the flow and notice the difference in the translations for the different domains. For example, the difference between **News** and **Conversation** domains is the difference between how a newspaper might be translated and how a spoken conversation might be translated.

Most of the time, there will be no difference, but occasionally certain terms and phrases are translated differently.



You should now have a running instance of a Node-RED application on Bluemix with the Watson Language Translator and Conversation services, and you should be able to create basic flows.

## Part 2: Build a web page and create a REST API in Node-RED

## Overview

This section shows you how to build on your first Node-RED flow. You'll create Node-RED flows that use:

- HTTP and HTML web pages
- JavaScript
- AJAX to consume a REST API
- Language identification feature of the Watson Language Translator service

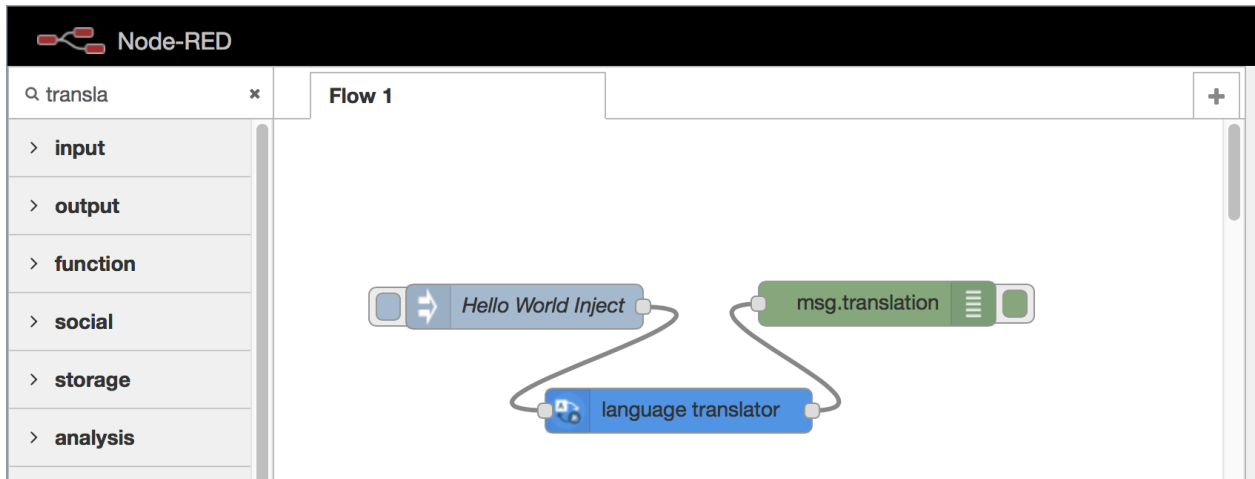
## Prerequisites

Complete part 1. You should already have a running instance of Node-RED with the Watson Language Translator service connected.

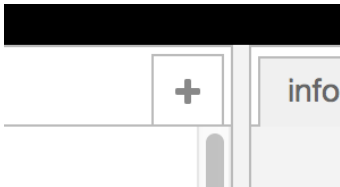
## Step 1. Create a simple web page

In this section, you will create a basic "Hello World" web page by using Node-RED.

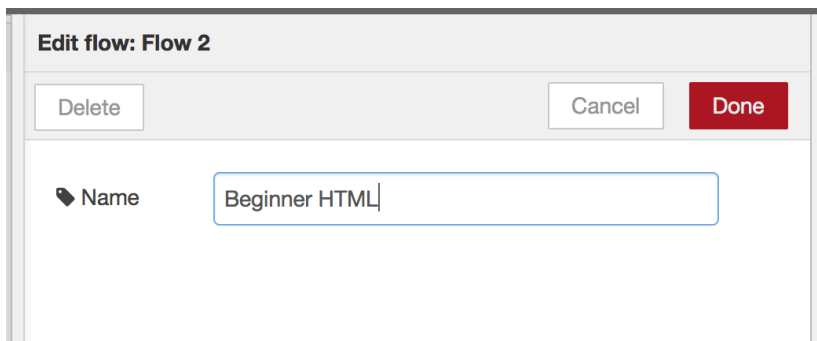
6. Open the flow editor in your instance of Node-RED.



