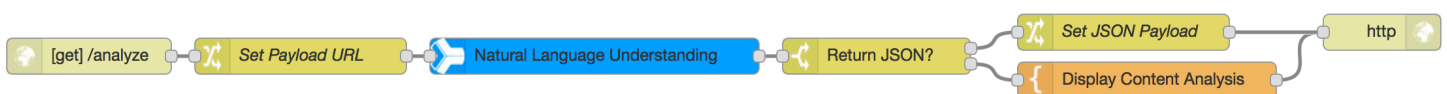# Natural Language Understanding in Node-RED

Hands-On Lab

JeanCarl Bisson | jbisson@us.ibm.com | @dothewww

Extract keywords, entities, concepts, sentiment and more from a news article
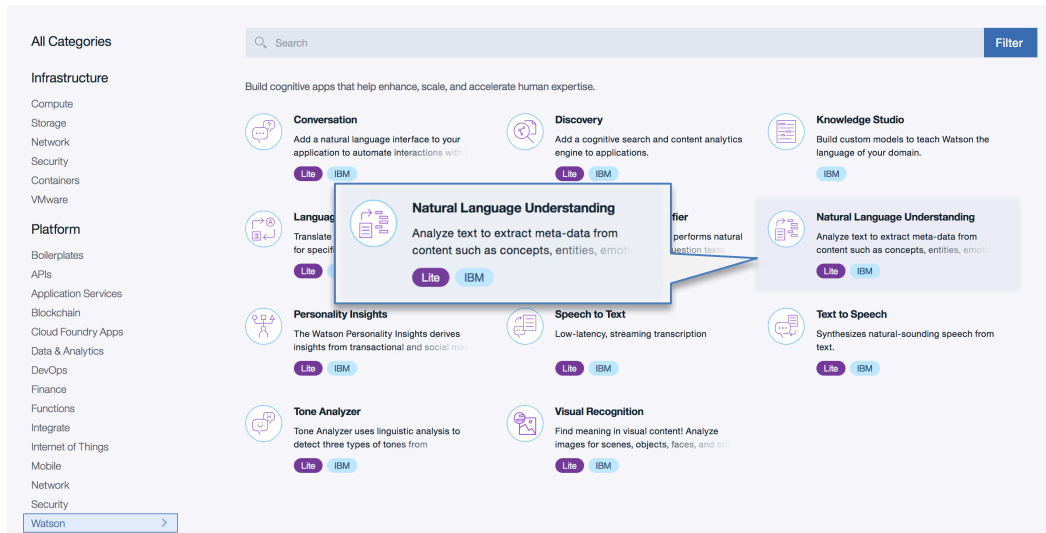(see *Analyze a News Article in Node-RED*)

Add Natural Language Understanding Service in IBM Cloud

The Natural Language Understanding service analyzes text to extract meta-data from content such as concepts, entities, keywords, categories, sentiment, emotion, relations, semantic roles, using natural language understanding. This tutorial uses the Node-RED boilerplate in IBM Cloud with the Natural Language Understanding service found under the **Boilerplates** section of the IBM Cloud catalog. To get started using the Natural Language Understanding service, you'll need to create service credentials.
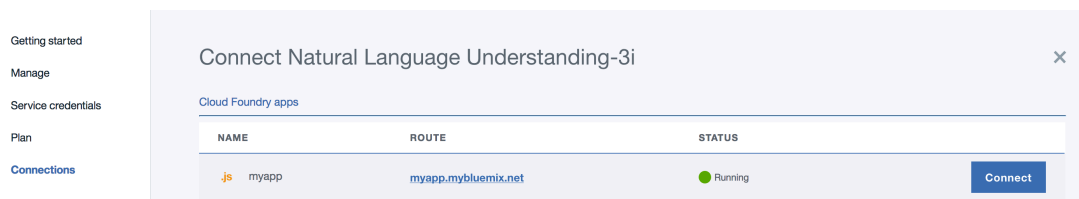
1. Click on the **Catalog** link in the top-right of the IBM Cloud Dashboard. Under the Watson section, click on the **Natural Language Understanding** tile.
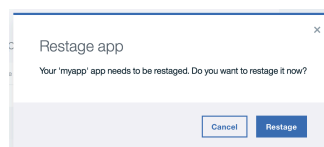

2. You can optionally give the service a custom name or leave it as the one given. Click **Create**.
3. Click on **Connections** in the menu on the left.
4. Click **Create connection** on the right.

