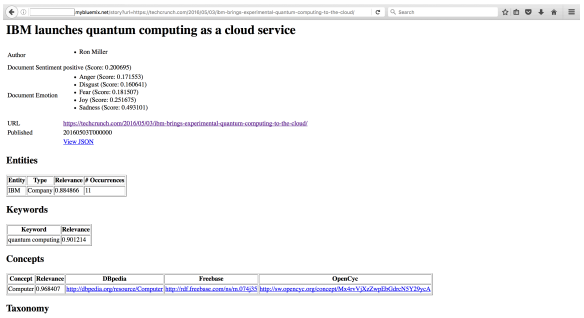
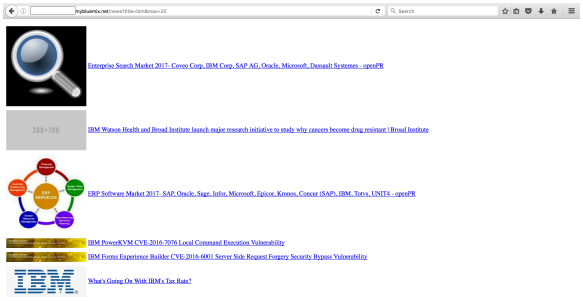
AlchemyAPI in Node.js

Hands-On Lab

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



Analyze a News Article (pg. 6)



Search for News with AlchemyData News (pg. 10)

A digital copy of this lab and code snippets can be found at:
`http://ibm.biz/nodejs-alchemyapi`

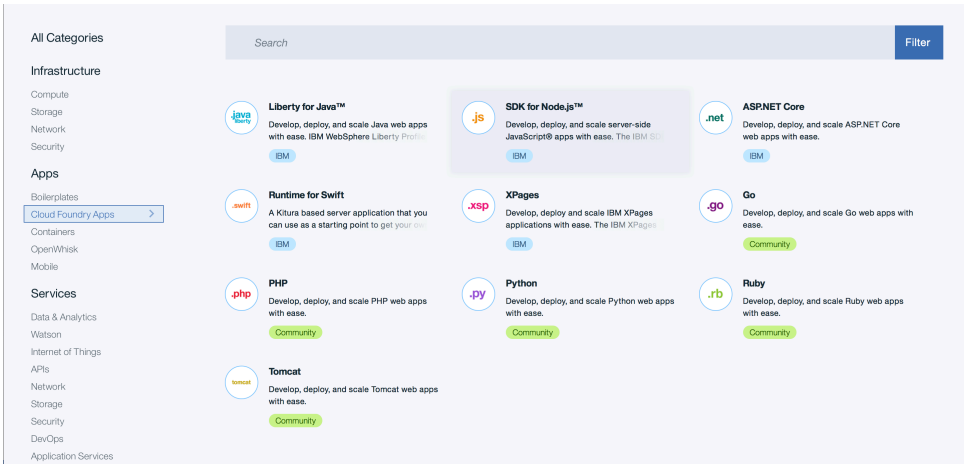
Setup Environment in IBM Bluemix

AlchemyAPI uses natural language processing technology and machine learning algorithms to extract semantic meta-data from content, such as information on people, places, companies, topics, facts, relationships, authors, and languages.

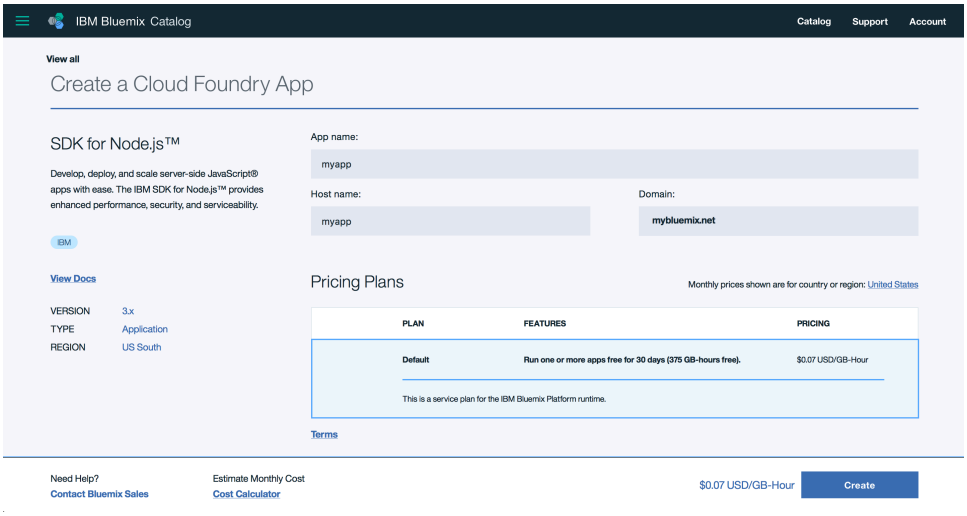
The AlchemyAPI (alchemy.ai) offers two services: Language and Data News. In this section, we'll use the Language API to analyze content from a webpage URL. The Language API returns keyword, entities, concepts, sentiment, emotions, and other information about the content being analyzed. We'll also use the Data News API to search for news content with these attributes. This tutorial uses the SDK for Node.js runtime in IBM Bluemix and the Watson AlchemyAPI service.