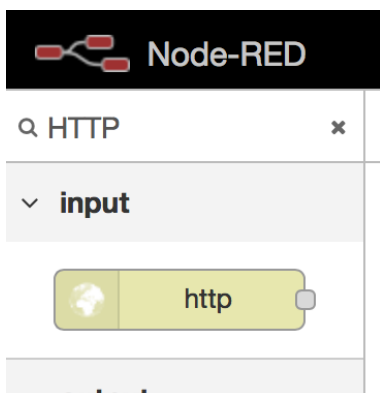
7. Create a new flow tab by clicking +.



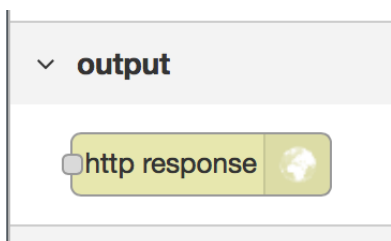
8. Double-click the new tab and enter a name for the new flow tab. Then click **Done**.



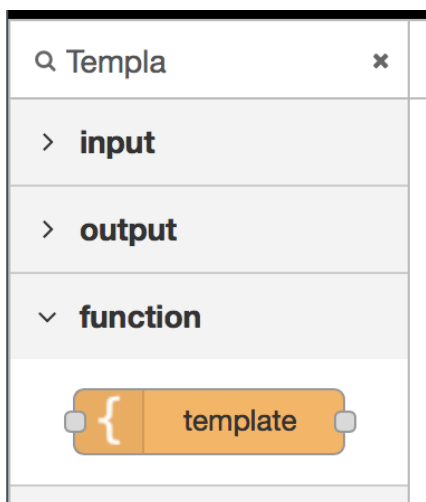
9. Drag and drop an input **http** node onto the canvas. Use the **filter nodes** search field to find the nodes.



10. Drag and drop an output **http response** node onto the canvas.



11. Drag and drop a **template** node onto the canvas between the **http** and **http response** nodes.



12. Wire the three nodes together.





13. Double-click the **template** node to edit it. Enter the following simple HTML code:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>Hello to Watson on Node-RED</div>
  </body>
</html>
```

**Edit template node**

Cancel Done

Name

Set property

Template Syntax Highlight:

```
1 <html>
2 <head>
3   <title>Hello World</title>
4 </head>
5 <body>
6   <div>Hello to Watson on Node-RED</div>
7 </body>
8 </html>
```

14. Click **Done** to save your changes.

15. Double-click the input **http** node. Edit it to create an HTTP route to your web page by

**Edit http in node**

Cancel Done

Method

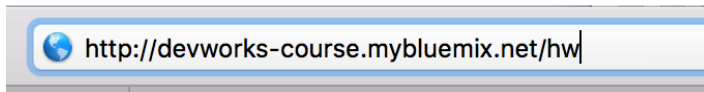
URL

Name

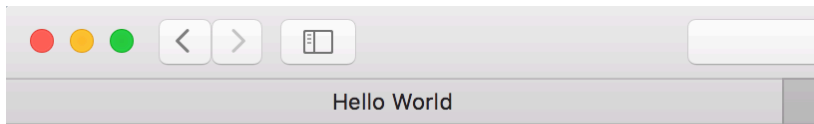
entering `/<some string>` in the **URL** field. Enter a name such as HTTP Hello World.

16. Click **Done** and deploy your changes.

17. Open a new browser tab and navigate to your new web page. The web address will be based on your Node-RED web address that is appended with the URL of your web page.



You should see your "Hello World" web page.

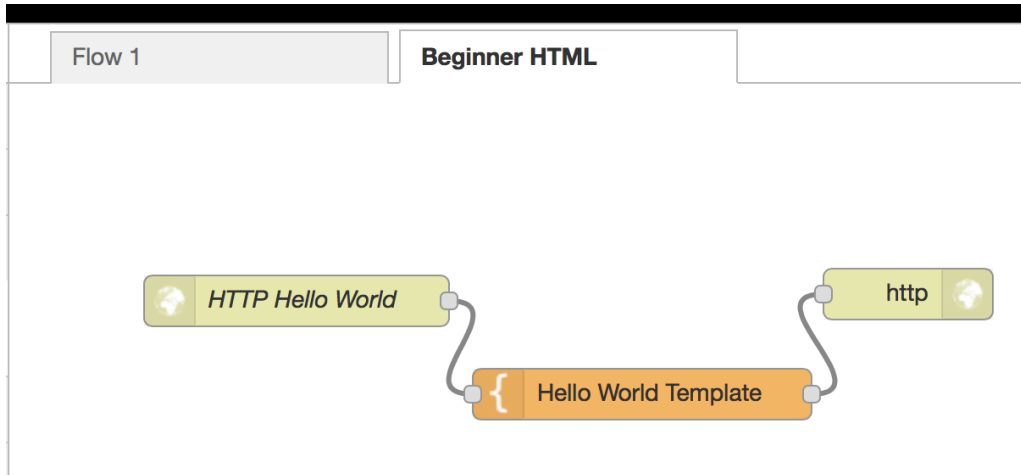


**Hello to Watson on Node-RED**

## Step 2. Add JavaScript to your web application

In this section, you'll modify your "Hello World" web page to include a text entry field and JavaScript.

1. Open the flow editor to your instance of Node-RED.



2. Double-click the **template** node and replace the contents with the following HTML:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>Hello to Watson on Node-RED</div>
    <div id="id_hello">
      <span>Hello</span>
      &nbsp;
      <span id="id_nameout"></span>
    </div>

    <form id="id_form">
      <div>
        <span>
          What is your name:
        </span>
        <span>
          <input type="text" name="name"
            id="id_name"/>
        </span>
      </div>
      <div>
        <input type="submit" value="Enter"
          id="id_enter"/>
      </div>
    </form>
```

```

<script
  src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js">
</script>
<script type="text/javascript">
  $(document).ready(function() {
    setupPage();
  });

  function setupPage() {
    $('#id_hello').hide();
    $('#id_form').submit(onSubmitClicked);
    enterbutton();
  }

  function onSubmitClicked(event) {
    $('#id_nameout').text($('#id_name').val());
    $('#id_hello').show();
    $('#id_form').hide();
    event.preventDefault();
  }

  function enterbutton() {
    $(function() {
      $("form input").keypress(function (e) {
        if ((e.which && e.which == 13) || (e.keyCode && e.keyCode == 13)) {
          $('#id_enter').click();
          return false;
        } else {
          return true;
        }
      });
    });
  }
</script>
</body>
</html>

