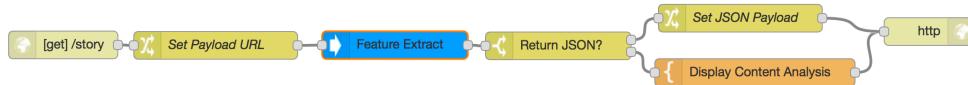
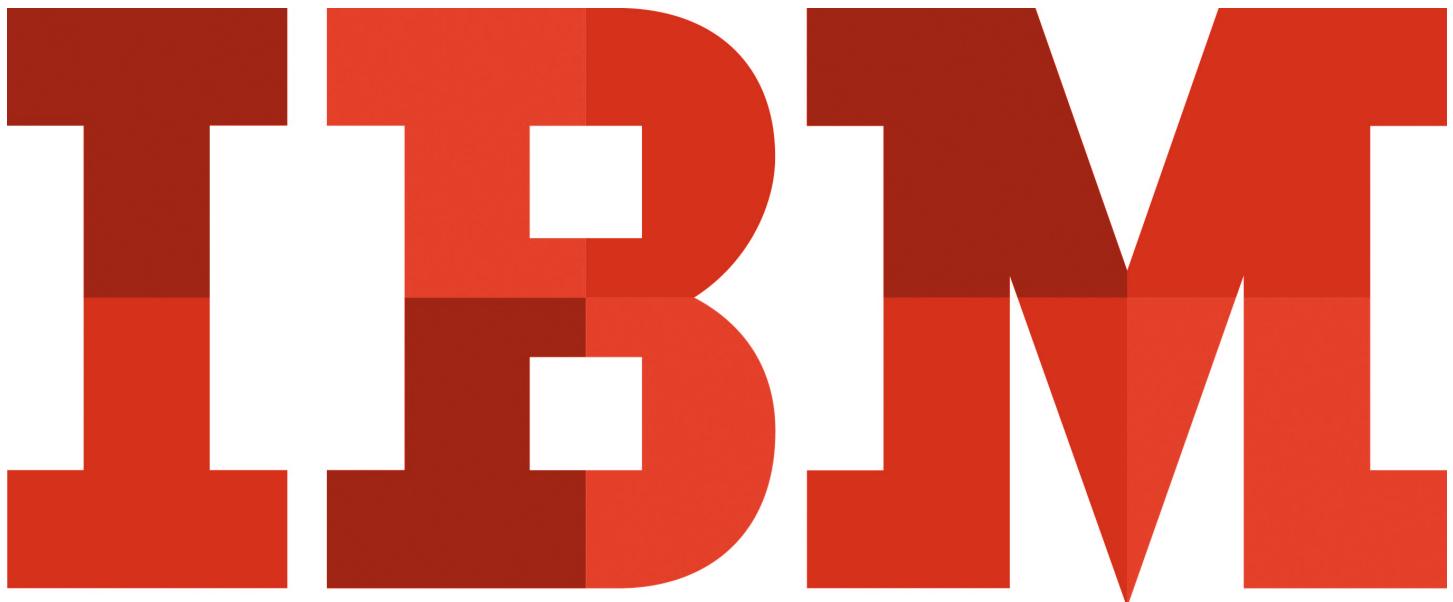


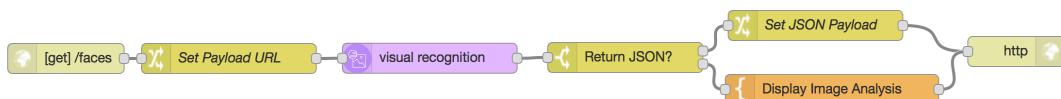
AlchemyAPI in Node-RED

Hands-On Lab

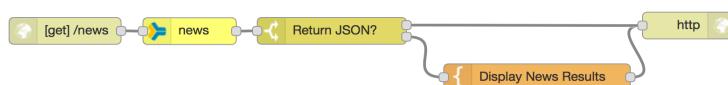
JeanCarl Bisson | jbisson@us.ibm.com | [@dothewww](https://twitter.com/dothewww)



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



Extract gender, name, categories, and age-range of faces in an image
(see *Analyzing Faces in an Image*)



Search for news articles
(see *Search for News with AlchemyData News*)