To get started, first create a Node.js application in IBM Bluemix. Visit `bluemix.net` and click on the **Catalog** link.

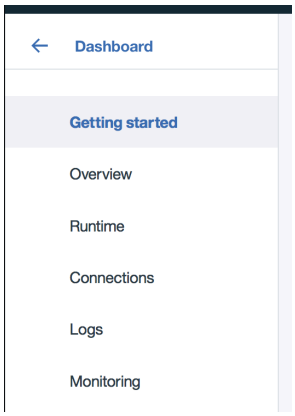
1. Select **SDK for Node.js** under the section titled `Cloud Foundry Apps`.



2. On the next screen, enter a name for your application in the field labeled `App name`. This name is for your convenience. Enter a host name, which will be the first part of the URL for your application. If you choose myapp, the application URL will be myapp.mybluemix.net. There can only be one application with the same host name across all of IBM Bluemix. Click on **Create** to create the application.

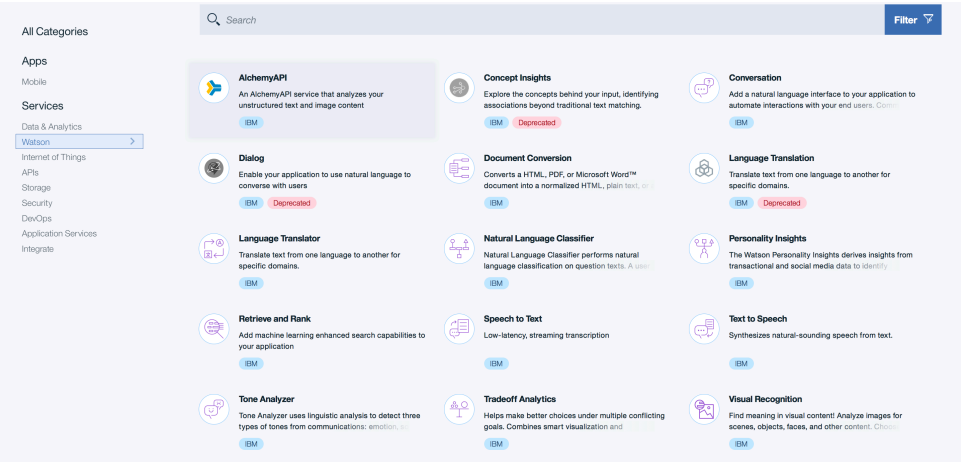


3. On the next screen, select **Connections** from the left sidebar.

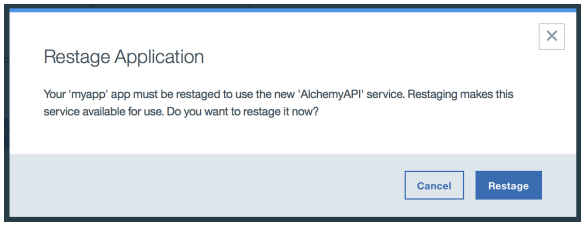


4. In order to use an IBM Watson service, we need to create and bind the applicable Watson service to the IBM Bluemix application. In this lab, we'll use the AlchemyAPI service. Click on the **Connect New** button on the right of the page.

5. Click on the **AlchemyAPI** tile under the Watson section. You can leave the fields as the default values. Click on **Create**.



6. IBM Bluemix will prompt to restage the application. Click on **Restage**. The application will restart.

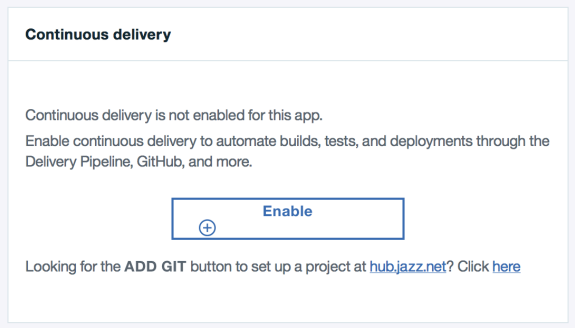


7. When the application has finished restaging, the Node.js application environment will contain the Watson service credentials.

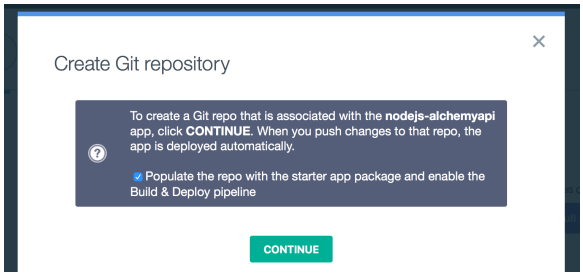
Create Git Repository/Import Project Files

In this section, we'll create a project that can utilize the web-based editor in the IBM Bluemix DevOps Services. This step needs to be completed only once before proceeding through the sections titled **Analyze a News Article** and **Search for News with AlchemyData News**.