```

3. Click **Done** and deploy your changes.
4. Try out your JavaScript-enabled web page. Enter your name and click **Enter**.

Hello World

Hello to Watson on Node-RED

What is your name:

Hello World

Hello to Watson on Node-RED

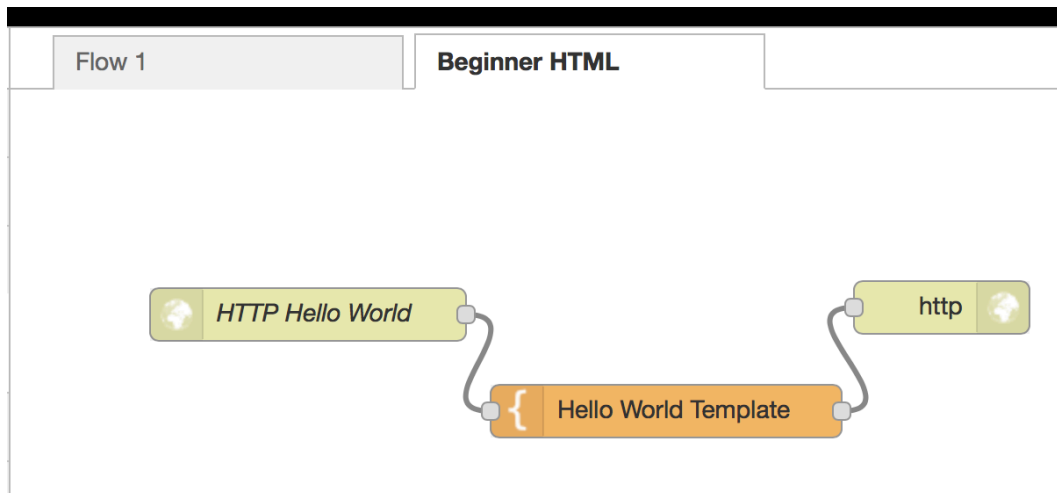
Hello Soheel

## Step 3. Create a REST API

In this section, you'll modify your "Hello World" web page to invoke a REST API that you will create. This will show you how to do two important tasks:

- How to consume a REST API. This could be any REST API from any source, which allows you to add extra capability into your application.
- How to create a REST API in Node-RED. This will allow you to create reusable chunks of functionality that can be consumed by other flows that you create in Node-RED, and also be consumed by other applications, even if they are not Node-RED applications and even if they are not running in Bluemix.