A digital copy of this lab and code snippets can be found at:
<http://ibm.biz/node-red-alchemyapi>

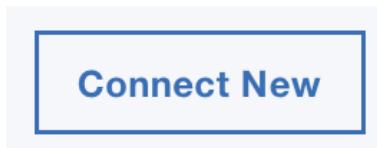


Add Alchemy API Service in IBM Bluemix

The AlchemyAPI (alchemy.ai) offers two services: Language and Data news. In this tutorial, we'll use the Language API to analyze content from a webpage URL. The Language API returns keyword, entities, and concepts, sentiment, and other information about the content being analyzed. We'll also use the DataNews service to search for news articles. This tutorial uses the Node-RED boilerplate in IBM Bluemix with the AlchemyAPI service.

To get started using the Alchemy API, you'll need to create an AlchemyAPI service to get an API key.

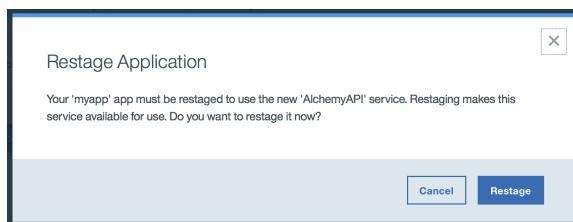
1. Go to the Connections tab under the application overview for your Node-RED application in the IBM Bluemix dashboard and click on **Connect New**.



2. Click the **AlchemyAPI** tile under the Watson section. Click on **Create**.

A screenshot of the IBM Bluemix service catalog. On the left, there's a sidebar with categories like All Categories, Apps, Services, and Watson. Under Watson, "AlchemyAPI" is highlighted. The main area shows various Watson services with their icons and brief descriptions. The AlchemyAPI service is highlighted with a light gray background.

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

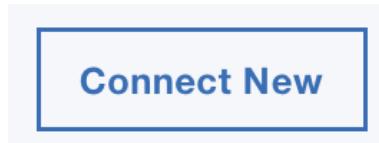


4. When the application has finished restaging, open the Node-RED Flow Editor. If you already have Node-RED open, refresh the page.

Add Visual Recognition Service in IBM Bluemix

Formerly known as the AlchemyVision API, the capabilities of this API were merged into the Watson Visual Recognition service. We'll also use the Visual Recognition API to locate faces and attributes about people in an image. To get started using the Watson Visual Recognition service, you'll need to create a Visual Recognition service to get service credentials.

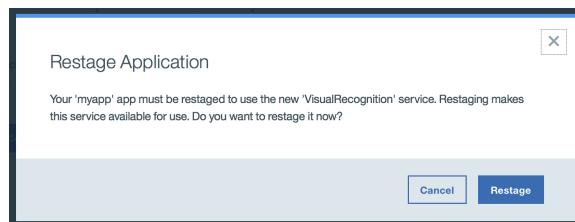
1. Go to the Connections tab under the application overview for your Node-RED application in the IBM Bluemix dashboard and click on **Connect New**.



2. Click the **Visual Recognition** tile under the Watson section. Click on **Create**.

A screenshot of the IBM Bluemix service catalog. On the left, there's a sidebar with categories: All Categories, Apps (Mobile), Services (Watson selected), Data & Analytics (Watson expanded), Internet of Things, APIs, Storage, Security, DevOps, Application Services, and Integrate. The main area shows a grid of Watson services. The "Visual Recognition" service is visible in the bottom right corner of the grid. Each service has a thumbnail icon, a name, a brief description, and an "IBM" badge.

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

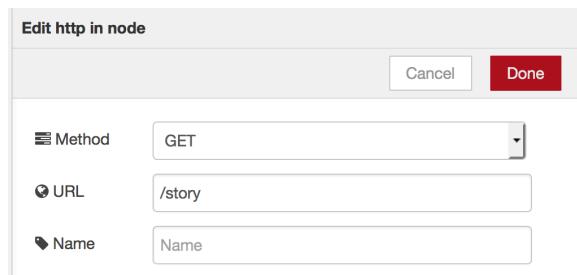


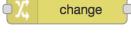
4. When the application has finished restaging, open the Node-RED Flow Editor. If you already have Node-RED open, refresh the page.