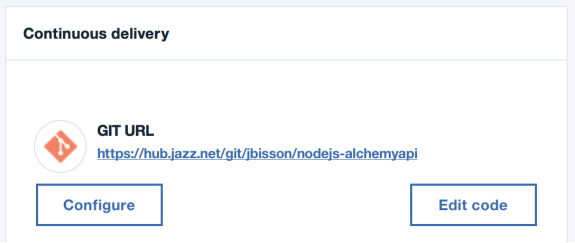
1. Open the application overview for the Node.js application. In the bottom right corner in the box titled **Continuous delivery**, click on the last link, `Click here`, to create a project at hub.jazz.net.



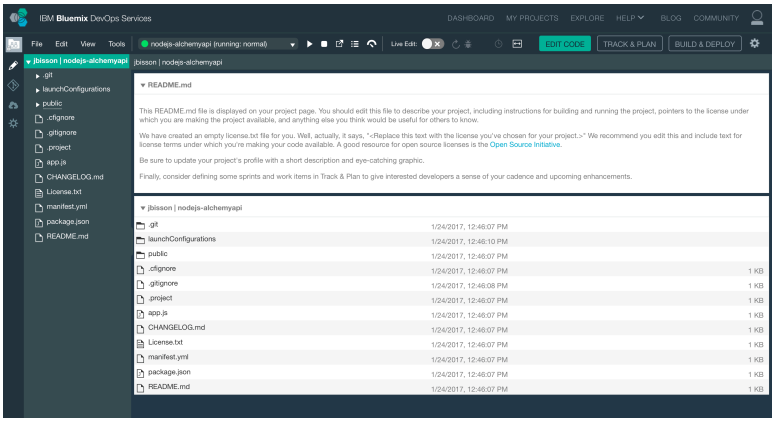
2. Ensure the checkbox is checked and click on **Continue**.



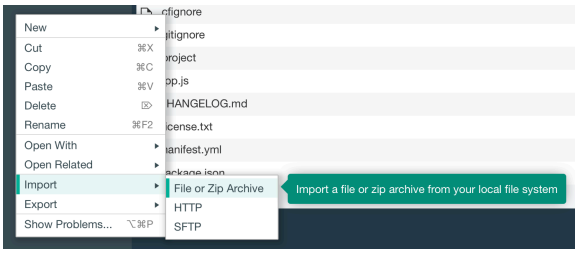
3. You will be returned to the application overview. Click on **Edit code**.



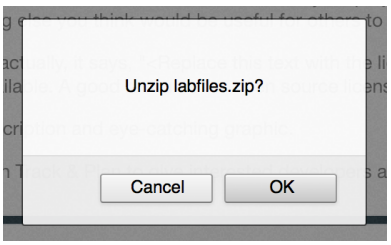
4. You will be taken into the IBM Bluemix DevOps Services tooling. You can edit code in the browser, commit changes to the Git repository, and setup testing and deployment pipeline options when code changes are made. The default pipeline automatically deploys new changes that are committed to the Git repository.



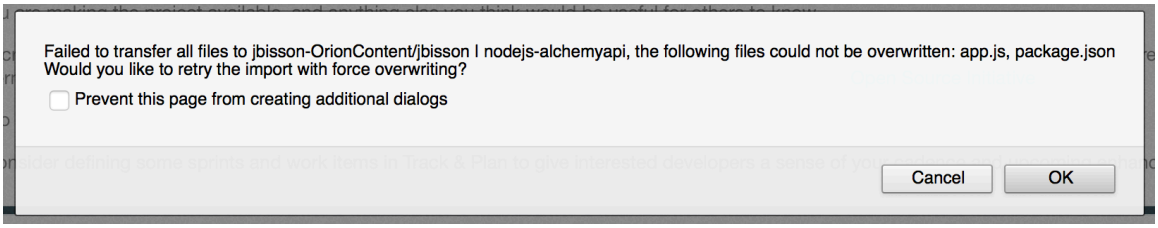
5. Let's import a couple of files that the following sections need. Download the Zip file from `ibm.biz/nodejs-alchemyapi-labfiles`
6. Under the file listing in the left sidebar, right click to display the context menu. Select **Import** -> **File or Zip Archive**. Select the downloaded Zip file.



7. When prompted to unzip the file, click **OK**.



8. Some of the files in the ZIP file already exist in the project. Click **OK** to force overwriting.



9. Edit the **package.json** file and add the `mustache` and `watson-developer-cloud` npm packages.


```
"dependencies": {
    "express": "4.13.x",
    "cfenv": "1.0.x",
    "mustache": "2.3.x",
    "watson-developer-cloud": "2.15.x"
```

Analyze a News Article

The AlchemyLanguage API analyzes unstructured text and provides a handful of attributes including keywords, entities, concepts, taxonomy, sentiment, emotion, 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.