5. Click **Connect** next to the Node-RED application you created earlier.

6. IBM Cloud will prompt to restage the application. Click on **Restage**. The application will restart and include the new service credentials in the environment.
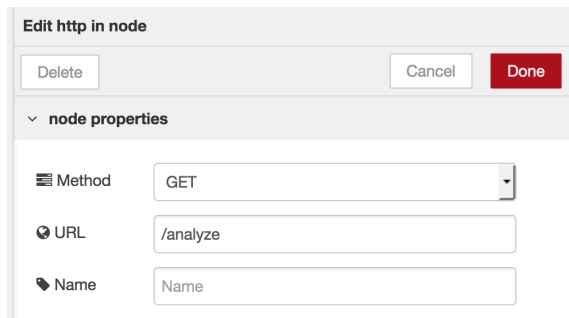

7. When the application has finished restaging, open the Node-RED Flow Editor. If you already have Node-RED open, refresh the page.

Analyze a News Article in Node-RED

The Watson Natural Language Understanding service takes either a body of text or a publicly-accessible URL to content which the service can analyze. In this section, we will analyze a news article that is accessible via an URL. You can also choose to analyze other URLs that contain a body of text. Please refer to the **Add Natural Language Understanding Service in IBM Cloud** section to create and bind the Natural Language Understanding service to your Node-RED application.

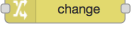1. Add a `http` node as shown below.

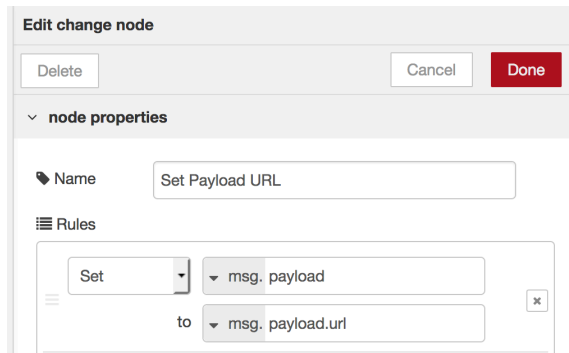   **Edit http in node**

    | Delete | | Cancel | Done |

    ∨ **node properties**

☰ Method	GET
⊕ URL	/analyze
🏷 Name	Name

2. Add a `change` node as shown below. This will take the *url* query parameter and place it in the message *payload* to be passed to the Natural Language Understanding node in the next step.
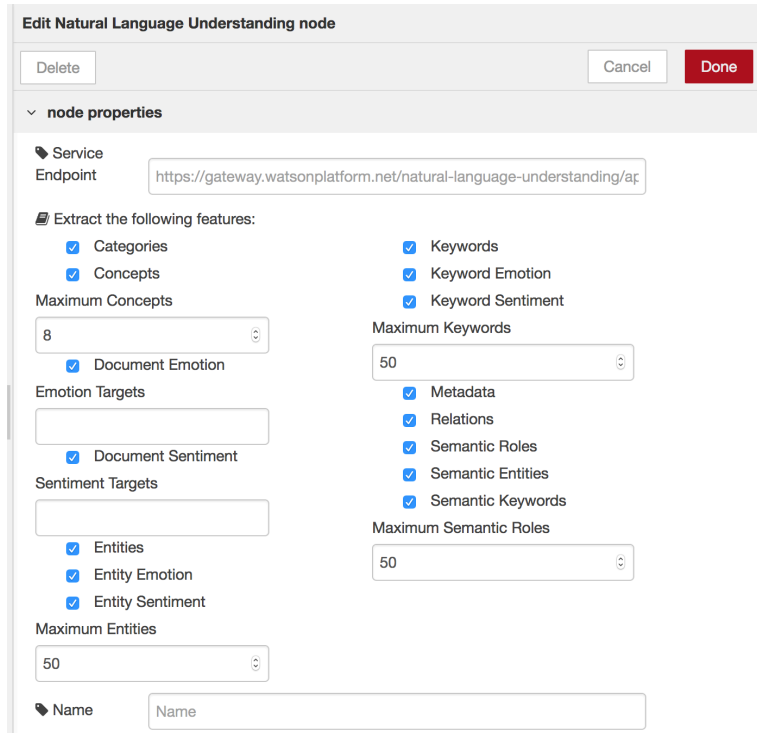
    Edit change node