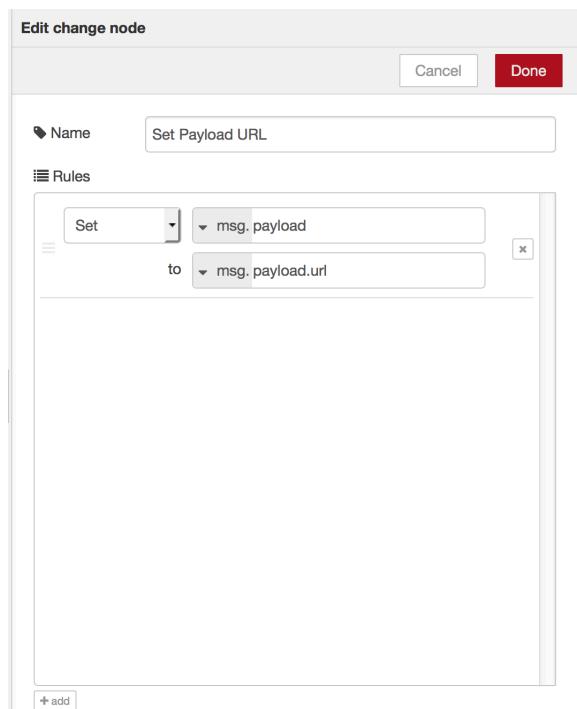
Analyze a News Article in Node-RED

The AlchemyLanguage API analyzes unstructured text and provides a handful of attributes including keywords, entities, concepts, taxonomy, document sentiment, author, publication date, title and more. You can either provide a URL where the content resides, or a body of text to analyze. In this section, we will analyze a news article accessible via an URL. Please refer to the **Add Alchemy API Service in IBM Bluemix** section to create and bind the AlchemyAPI service to your application.

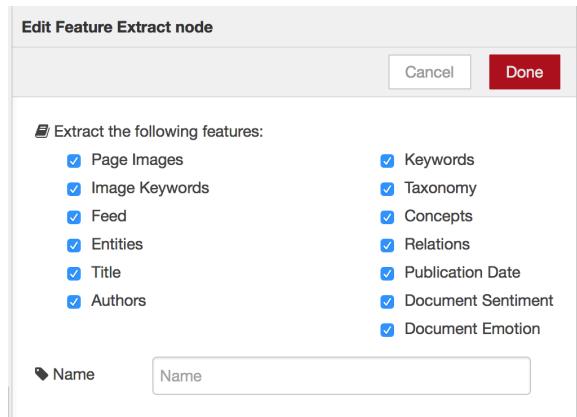
1. Add a  node as shown below.



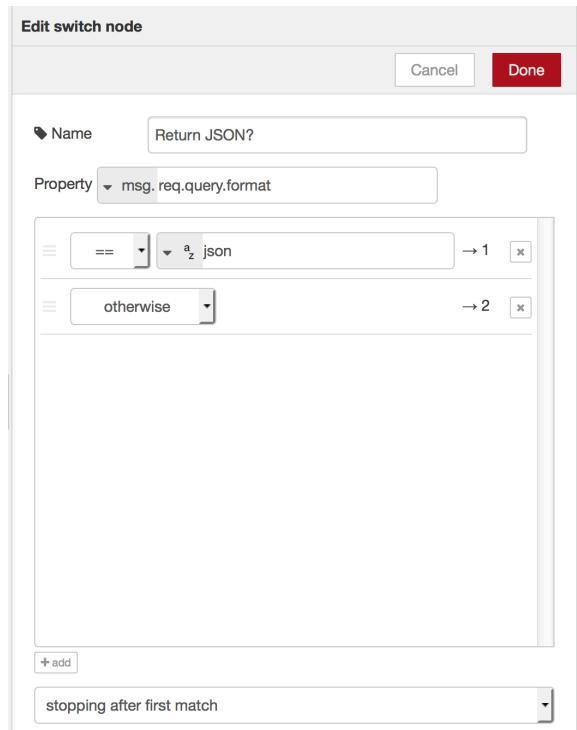
2. Add a  node as shown below. This will take the *url* query parameter and place it in the message *payload* to be passed to the Feature Extract node.