The completed code snippet is shown below for reference.

Listing 1: Node.js code in app.js to call the AlchemyLanguage API

```
app.get("/story", function (req, res) {
  var alchemyLanguage = new watson.AlchemyLanguageV1({
    // api_key: '<api-key>'
  });

  var features = [
    "entities",
    "keywords",
    "title",
    "authors",
    "taxonomy",
    "concepts",
    "relations",
    "pub-date",
    "doc-sentiment",
    "doc-emotion",
    "page-image",
    "feeds"
  ];

  var parameters = {
    extract: features.join(","),
    sentiment: 1,
    emotion: 1,
    maxRetrieve: 1,
    url: req.query.url
  };

  alchemyLanguage.combined(parameters, function (err, response) {
    if (err) {
      res.send(err);
    } else {
      if (req.query.format == "json") {
        res.send(response);
      } else {
        response.url = req.query.url;
        res.send(mustache.render(templates.story, response));
      }
    }
  });
});
```

Copy and paste the code at:

ibm.biz/nodejs-alchemyapi-story

Listing 2: HTML in the file named template/story.html to display results from the AlchemyLanguage API

```
<h1>{{title}}</h1>
<table>
  <tr>
    <td>Author</td>
    <td>
      <ul>
        {{#authors.names}}
          <li>{{.}}</li>
        {{/authors.names}}
      </ul>
    </td>
  </tr>
  <tr>
    <td>Document Emotion</td>
    <td>
      <ul>
        <li>Anger (Score: {{docEmotions.anger}})</li>
        <li>Disgust (Score: {{docEmotions.disgust}})</li>
        <li>Fear (Score: {{docEmotions.fear}})</li>
```

```
        <li>Joy (Score: {{docEmotions.joy}})</li>
        <li>Sadness (Score: {{docEmotions.sadness}})</li>
      </ul>
    </td>
  </tr>
  <tr>
    <td>URL</td>
    <td><a href="{{href}}" target="_blank">{{href}}</a></td>
  </tr>
  <tr>
    <td>Published</td>
    <td>{{publicationDate.date}}</td>
  </tr>
  <tr>
    <td></td>
    <td><a href="/story?format=json&url={{href}}">View JSON</a></td>
  </tr>
</table>

<h2>Entities</h2>
<table border="1">
  <tr>
    <th>Entity</th>
    <th>Type</th>
    <th>Relevance</th>
    <th># Occurrences</th>
  </tr>
{{#entities}}
  <tr>
    <td>{{text}}</td>
    <td>{{type}}</td>
    <td>{{relevance}}</td>
    <td>{{count}}</td>
  </tr>
{{/entities}}
</table>

<h2>Keywords</h2>
<table border="1">
  <tr>
    <th>Keyword</th>
    <th>Relevance</th>
  </tr>
{{#keywords}}
  <tr>
    <td>{{text}}</td>
    <td>{{relevance}}</td>
  </tr>
{{/keywords}}
</table>

<h2>Concepts</h2>
<table border="1">
  <tr>
    <th>Concept</th>
    <th>Relevance</th>
    <th>DBpedia</th>
    <th>Freebase</th>
    <th>OpenCyc</th>
  </tr>
{{#concepts}}
  <tr>
    <td>{{text}}</td>
    <td>{{relevance}}</td>
    <td><a href="{{dbpedia}}" target="_blank">{{dbpedia}}</a></td>
    <td><a href="{{freebase}}" target="_blank">{{freebase}}</a></td>
    <td><a href="{{opencyc}}" target="_blank">{{opencyc}}</a></td>
  </tr>
{{/concepts}}
</table>

<h2>Taxonomy</h2>
<table border="1">
<tr>
  <th>Label</th>
  <th>Score</th>
  <th>Confident</th>
</tr>
```

```
 98. {{#taxonomy}}
 99.   <tr>
100.     <td>{{label}}</td>
101.     <td>{{score}}</td>
102.     <td>{{confident}}</td>
103.   </tr>
104. {{/taxonomy}}
105. </table>
106.
107. <h2>Image</h2>
108. <img src="{{image}}" />
```

1. Add the code in *Listing 1* to the file named `app.js`. You can also find the code snippet at `ibm.biz/nodejs-alchemyapi-story`. Here's a brief introduction of what this code does:
    - On line 1, we create a Node.js Express endpoint, `/story`. This endpoint can be accessed by taking the application URL and appending `/story`. The application takes one query parameter, `url`, and an optional query parameter, `format`. The query parameter `url` can be any web accessible URL that the AlchemyLanguage API should analyze. To make this application versatile, the query parameter `format` provides two options for the returned result. By setting `format=json` in the query string, the application will return the raw JSON from the AlchemyLanguage API. If omitted, HTML displaying the results, from the template file `template/story.html`, will be returned.
    - Lines 2-4 use the `watson-developer-cloud` NPM package, which abstracts the HTTP request made to the IBM Watson API. On line 3, the Alchemy API key is passed to the library. If left blank, the NPM package will look for the API key in the VCAP services in the IBM Bluemix environment.
    - Lines 6-19 create an array of parameters that AlchemyLanguage API should return. See the table below for more details on each parameter.

