

**Hands-on exercise**

**Add language translation to your apps with IBM Watson**

**Introduction**

The goal of this exercise is to use the Language Translation service, Node-RED, and AngularJS for a Bluemix app that translates user-supplied text.

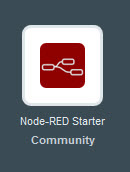
Do you want to add language translation to your applications? The Language Translation service in IBM Bluemix uses the IBM Watson cognitive computing system to convert user-entered text to other languages. The Node-RED flow editor makes it point-and-click easy to use the Language Translation service in your Bluemix apps. We explore how to use Node-RED to create an app that exposes a REST endpoint for a Language Translation service instance. Then, you'll add a user interface that we created with AngularJS. The UI includes a visualization of the AFINN-111 sentiment score for the text that the user enters. In just a few minutes, your application will be speaking English, Spanish, French, Portuguese, or Arabic — and reporting on the user's associated feelings at the same time.

*“ The Node-RED flow takes a message, gets the sentiment score for it, and then uses Watson to translate the message into Spanish. ”*

**Step 1. Set up the Bluemix services**

The language translation flow uses the Node-RED capabilities in Bluemix. To develop the flow, you first need to create a Node-RED application and add the Machine Translation service to it.

1. Log in to Bluemix.

2. In the Catalog, choose **Node-RED Starter**: 

3. Enter the unique name of your choosing in the Name and Host fields and click **CREATE**.

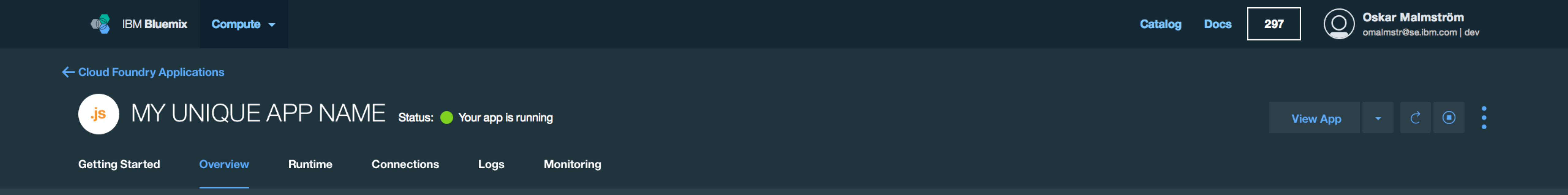
4. Return to the Bluemix dashboard and click the Node-RED application that you just created.

5. Click **Add a Service**.

6. Click **Language Translation** in the catalog's Watson category. You can see your Node-RED application in the App field of the **Add Service** dialog box.

7. Click **CREATE** and, when prompted, click **RESTAGE**.

8. After the application is restaged, click the **View App** URL to open your Node-RED application in  a new browser page:

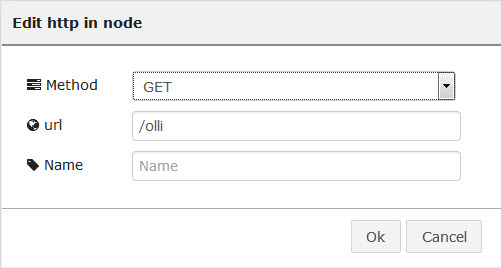


9. In your Node-RED application, click the **Go to your Node-RED flow editor** button.

**Step 2. Build the REST language translation flow in Node-RED**

Now you'll use the Node-RED flow editor to build a REST service. The flow takes a message, gets the sentiment score for it, and then uses Watson to translate the message (into Spanish in this case). Follow these steps to create, configure, and test the workflow:

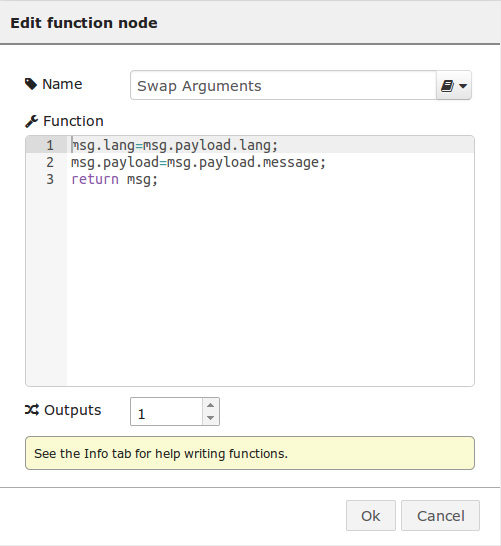
1. From the palette, drag an http input node onto the canvas.

2. Double-click the http node in the canvas to configure it for the GET method, and enter /olli in the url field: Click **Ok** to create a REST service listening on /olli.  

3. Drag the function node named function to the canvas. Using your mouse, connect the  output square on the http node to the input square on the function node.