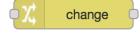


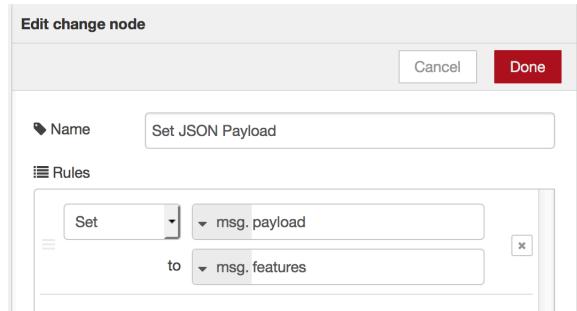
3. Add a  node. Check the options that you want to be included in the response.



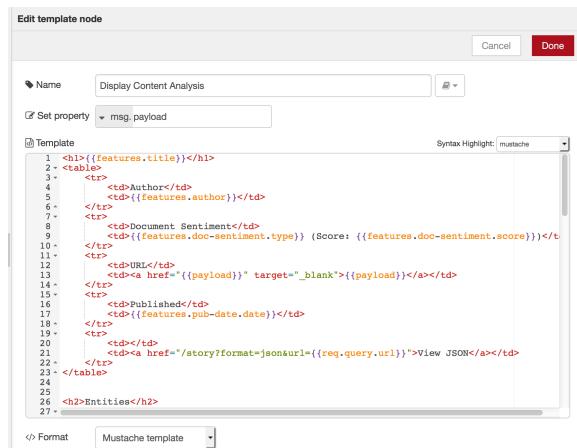
4. To make this application versatile, we'll split the flow so our application will return the results in two formats. The first option will be where the results are returned in JSON format, great for use in applications that can call the web endpoint and consume the JSON. The second option will be a 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`.

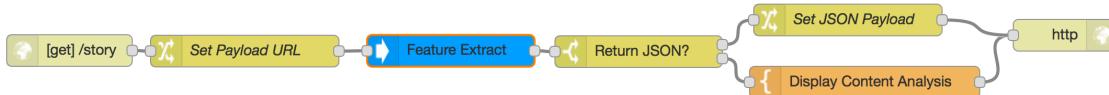


6. For the flow where a webpage should be returned, add a  template node with the HTML in the file named `1-display-content-analysis.html`.



Get the code:
ibm.biz/Bd48EZ

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



8. Click on the red  Deploy button in the upper-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 have 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/story?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 keywords, entities, concepts, taxonomy, document sentiment, author, publication date, title and more mentioned within the text.

The screenshot shows the AlchemyAPI interface with the following details:

- Title:** IBM launches quantum computing as a cloud service
- Author:** Ron Miller
- Document Sentiment positive (Score: 0.237881)**
- URL:** <http://rechnerich.com/2016/05/03/ibm-brings-experimental-quantum-computing-to-the-cloud/>
- Published:** 2016/05/03T000000
- View JSON**
- Entities** table:

Entity	Type	Relevance /# Occurrences
IBM	Company	0.884866 [1]
Quantum computing	FieldTerminology	0.655543 [3]
Experimental Quantum Computing Group	PrintMedia	0.484536 [1]
Quantum Experience	Technology	0.469095 [1]
IBM Research	Company	0.311148 [1]
Jerry Chow	Person	0.253692 [2]
Charles King	Person	0.247544 [2]
Earl Joseph	Person	0.222398 [3]
D-Wave Systems	Company	0.216912 [1]
programming language	FieldTerminology	0.202103 [1]
New York State	City	0.202032 [1]
programmer	JobTitle	0.201049 [1]
Google	Company	0.195871 [1]
NASA	Organization	0.194807 [1]
UCSB	Organization	0.192451 [1]
TechCrunch	Company	0.191462 [1]
Moore	Person	0.157148 [1]
PandIT, Inc.	Company	0.108071 [1]
IDC	Company	0.175817 [1]
Artificial Intelligence	FieldTerminology	0.173867 [1]

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

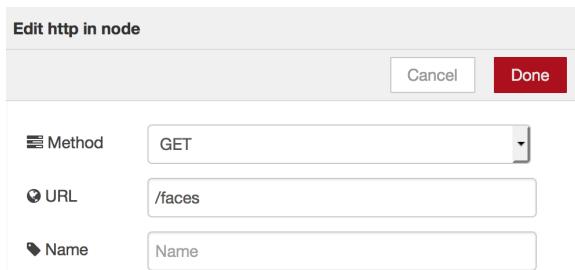
The screenshot shows the JSON representation of the content, which is a large object containing various entities and their properties. Key parts of the JSON include:

- pageImage:** "https://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "image-ix": [{"text": "MO_TOGS", "score": "1"}], "entity": [{"type": "Company", "relevance": "0.884866", "count": "1", "text": "IBM", "disambiguated": "1"}, {"type": "FieldTerminology", "relevance": "0.655543", "count": "3", "text": "Quantum computing", "disambiguated": "1"}, {"type": "PrintMedia", "relevance": "0.484536", "count": "1", "text": "ProgrammingLanguageDeveloper", "name": "IBM", "website": "http://www.ibm.com/", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "IBM", "labelType": "Company", "labelRelevance": "0.884866", "labelCount": "1"}, {"type": "Technology", "relevance": "0.469095", "count": "1", "text": "Experimental Quantum Computing Group", "disambiguated": "1"}, {"type": "FieldTerminology", "relevance": "0.469095", "count": "1", "text": "Quantum computing", "disambiguated": "1"}, {"type": "PrintMedia", "relevance": "0.469095", "count": "1", "text": "Quantum computing", "name": "Experimental Quantum Computing Group", "website": "http://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "Experimental Quantum Computing Group", "labelType": "PrintMedia", "labelRelevance": "0.469095", "labelCount": "1"}, {"type": "Person", "relevance": "0.253692", "count": "2", "text": "Jerry Chow", "disambiguated": "1"}, {"type": "JobTitle", "relevance": "0.253692", "count": "2", "text": "Jerry Chow", "name": "Jerry Chow", "website": "http://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "Jerry Chow", "labelType": "Person", "labelRelevance": "0.253692", "labelCount": "2"}, {"type": "Person", "relevance": "0.247544", "count": "2", "text": "Charles King", "disambiguated": "1"}, {"type": "JobTitle", "relevance": "0.247544", "count": "2", "text": "Charles King", "name": "Charles King", "website": "http://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "Charles King", "labelType": "Person", "labelRelevance": "0.247544", "labelCount": "2"}, {"type": "Company", "relevance": "0.216912", "count": "1", "text": "D-Wave Systems", "disambiguated": "1"}, {"type": "FieldTerminology", "relevance": "0.216912", "count": "1", "text": "D-Wave Systems", "disambiguated": "1"}, {"type": "PrintMedia", "relevance": "0.216912", "count": "1", "text": "D-Wave Systems", "name": "D-Wave Systems", "website": "http://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "D-Wave Systems", "labelType": "PrintMedia", "labelRelevance": "0.216912", "labelCount": "1"}, {"type": "City", "relevance": "0.202032", "count": "1", "text": "New York State", "disambiguated": "1"}, {"type": "JobTitle", "relevance": "0.202032", "count": "1", "text": "New York State", "name": "New York State", "website": "http://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "New York State", "labelType": "City", "labelRelevance": "0.202032", "labelCount": "1"}, {"type": "Person", "relevance": "0.195871", "count": "1", "text": "Google", "disambiguated": "1"}, {"type": "Organization", "relevance": "0.195871", "count": "1", "text": "Google", "name": "Google", "website": "http://rechnerichuse2011.files.wordpress.com/2015/05/quantum-dash_1.jpg?w=80&h=387", "dipedia": "http://dipedia.org/resource/nm/03e8", "fb": "http://rdf.freebase.com/n/m.03e8", "openyo": "http://sw.openyo.org/concept/nm/03e8", "label": "Google", "labelType": "Organization", "labelRelevance": "0.195871", "labelCount": "1"}, {"type": "Organization", "relevance": "0.194807", "count": "1", "text": "NASA", "disambiguated": "1"}, {"type": "Organization", "relevance": "0.192451", "count": "1", "text": "UCSB", "disambiguated": "1"}, {"type": "Company", "relevance": "0.191462", "count": "1", "text": "TechCrunch", "disambiguated": "1"}, {"type": "Person", "relevance": "0.157148", "count": "1", "text": "Moore", "disambiguated": "1"}, {"type": "Company", "relevance": "0.108071", "count": "1", "text": "PandIT, Inc.", "disambiguated": "1"}, {"type": "Company", "relevance": "0.175817", "count": "1", "text": "IDC", "disambiguated": "1"}, {"type": "FieldTerminology", "relevance": "0.173867", "count": "1", "text": "Artificial Intelligence", "disambiguated": "1"}]