Parameter	What AlchemyLanguage API returns
entities	things such as persons, places and organizations in your content
keywords	important topics in your content that can be used to index data, generate tag clouds or for searching
title	title of content
authors	extract author information from news articles or blog posts
taxonomy	group your content before performing further analysis or to track the high-level topics of your documents
concepts	make high-level abstractions by understanding how concepts relate, and identify concepts that aren't necessarily directly referenced in the text
relations	subject, action and object relations within sentences
pub-date	publication date of content
doc-sentiment	the attitude, opinion or feeling toward something, such as a person, organization, product or location.
doc-emotion	confidence scores for anger, disgust, fear, joy, and sadness, indicating the probability that the corresponding emotion is implied by the sample text
page-image	returns the primary image in content
feeds	RSS/ATOM feed links from a webpage or HTML

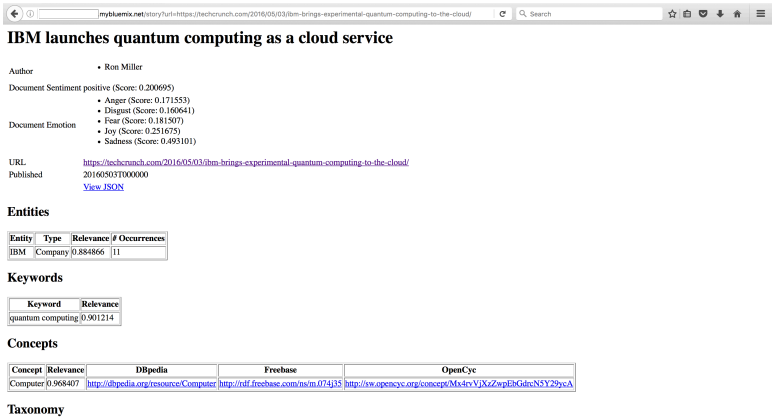
    - These features are concatenated together on line 22 and assigned to the parameter labeled `extract`.
    - The parameters labeled `sentiment` and `emotion` return the sentiment and emotions of keywords and entities if set to 1.
    - Line 26 accepts an URL to the content, or you can change this property name to `text` and pass in a string of content to analyze. Since this example analyzes content located at an URL, the URL is retrieved from the query parameter `url`.
    - Lines 29-40 execute the API call and then handle the result.
    - Lines 30-39 make a call to the AlchemyLanguage API and take one of three actions:
        i. Displays an error if one occurs (lines 30-32)
        ii. Displays the raw JSON result from the API (lines 33-35)
        iii. Displays a rendered HTML page (lines 35-38)
    - The last option uses the mustache templating library to render the HTML located in the `templates/story.html` file.
2. Commit and deploy the changes by following the directions in the section titled **IBM Bluemix DevOps Services** on page 12. When finished with instructions in that section, continue onto Step #3.

3. When the application has been deployed, open a browser tab and visit your application's endpoint, passing in the URL to the webpage of content:

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

   - Replace <<MY-APP>> with the host of the Node.js application you created.
   - Replace <<URL-TO-STORY>> with the URL to the content.

4. Depending on the content located at the URL, you may see a list of attributes including keywords, entities, concepts, taxonomy, document sentiment and emotion, author, publication date, title and more mentioned within the text.



IBM launches quantum computing as a cloud service

Author	• Ron Miller
Document Sentiment	positive (Score: 0.200695)
Document Emotion	• Anger (Score: 0.171553) • Disgust (Score: 0.160641) • Fear (Score: 0.181507) • Joy (Score: 0.251675) • Sadness (Score: 0.493101)
URL	https://techcrunch.com/2016/05/03/ibm-brings-experimental-quantum-computing-to-the-cloud/
Published	20160503T000000
	View JSON

Entities

Entity	Type	Relevance	# Occurrences
IBM	Company	0.884866	11

Keywords

Keyword	Relevance
quantum computing	0.901214

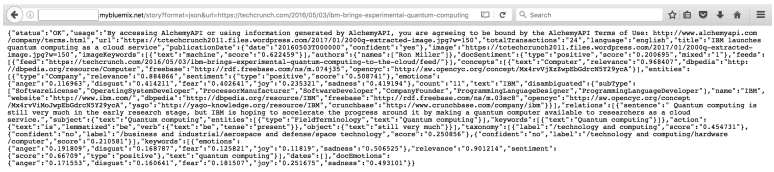
Concepts

Concept	Relevance	DBpedia	Freebase	OpenCyc
Computer	0.968407	http://dbpedia.org/resource/Computer	http://rdf.freebase.com/ns/m.074j35	http://sw.opencyc.org/concept/Mx4rvVjXzZwpEbGdrcN5Y29ycA