1. Open the flow editor to your instance of Node-RED.



2. Double-click the template node and replace the contents with the following HTML:

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <div>Hello to Watson on Node-RED</div>
    <div id="id_hello"><span>I think you are typing text in:</span>
      &nbsp;
    <span id="id_textlang"></span></div>

    <form id="id_form">
      <div>
        <span>
          Enter your text to be processed:
        </span>
        <span>
          <input type="text" name="name" id="id_text"/>
        </span>
      </div>
    </div>
```

```
        <input type="submit" value="Enter" id="id_enter"/>
    </div>
</form>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js">
```

```
</script>
```

```
<script type="text/javascript">
    $(document).ready(function() {
        setupPage();
    });

    var LANGUAGES = { 'ar' : 'Arabic',
        'en': 'English' ,
        'es': 'Spanish',
        'fr': 'French',
        'it': 'Italian',
        'de': 'German',
        'pt': 'Portuguese',
        'ko': 'Korean',
        'zh': 'Chinese'
    };

    function setupPage(){
        $('#id_hello').hide();
        $('#id_form').submit(onSubmitClicked);
        enterbutton();
    }

    function langLookup(isocode) {
        return (LANGUAGES[isocode] ? LANGUAGES[isocode] : isocode);
    }

    function processOK(response) {
        console.log(response);
        if (response.identifyresponse){
            $('#id_textlang').text(langLookup(response.identifyresponse));
        } else {
            $('#id_textlang').text('No response received from service');
        }
    }

    function processNotOK() {
        console.log('Error Processing');
        $('#id_textlang').text('Oops something went wrong');
    }

    function invokeAjax(message) {
        console.log('AJAX about to be invoked. ');
        console.log(message);

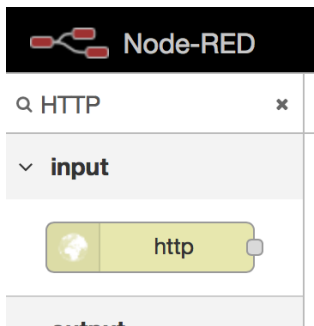
        var ajaxData = {};
        ajaxData.msgdata = message;

        $.ajax({
            type: 'POST',
            url: 'langidentify',
```

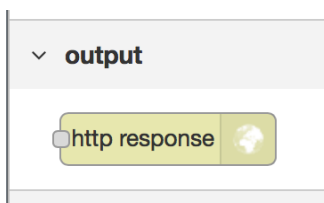
## Lab 1: create your first Node-RED flow

```
        data: ajaxData,  
        success: processOK,  
        error: processNotOK  
    });  
}  
  
function onSubmitClicked(event){  
    $('#id_hello').show();  
    event.preventDefault();  
  
    var txt = $('#id_text').val();  
    invokeAjax(txt);  
}  
  
function enterbutton(){  
    $(function() {  
        $("form input").keypress(function (e) {  
            if ((e.which && e.which == 13) || (e.keyCode && e.keyCode == 13)) {  
                $('#id_enter').click();  
                return false;  
            } else {  
                return true;  
            }  
        });  
    });  
}  
  