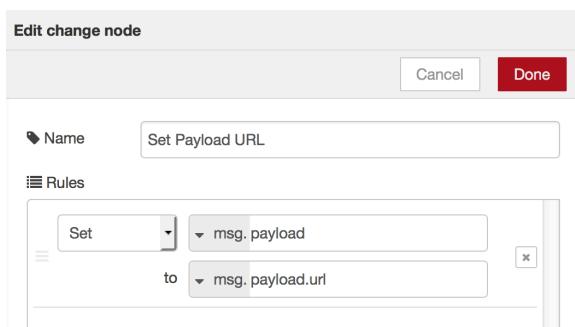
Analyzing Faces in an Image

The Visual Recognition API analyzes images and provides a handful of attributes including gender, age-range, and if known, the name and type, of each person in the given image. You can either provide a URL or the binary data of the image to analyze. In this section, we will locate faces in an image, provide an age-range, gender, and if known, the name and type of the person. Please refer to the **Add Visual Recognition Service in IBM Bluemix** section to create and bind the Watson Visual Recognition service to your application.

1. Add a  node as shown below.

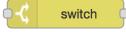


2. Add a  node as shown below. This will take the *url* query parameter and place it in the message *payload* to be passed to the Image Analysis node.

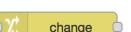


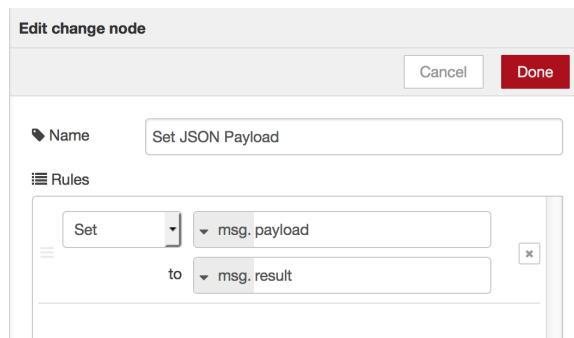
3. Add a  node. Select **Detect Faces** from the Detect menu as shown below.



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 in 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 JSON should be returned, we can simply return the contents of *msg.result*. Add a  node as shown below. This will move the results from the *result* property to the message *payload*.



6. For the flow where a webpage should be returned, add a  node with the HTML in the file named 2-display-image-analysis.html.



Edit template node

Name: Display Image Analysis

Set property: msg.payload

Template:

```

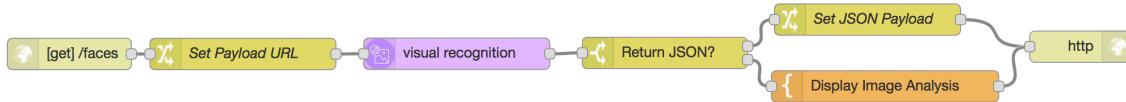
1  {{#result.images}}
2   
3 
4  {{#faces}}
5   <table border="1">
6     <tr>
7       <td colspan="2">{{identity.name}}</td>
8     </tr>
9     <tr>
10      <td>{{age}}</td>
11      <td>{{age.min}} - {{age.max}} {{age.score}}</td>
12    </tr>
13    <tr>
14      <td>{{gender}}</td>
15      <td>{{gender.gender}} {{gender.score}}</td>
16    </tr>
17    <tr>
18      <td>{{position}}</td>
19      <td>{{face_location.left}}, {{face_location.top}}) {{face_location.width}} x {{face_location.height}}</td>
20    </tr>
21    <tr>
22      <td>{{type}}</td>
23      <td>{{identity.type_hierarchy}}</td>
24    </tr>
25  </table>
26</tr>
27