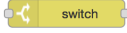 | Delete | | Cancel | Done |

    ∨ **node properties**

🏷 Name	Set Payload URL

    ☰ Rules

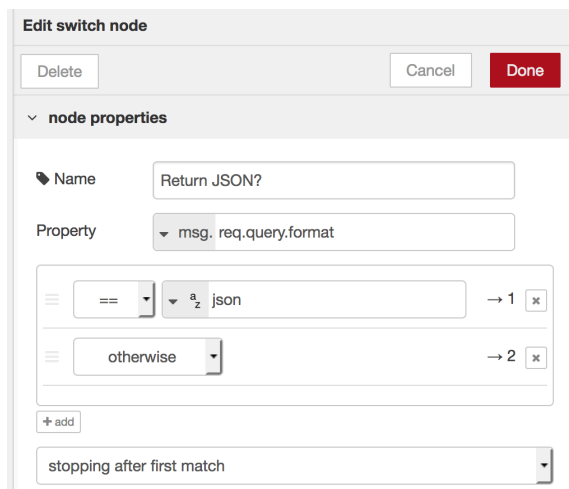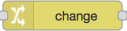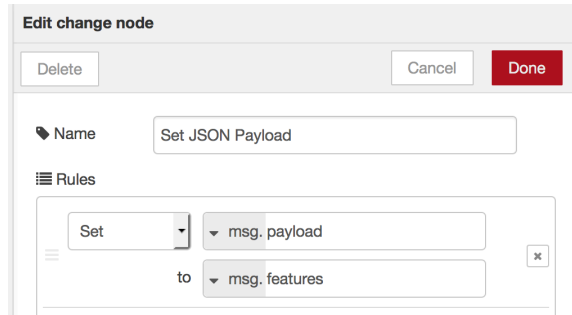 | Set | ▾ msg. payload | ✕ |
    |-----|----------------|---|
    | to | ▾ msg. payload.url | |

3. Add a node. Check the options that you want to be included in the response. You can narrow the results by changing the maximum values and/or by unchecking checkboxes next to each feature. For this lab, we have enabled all the options to showcase many of the insights returned by the Watson service.
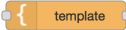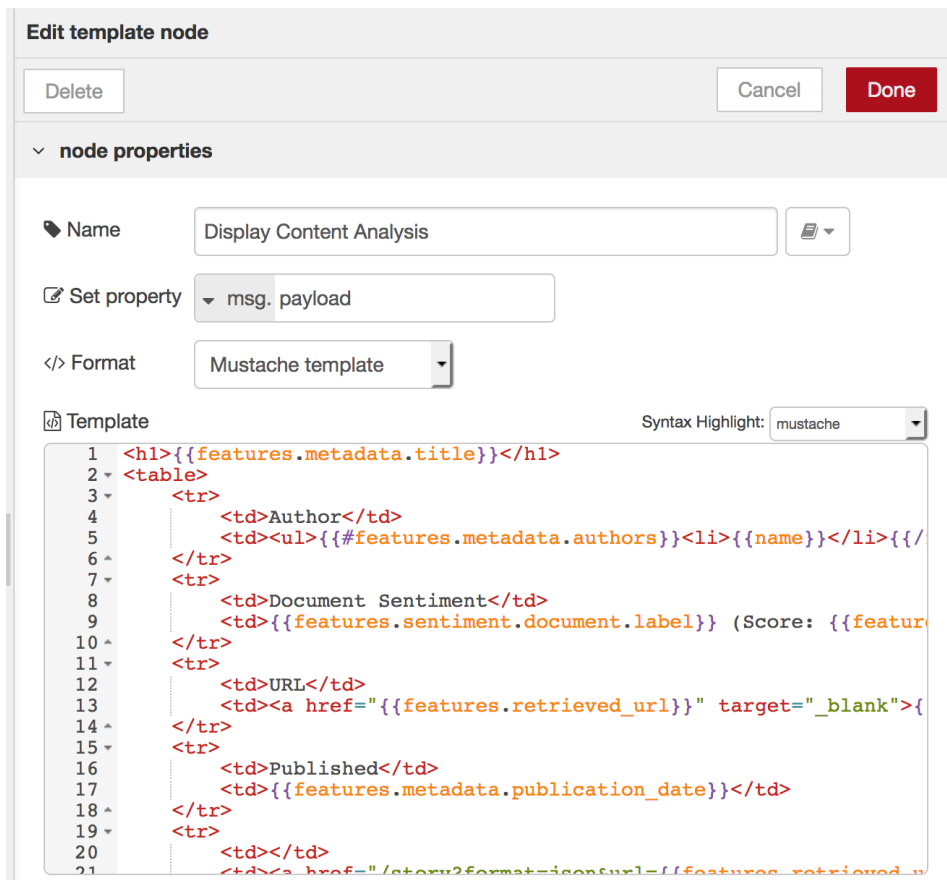