</script>  
</body>  
</html>
```

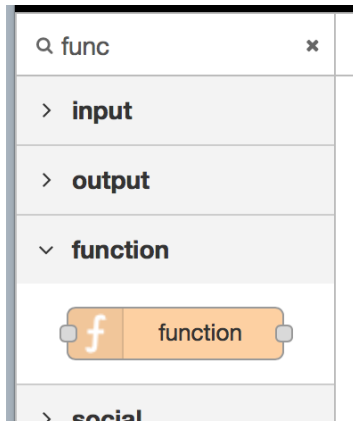
3. Drag and drop another **http** input node onto the canvas.



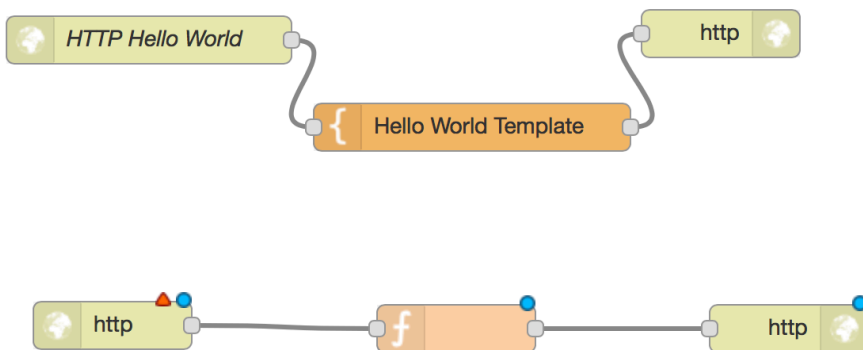
4. Drag and drop another output **http response** node onto the canvas.



5. Drag and drop a **function** node onto the canvas between the **http** and **http response** nodes.

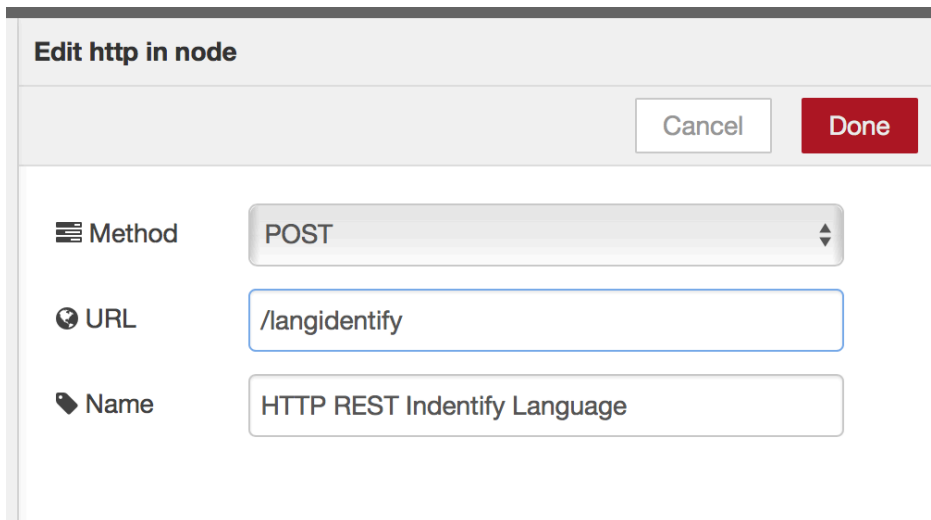


6. Wire the nodes together.



7. Double-click the input **http** node and specify the following information for each field:
- Method: POST
  - URL: /langidentify
  - Name: HTTP REST Identify Language





**Edit http in node**

Cancel Done

Method POST

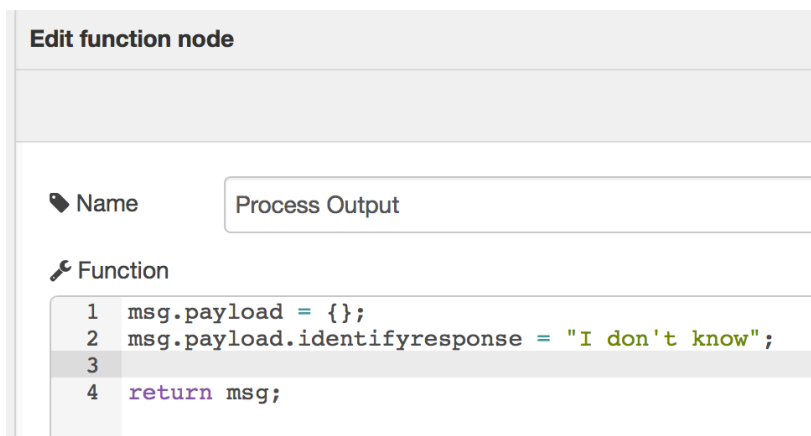
URL /langidentify

Name HTTP REST Indentify Language

8. Click **Done**.
9. Double-click the **function** node. Add the following code under **Function**:

```
msg.payload = {};  
  
msg.payload.identifyresponse = "I don't know";  
  
return msg;
```

The phrase "I don't know" will be your default answer when the service is unable to process the request. You haven't yet added the service, so this response is appropriate because



**Edit function node**

Name Process Output

Function