```

Format: Mustache template

Get the code:
ibm.biz/Bd48Ey

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

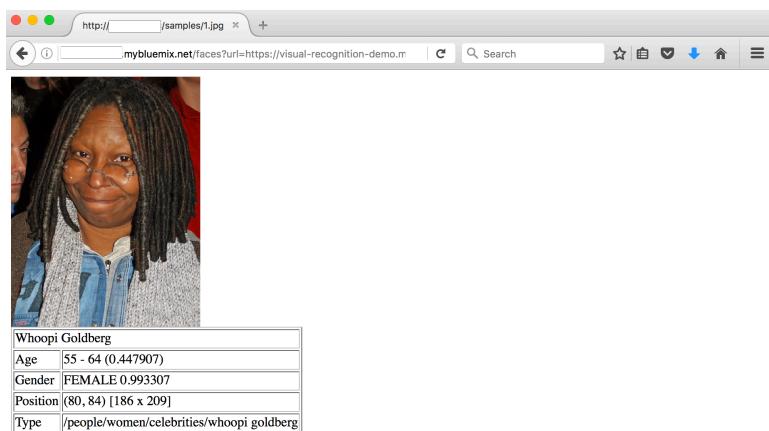


8. Click on the red  button in the upper-right of the screen to save and deploy your changes.
 9. Locate an image of at least one person and copy the URL of the image. Make sure the content isn't behind an authentication wall where you have to sign in to access the content.
 10. Open a browser tab and visit your application's endpoint, passing in the URL to the image:

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

- Replace <<MY-APP>> with the hostname of the Node-RED application you chose.
- Replace <<URL-TO-IMAGE>> with the URL of the image.

11. Depending on the content located at the URL, you may see a list of attributes for each face.



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

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



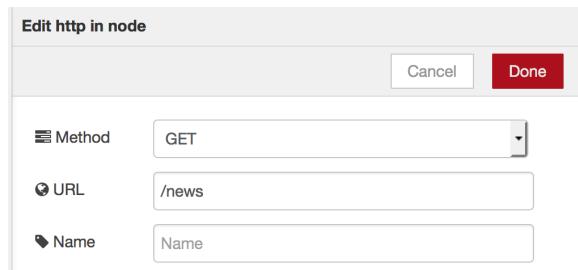
A screenshot of a web browser window. The address bar shows the URL: `http://i/samples/1.jpg`. The page content displays a JSON object representing a face detection result. The JSON structure is as follows:

```
{"images": [{"faces": [{"age": {"max": 64, "min": 55, "score": 0.447907}, "face_location": {"height": 209, "left": 80, "top": 84, "width": 186}, "gender": {"gender": "FEMALE", "score": 0.993307}, "identity": {"name": "Whoopi Goldberg", "score": 0.952574}, "type_hierarchy": "/people/women/celebrities/whoopi goldberg"}]}, {"resolved_url": "https://visual-recognition-demo.mybluemix.net/images/samples/1.jpg"}, {"source_url": "https://visual-recognition-demo.mybluemix.net/images/samples/1.jpg"}], "images_processed": 1}
```

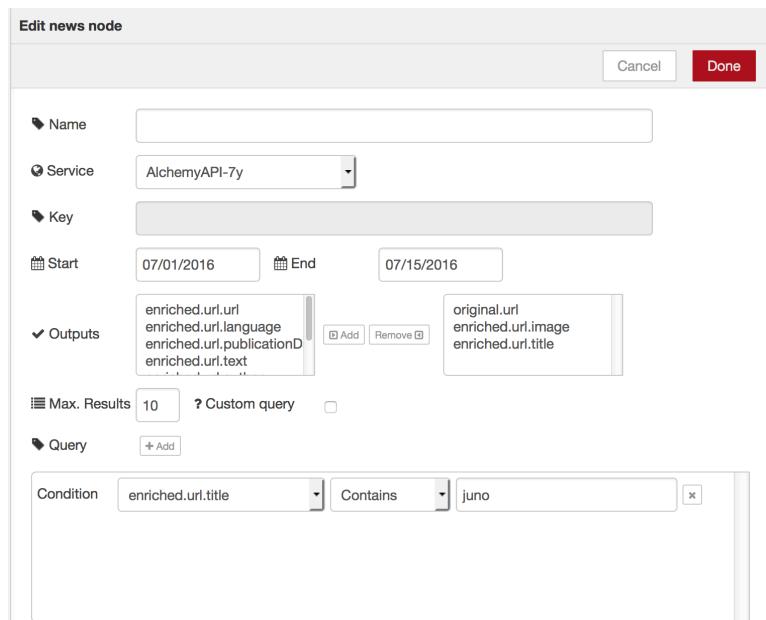
Search for News with AlchemyData News

The AlchemyData News API analyzes and categorizes over a quarter-million news articles everyday. In this section, we'll use the AlchemyData News API service to retrieve news articles on NASA's Juno mission. Please refer to the [Add Alchemy API Service in IBM Bluemix](#) section to create and bind the AlchemyAPI service to your application.