4. Double-click the new node and type “Swap Arguments” in the Name field. Add the following three lines of code into the Function field:   
  
**msg.lang=msg.payload.lang;   
msg.payload=msg.payload.message;   
return msg;**

This function is needed because the http input node's default message payload output is a complex JSON object in which each request parameter is a field. The translation service and sentiment analysis both expect a simple string. The function copies the incoming message as a request parameter into the msg object's payload field so that it's consumable by the rest of the flow.



1. Click **Ok**.

6. Drag the sentiment analysis node to the canvas and connect it with the function node named  Swap Arguments. Sentiment analysis expects a string in human language and returns a sentiment score ranging from -5 to +5, where negative values code for negative sentiment, positive values code for positive sentiment, and zero means neutral sentiment.

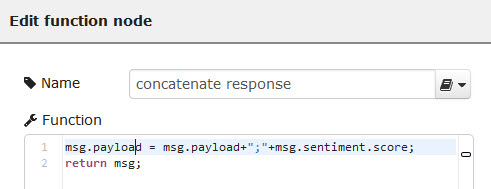
7. Drag the Language Translation IBM Watson node onto the canvas and connect it with the  sentiment node.

8. This node extracts the output from the Language Translation call and the sentiment node and transforms it into an easily consumable string to return.

Add another function node and connect it to the Language Translation node. Double-click the new node, provide a name (such as concatenate response), and add the following two lines of code into the Function field:

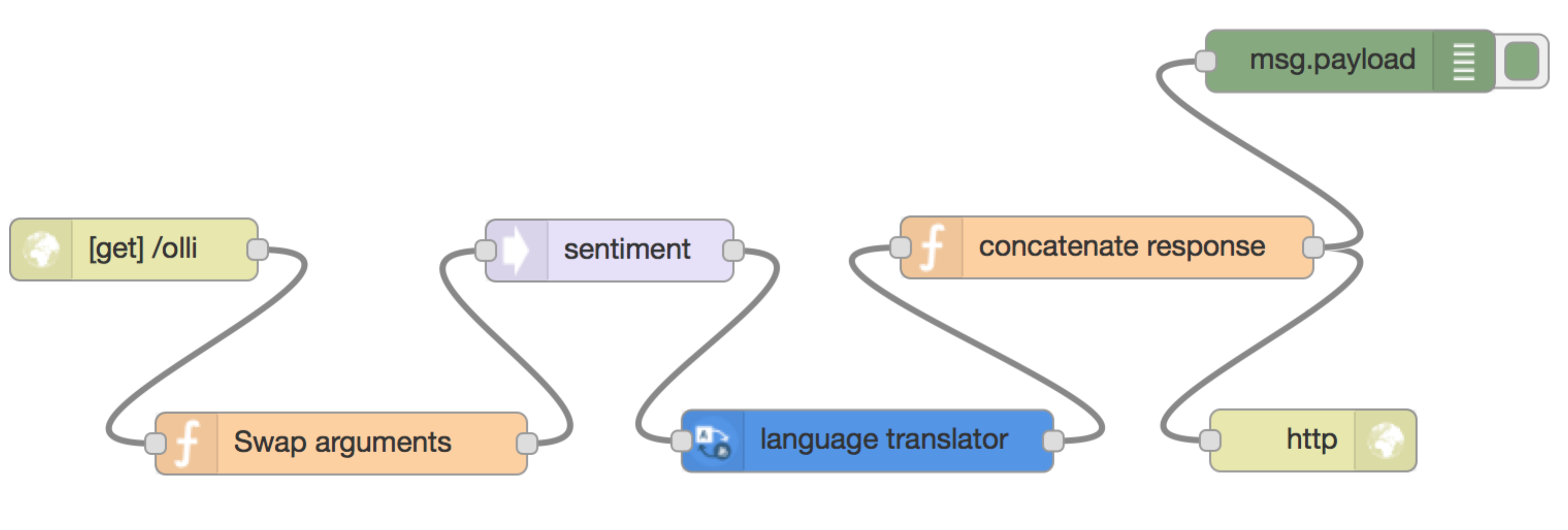
**msg.payload=msg.payload + “;” + msg.sentiment.score;**

**return msg;**



1. Click **Ok**.

10. Add an http response node and connect it to the function node. The completed flow in Node-RED should look like this:



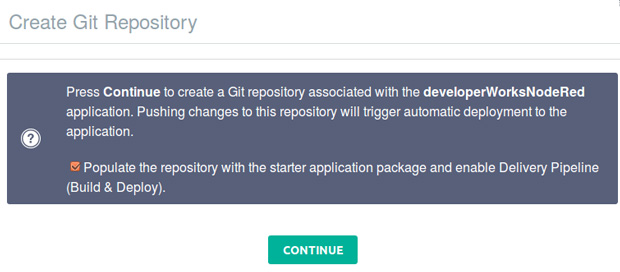
10. Click **DEPLOY**. Your REST service is up and running and can be used by everybody.