4. To make this application versatile, we'll split the flow so our application will return the results in two formats. The first option will be where the results are returned in JSON format, great for use with applications that can call the web endpoint and consume the JSON. The second option will be a webpage showing the data in a human readable report. Add a node as shown below.

5. For the flow where the JSON should be returned, we can simply return the contents of *msg.features*. Add a change node as shown below. This will move the results from the *features* property to the message *payload*.

Edit change node

| Delete | | Cancel | Done |

🏷 **Name**   `Set JSON Payload`

☰ **Rules**

| Set | ▾ msg. payload |
| to | ▾ msg. features |

6. For the flow where a webpage should be returned, add a template node with the HTML from the file at `ibm.biz/BdiBpU`

Edit template node

| Delete | | Cancel | Done |

∨ **node properties**

🏷 **Name**   `Display Content Analysis`

✎ **Set property**   ▾ msg. payload

</> **Format**   `Mustache template` ▾

⊞ **Template**   Syntax Highlight: `mustache` ▾

```
1   <h1>{{features.metadata.title}}</h1>
2 ▾ <table>
3 ▾     <tr>
4           <td>Author</td>
5           <td><ul>{{#features.metadata.authors}}<li>{{name}}</li>{{/
6 ▴     </tr>
7 ▾     <tr>
8           <td>Document Sentiment</td>
9           <td>{{features.sentiment.document.label}} (Score: {{featur
10 ▴    </tr>
11 ▾    <tr>
12          <td>URL</td>
13          <td><a href="{{features.retrieved_url}}" target="_blank">{
14 ▴    </tr>
15 ▾    <tr>
16          <td>Published</td>
17          <td>{{features.metadata.publication_date}}</td>
18 ▴    </tr>
19 ▾    <tr>
20          <td></td>
21          <td><a href="/story?format=json&url={{features_retrieved_u
```

Get the code:
`ibm.biz/BdiBpU`

7. Add a http response node. Connect the nodes together as shown below.

[get] /analyze → Set Payload URL → Natural Language Understanding → Return JSON? → Set JSON Payload → http
                                                                                    → Display Content Analysis

8. Click on the red **Deploy ▾** button in the top-right corner of the screen to save and deploy your changes.
9. Visit a news website and copy the URL of a publicly accessible news article. Make sure the content isn't behind an authentication wall where you need to sign in to access the content.

10. Open a browser tab and visit your application's endpoint, passing in the URL to the content:

```
http://<<MY-APP>>.mybluemix.net/analyze?url=<<URL-TO-STORY>>
```

- Replace <<MY-APP>> with the host of the Node-RED application you chose.
- Replace <<URL-TO-STORY>> with the URL of the content.

11. Depending on the content located at the URL, you may see a list of attributes including concepts, entities, keywords, categories, sentiment, emotion, relations, semantic roles and more mentioned within the text.

12. To see the JSON representation of the content insert *format=json* in the URL query string:

```
http://<<MY-APP>>.mybluemix.net/analyze?format=json&url=<<URL-TO-STORY>>
```


13. Return to Step #3 and experiment by disabling some of the features to see how the results change. Try analyzing other URLs and see what results are returned.