Taxonomy

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

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



{"status":"OK","usage":"By accessing AlchemyAPI or using information generated by AlchemyAPI, you are agreeing to be bound by the AlchemyAPI Terms of Use: http://www.alchemyapi.com
/company/terms.html","url":"https://tctechcrunch2011.files.wordpress.com/2017/01/2000g-extracted-image.jpg?w=150","totalTransactions":"24","language":"english","title":"IBM launches
quantum computing as a cloud service","publicationDate":{"date":"20160503T000000","confident":"yes"},"image":"https://tctechcrunch2011.files.wordpress.com/2017/01/2000g-extracted-
image.jpg?w=150","imageKeywords":[{"text":"machine","score":"0.622459"}],"authors":{"names":["Ron Miller"]},"docSentiment":{"type":"positive","score":"0.200695","mixed":"1"},"feeds":
[{"feed":"https://techcrunch.com/2016/05/03/ibm-brings-experimental-quantum-computing-to-the-cloud/feed/"}],"concepts":[{"text":"Computer","relevance":"0.968407","dbpedia":"http:
//dbpedia.org/resource/Computer","freebase":"http://rdf.freebase.com/ns/m.074j35","opencyc":"http://sw.opencyc.org/concept/Mx4rvVjXzZwpEbGdrcN5Y29ycA"}],"entities":
[{"type":"Company","relevance":"0.884866","sentiment":{"type":"positive","score":"0.508741"},"emotions":
{"anger":"0.116963","disgust":"0.414211","fear":"0.402641","joy":"0.235321","sadness":"0.419194"},"count":"11","text":"IBM","disambiguated":{"subType":
["SoftwareLicense","OperatingSystemDeveloper","ProcessorManufacturer","SoftwareDeveloper","CompanyFounder","ProgrammingLanguageDesigner","ProgrammingLanguageDeveloper"],"name":"IBM",
"website":"http://www.ibm.com/","dbpedia":"http://dbpedia.org/resource/IBM","freebase":"http://rdf.freebase.com/ns/m.03mc8","opencyc":"http://sw.opencyc.org/concept
/Mx4rvViMoJwpEbGdrcN5Y29ycA","yago":"http://yago-knowledge.org/resource/IBM","crunchbase":"http://www.crunchbase.com/company/ibm"}},"relations":[{"sentence":" Quantum computing is
still very much in the early research stage, but IBM is hoping to accelerate the progress around it by making a quantum computer available to researchers as a cloud
service.","subject":{"text":"Quantum computing","entities":[{"type":"FieldTerminology","text":"Quantum computing"}],"keywords":[{"text":"Quantum computing"}]},"action":
{"text":"is","lemmatized":"be","verb":{"text":"be","tense":"present"}},"object":{"text":"still very much"}}],"taxonomy":[{"label":"/technology and computing","score":"0.454731"},
{"confident":"no","label":"/business and industrial/aerospace and defense/space technology","score":"0.250856"},{"confident":"no","label":"/technology and computing/hardware
/computer","score":"0.210581"}],"keywords":[{"emotions":
{"anger":"0.191809","disgust":"0.168787","fear":"0.125821","joy":"0.11819","sadness":"0.506525"},"relevance":"0.901214","sentiment":
{"score":"0.66709","type":"positive"},"text":"quantum computing"}],"dates":[],"docEmotions":
{"anger":"0.171553","disgust":"0.160641","fear":"0.181507","joy":"0.251675","sadness":"0.493101"}}

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 matching a search query.

The completed code snippet is shown below for reference.

Listing 1: Node.js code in app.js to call the AlchemyData News API

```
1.  app.get("/news", function (req, res) {
2.    var alchemyDataNews = new watson.AlchemyDataNewsV1({
3.      // api_key: '<api-key>'
4.    });
5.
6.    var fields = [
7.      "enriched.url.image",
8.      "enriched.url.url",
9.      "enriched.url.title"
10.   ];
11.
12.   var params = {
13.     maxResults: req.query.hasOwnProperty("max") ? req.query.max : 10,
14.     start: req.query.hasOwnProperty("start") ? req.query.start : "now-1d",
15.     end: req.query.hasOwnProperty("end") ? req.query.end : "now",
16.     return: fields.join(",")
17.   };
18.
19.   // For more search fields, please refer to ibm.biz/nodejs-alchemyapi-parameters
20.   if (req.query.hasOwnProperty("title")) {
21.     params["q.enriched.url.title"] = req.query.title;
22.   }
23.
24.   alchemyDataNews.getNews(params, function (err, news) {
25.     if (err) {
26.       res.send(err);
27.     } else {
28.       if (req.query.format == "json") {
29.         res.send(news);
30.       } else {
31.         res.send(mustache.render(templates.news, news));
32.       }
33.     }
34.   });
35. });
```