```
1 msg.payload = {};  
2 msg.payload.identifyresponse = "I don't know";  
3  
4 return msg;
```

your application doesn't know how to process the request.

10. Click **Done** and deploy your changes.

11. Test your application.

Hello World

Hello to Watson on Node-RED

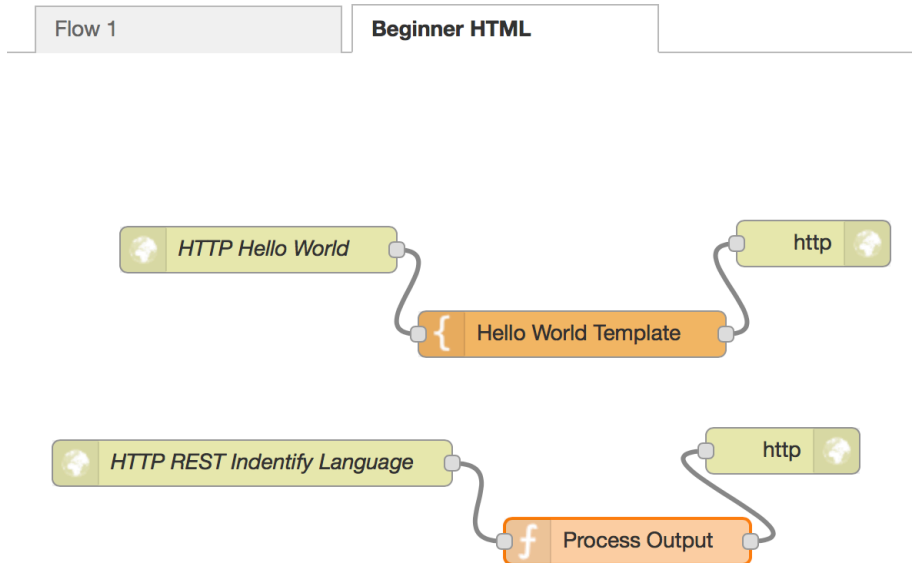
I think you are typing text in: I don't know

Enter your text to be processed: Can you identify this langu

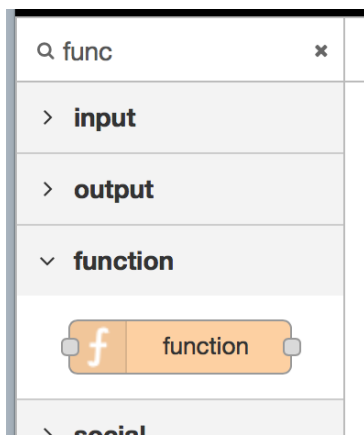
Enter

## Step 4. Consume the Watson Translator service

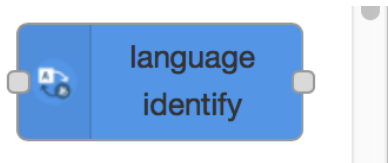
In this section, you will modify the REST API to invoke the Watson Language Translator service's Language Identification method.



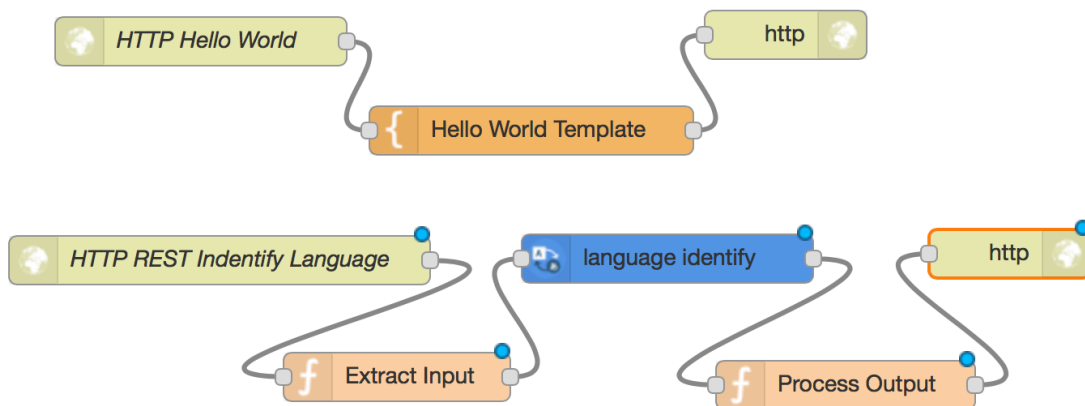
1. Open the flow editor in your instance of Node-RED.
2. Delete the connections between the last three nodes by clicking the connectors and pressing Delete on your keyboard.
3. Drag and drop a **function** node onto the canvas.



4. Drag and drop the **language identify** node onto the canvas.



5. Wire the nodes together.

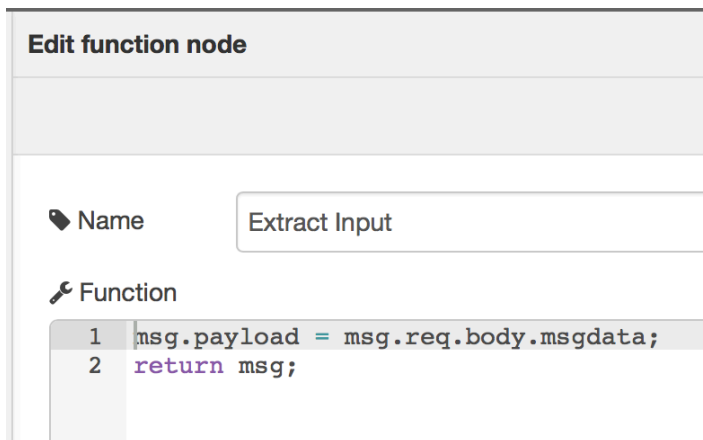


6. Double-click the new **function** node and enter code that takes the input from the REST call and makes it available for the Language Identify service to use. Enter this information and then click **Done**:

- Name: Extract Input

- Function:

```
msg.payload = msg.req.body.msgdata;  
return msg;
```



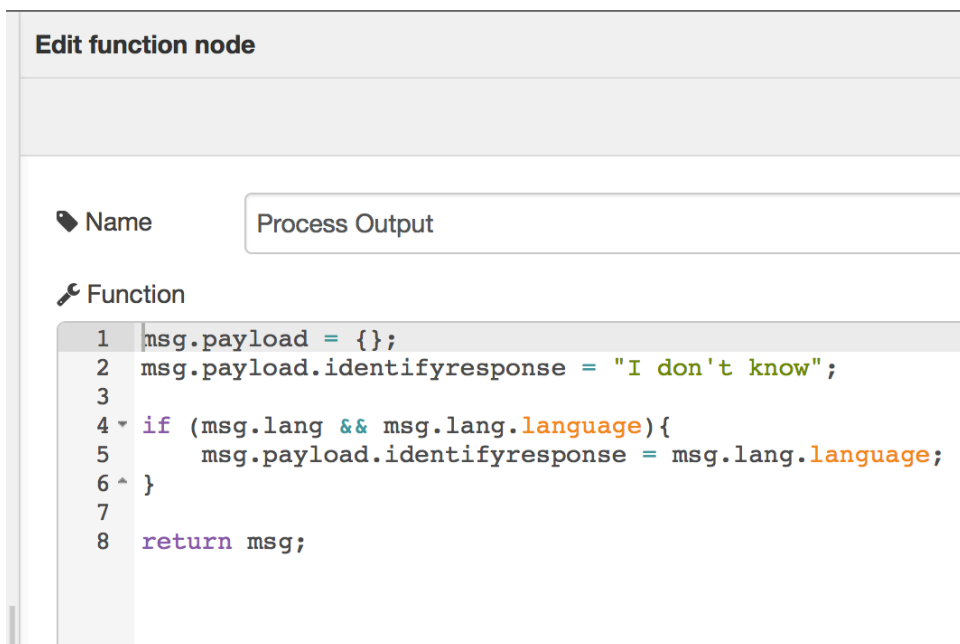
7. Double-click the second **function** node and add code that will send the response from the language identification service back to the client that is invoking the service. Enter this information and then click **Done**:

- Name: Process Output

- Function:

```
msg.payload = {};
msg.payload.identifyresponse = "I don't know";
if (msg.lang && msg.lang.language){
    msg.payload.identifyresponse = msg.lang.language;
}