1. Add a  node as shown below.

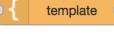


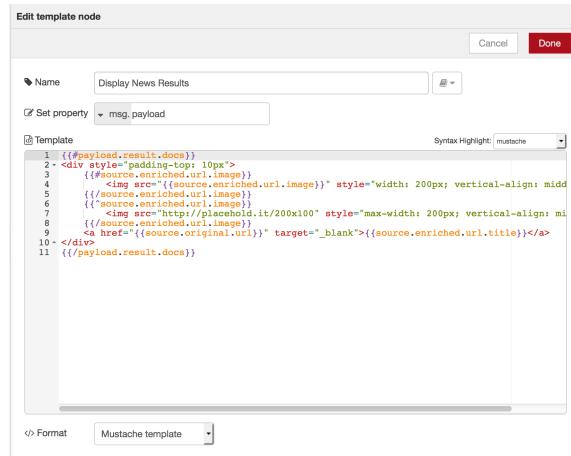
2. Add a  node as shown below.



3. 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 in 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.

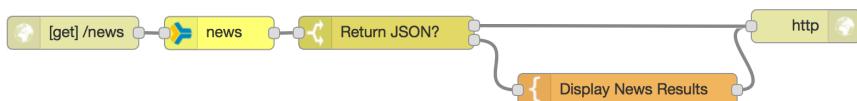


4. For the flow where a webpage should be returned, add a  node with the HTML in the file named 3-display-news-results.html.



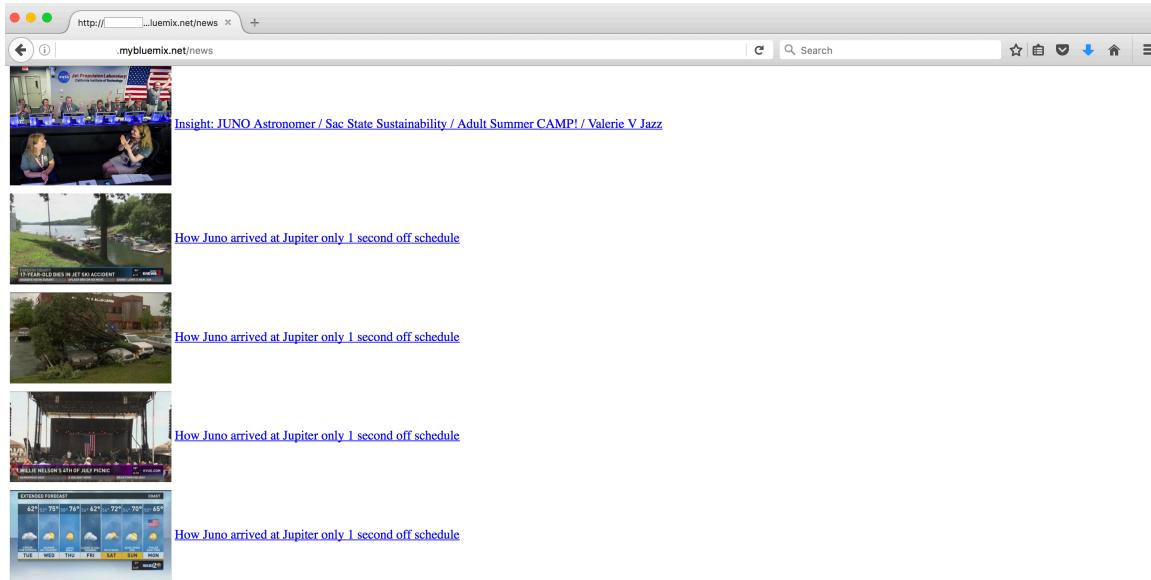
Get the code:
ibm.biz/BdrDtw

5. Add a  node. Connect the nodes together as shown below.



6. Click on the red  Deploy button in the upper-right of the screen to save and deploy your changes.
Open a browser tab and visit your application's endpoint:

<http://<<MY-APP>>.mybluemix.net/news>



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

<http://<<MY-APP>>.mybluemix.net/news?format=json>