Copy and paste the code at:

ibm.biz/nodejs-alchemyapi-news

Listing 2: HTML in the file named template/news.html to display results from the AlchemyData News API

```
1.  {{#result.docs}}
2.    <div style="padding-top: 10px">
3.      {{#source.enriched.url.image}}
4.        <img src="{{source.enriched.url.image}}" style="width: 200px; vertical-align: middle" />
5.      {{/source.enriched.url.image}}
6.      {{^source.enriched.url.image}}
7.        <img src="http://placehold.it/200x100" style="max-width: 200px; vertical-align: middle" />
8.      {{/source.enriched.url.image}}
9.      <a href="{{source.enriched.url.url}}" target="_blank">{{source.enriched.url.title}}</a>
10.   </div>
11. {{/result.docs}}
```

1. Add the code in *Listing 1* to the file named `app.js`. You can also find the code snippet at `ibm.biz/nodejs-alchemyapi-news`. Here's a brief introduction of what this code does:
    - On line 1, we create a Node.js Express endpoint `/news`. This endpoint can be accessed by taking the application URL and appending `/news`. The application takes one query parameter, `title`, and two optional query parameters, `max` and `format`. The query parameter `title` is passed to the AlchemyData News API and is used to match the title of any returned news articles. The query parameter `max` is used to limit the number of results returned. To make this application versatile, the query parameter `format` provides two options for the returned result. By setting `format=json` in the query string, the application will return the raw JSON from the AlchemyData News API. If omitted, HTML displaying the results, from the template file `template/news.html`, will be returned.
    - Lines 2-4 use the `watson-developer-cloud` NPM package, which abstracts the HTTP request made to the IBM Watson API. On line 3, the Alchemy API key is passed to the library. If left blank, the NPM package will look for the

API key in the VCAP services in the IBM Bluemix environment.

- Lines 6-10 create an array of fields that AlchemyData News API should return. See the table below for more details on each parameter.

Parameter	What AlchemyData News API returns
`enriched.url.image`	the primary image of the news article
`enriched.url.url`	the URL to the original source of the news article
`enriched.url.title`	title of the news article
…	Please refer to `ibm.biz/nodejs-alchemyapi-parameters` for over 400 parameters

- These fields are concatenated together on line 16 and assigned to the parameter labeled `return`.
- The parameter `maxResults` limits the number of results returned, or defaults to 10 items.
- Lines 20-22 set the search parameter of the title of the article to the value passed in via the URL query parameter `title`. There are many more fields that can be searched and more information can be found at `ibm.biz/nodejs-alchemyapi-parameters`
- Lines 24-34 execute the search and handle the result. One of three actions occur:
  i. Displays an error if one occurs (lines 25-27)
  ii. Displays the raw JSON result from the API (lines 28-30)
  iii. Displays a rendered HTML page (lines 30-32)
- The last option uses the mustache templating library to render the HTML located in the `templates/news.html` file.

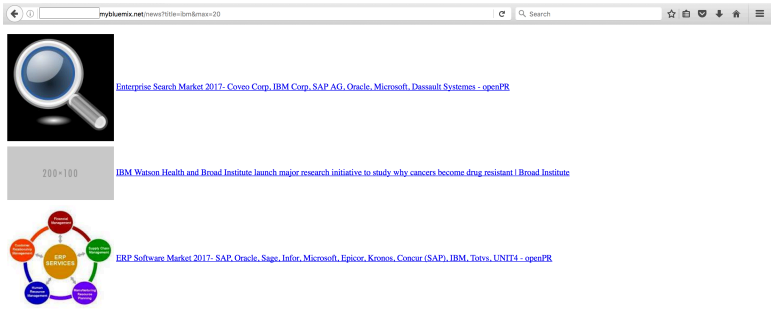
2. Commit and deploy the changes by following the directions in the section titled **IBM Bluemix DevOps Services** on page 12. When finished with instructions in that section, continue onto Step #3.
3. When the application has been deployed, open a browser tab and visit your application's endpoint.

   `http://<<MY-APP>>.mybluemix.net/news?title=<<SEARCH-QUERY>>`

   - Replace <<MY-APP>> with the host of the Node.js application you created.
   - Replace <<SEARCH-QUERY>> with a term to search article titles