return msg;
```



8. Deploy your changes.

9. Test your application by entering text other than English, such as Spanish text. Try other languages.

Hello World

Hello to Watson on Node-RED

I think you are typing text in: Spanish

Enter your text to be processed:

You now have a REST API that invokes the Watson Translator Language Identification method and an HTTP web application that invokes this API.

## You are done!

This concludes the Getting Started With Node-RED and Watson tutorial. This tutorial was based on labs 1 and 2 from the “Node-RED: basics to bots” developerWorks course. If you would like to complete this course, which contains 2 additional advanced labs, you can find more information about it here:

<https://developer.ibm.com/courses/all-courses/node-red-basics-bots/>

If you would like to explore more complex applications using Node-RED, try some of the starter kits available from Node Red Labs, part of the Watson Developer Clouds community, at this link:

<https://github.com/watson-developer-cloud/node-red-labs/tree/master/starter-kits>

Also if you have time, feel free to try the bonus section below, which shows you how to add new nodes to your Node-RED instance.

## Bonus Section

In this optional section, you will enhance the translator you built in step 1 to use a voice interface, leveraging the Watson “Speech to Text” (STT) and “Text to Speech” (TTS) services from Bluemix.

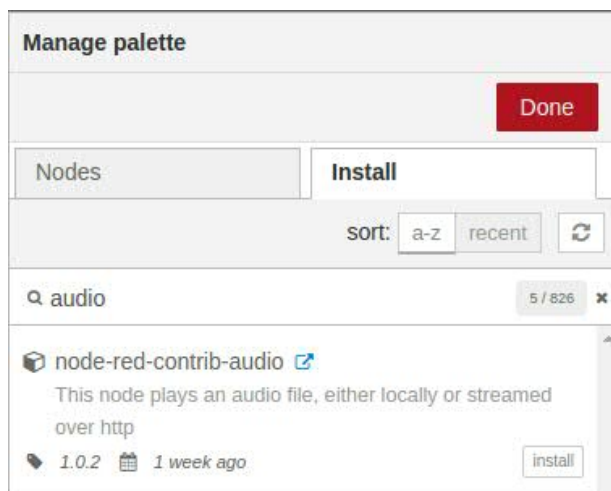
We will be using the Interpreter starter kit from Node-RED Labs at this URL:

<https://github.com/watson-developer-cloud/node-red-labs/blob/master/starter-kits/interpreter/README.md>

In order to get started, there are a few setup steps you will need to do first.

### Step 1. Add more nodes to your Node-RED

1. In the upper-right corner of your Node-RED, there are 3 vertical lines. Click on it, and then choose “Manage Palette”. Click on the “Install” tab and type in the “node-red-contrib-audio” package which contains a speaker node. Click “Done” to install.



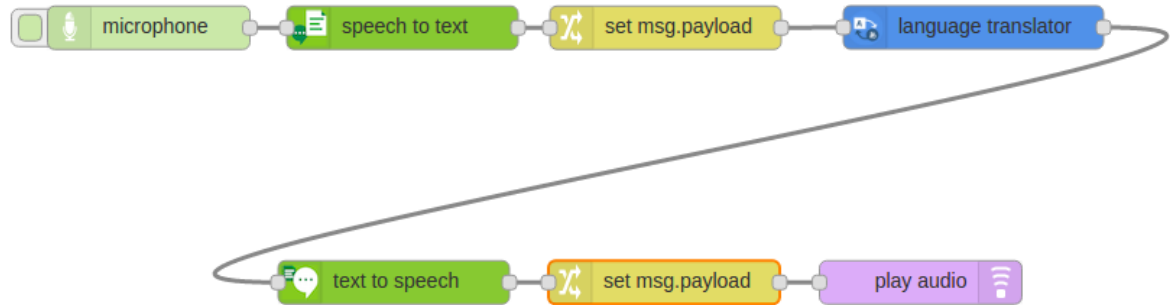
Do the same for “node-red-contrib-browser-utils” which contains a microphone node.

2. Make sure that your Chrome browser microphone is enabled. Click Chrome->Preferences, then click “Show advanced Settings” at the bottom. Under Privacy click “Content Settings”. Under Microphone, make sure “Ask when a site requires access to your microphone” is checked.
3. Now you will need to bind the Watson “Text to Speech” and “Speech to Text” services to your Node-RED application in Bluemix. Select your application from the dashboard page, and click the “Connect new” button, as you did in section 1 for the Language Translator service. Click the “Watson” category on the left side to filter for Watson services, then find the “Text to Speech” service and click on it. Click “Create” and when

it asks if you want to Restage, cancel. Next do the same for the “Speech to Text” service and then restage.

4. Next, import the flow for the translator application. You can find the flow here: <https://github.com/watson-developer-cloud/node-red-labs/blob/master/starter-kits/interpreter/interpreter.json>

To import the flow, click on the 3 vertical lines in the upper-right of Node-RED and select Import->Clipboard. Paste the JSON into the box and click “Import”. Your flow should look like this:



Please note that you should load your page with “https” instead of “http”, which is required by the microphone node.

5. To try out your translator, click once on the button to the left of the microphone to record, say a few words in English, and then click again to stop. If you are using a VM, it is important to leave a few seconds before the second click due to some delays when going through a VMware image. If you want to debug / test the microphone quality first (recommended), you could hook up the microphone directly to the “play audio” node to hear yourself.
6. If all goes well, the translator should speak the French translation of your words. To change the language, open up the “language translator” and “text to speech” nodes and change them to a different language, such as Spanish.