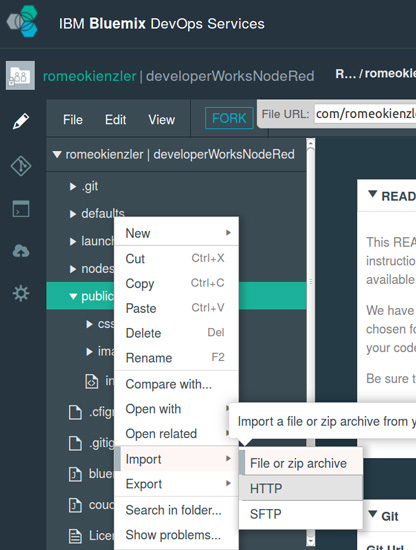
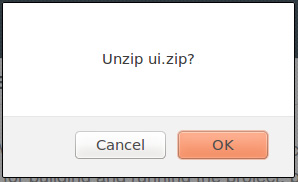
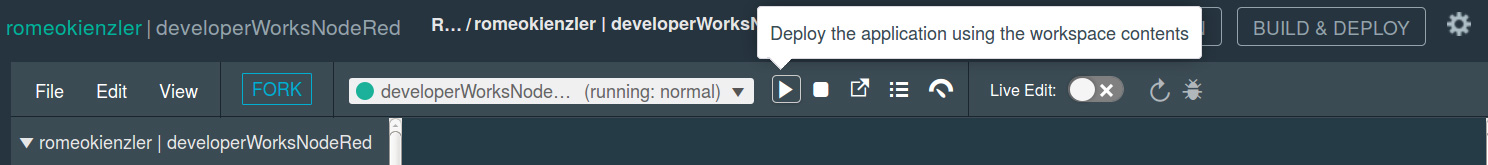
**Step 3. Add the AngularJS UI to Node-RED**

Now that the REST service is ready, you need an application to call it. We built a starter application for you by using AngularJS. This single-page app consists of a single index.html file and calls to REST services. All you need to do is download a copy of the code into Bluemix DevOps Services and deploy from there to inject the UI into the Node-RED runtime.

1. Return to the Bluemix dashboard and click your Node-RED application.
2. On the app overview page, click the **Add GIT** link. If you don’t have a git repository on Jazz hub, you will have to create one as part of this step.

3. Leave the check box selected and click **CONTINUE**:

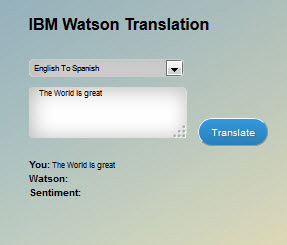


1. Click the **EDIT CODE** button.
2. In a new browser tab, go to “https://github.com/JohanRodin/WatsonTranslate” to download source code. The code is in the ui.zip file. Download the package to your local file system.
3. Unzip the files from ui.zip
4. In the directory tree of your Bluemix DevOps Services project's code editor, right-click the Public folder and   
   a) select New -> Folder and create a sub folder named “ui”   
   b) Drag and drop the unzipped files into this public folder:   
5. Select ui.zip from your local storage and click **OK** to unzip the file contents into your project's public directory:   
6. The code is now in your project's **public/ui** directory. You don't need to change the code, but you can examine index.html to see the logic. Line 36 is the REST service call to your Node- RED workflow. The customerController() function runs when the page is displayed and adds a submit() function to the scope of the view, which is called when the user clicks the Translate button. The button click triggers an HTTP request to the Node-RED REST service, passing it the message to translate along with the translation language the user selected. When the response is received, the translated message and the sentiment value are set in the scope and then displayed in the view.
7. Click the play button (the right-pointing triangle) to deploy the modified Node-RED boilerplate containing the AngularJS UI to Bluemix:   

Your app is running now at http://*yourappname*.mybluemix.net/ui.

**Step 4. Test the application**

1. Open http://*yourappname*.mybluemix.net**/ui** in a browser. Select the language translation you want, enter the text to translate, and click **Translate**: Under the covers,



the Angular code initiates a REST service call to your Node-RED workflow. The AngularJS application forms the string returned by the REST service.

2. Verify that the application worked correctly. You should see the text translated to the language you selected, the sentiment score, and the image of a stylized face whose expression reflects the score:

**Conclusion**

In this tutorial, you learned how to create a REST service quickly using Node-RED and how to access the service from an AngularJS application running on Bluemix. Your app uses the Bluemix Language Translation service to convert text between languages and also the sentiment analysis service to provide a sentiment score. You are now ready to build additional REST services using Node-RED and develop applications to consume those services.