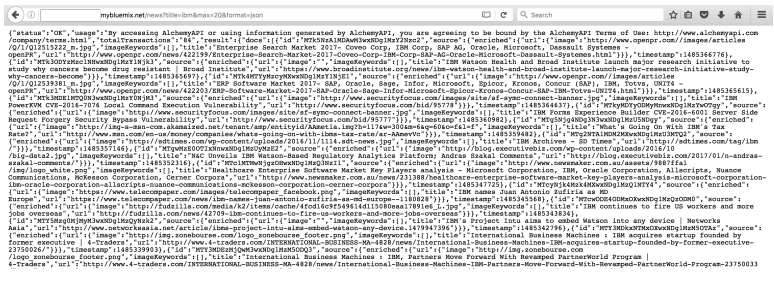
   Note: You can also insert in the query string:
   - `max=<<NUMBER>>` limit the number of results returned.
   - `start=now-<<TIME-VALUE>>` or `end=now-<<TIME-VALUE>>` where `<<TIME-VALUE>>` is a number followed by the unit `s` (seconds), `m` (minutes), `h` (hours), `d` (days), `M` (months), or `y` (years) (i.e. `now-1d` = 1 day ago)

4. Depending on the search term, a list of news articles should be displayed with an associated image and link to the article.



5. To see the JSON representation of the results, insert `format=json` in the URL query string.

   `http://<<MY-APP>>.mybluemix.net/news?format=json&title=<<SEARCH-QUERY>>`

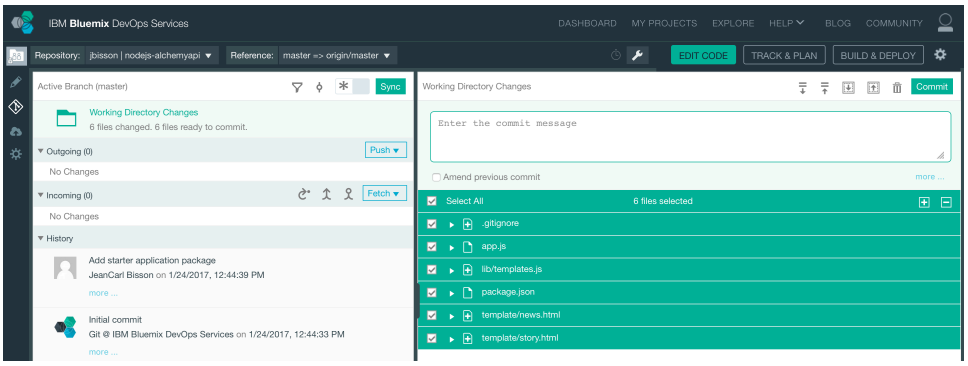


Deploy Changes via IBM Bluemix DevOps Services

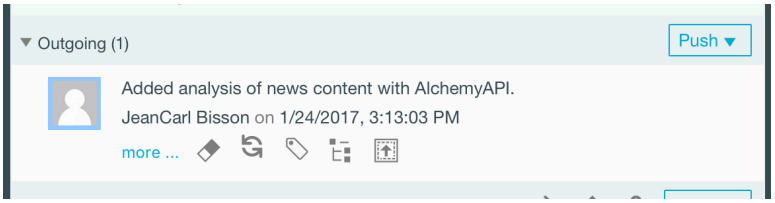
In this section, you will learn how to commit changes in your Git repository and deploy the changes to a Cloud Foundry application on IBM Bluemix.

1. After making modifications to the code and when ready to deploy the application, click on the Git icon on the left side of the web-based editor in the IBM Bluemix DevOps Services.

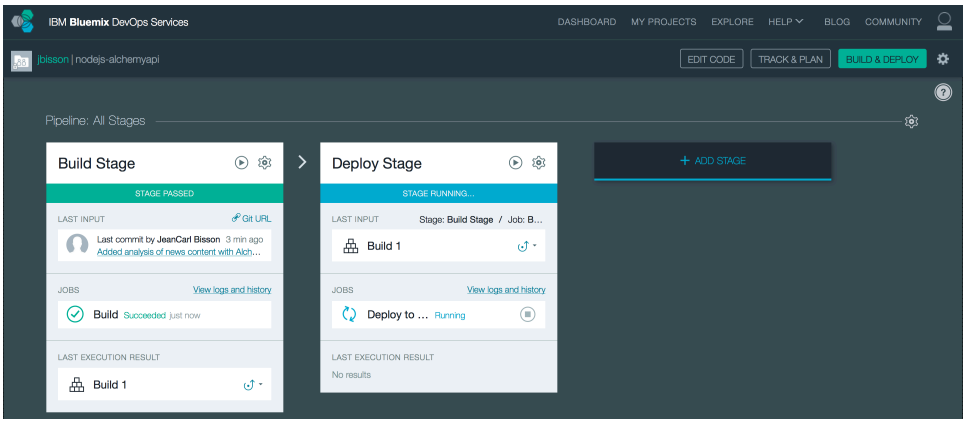
2. Files that have been changed will be listed in the pane on the right. Enter a commit message that describes these changes. Click on **Commit** in the top-right corner.



3. A new item will be added in the Outgoing list in the left pane. Click on the **Push** button to push these changes into the repository.



4. Click on the **Build and Deploy** button in the top-right corner of the page. This is where you can watch the two stages in the pipeline that are triggered when new code is committed to the Git repository.



5. When both stages pass, your application has been deployed successfully.