

AlchemyAPI in Node.js

Hands-On Lab

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



Setup Environment in IBM Bluemix	2
Create Git Repository/Import Project Files	4
Analyze a News Article	6
Search for News with AlchemyData News	10
Deploy Changes via IBM Bluemix DevOps Services.....	12

IBM launches quantum computing as a cloud service

Author • Roe Miller
Document Sentiment positive (Score: 0.20699)
Document Emotion • Joy (Score: 0.1655)
• Disgust (Score: 0.1666)
• Fear (Score: 0.11407)
• Sadness (Score: 0.11407)
• Surprise (Score: 0.48510)

URL <https://techcrunch.com/2016/05/03/ibm-brings-experimental-quantum-computing-to-the-cloud/>
Published 2016/05/03/00000
Views 20K

Entities

Entity	Type	Relevance	# Occurrences
IBM	Company	0.08466	[1]

Keywords

Keyword	Relevance
quantum computing	0.90214

Concepts

Concept	Relevance	Dfpedia	Freebase	OpenCyc
Computer	0.096407	http://dbpedia.org/resource/Computer	http://freebase.com/api/actm/07453	http://www.opency.org/concepts/MolcYXZqplG6abNSY2byuA

Taxonomy

Analyze a News Article (pg. 6)

Enterprise Search Market 2017: Cisco Corp, IBM Corp, SAP AG, Oracle, Microsoft, Dell/EMC Systems, openPR

IBM Watson Health and Broad Institute launch major research initiative to study why cancers become drug resistant | Broad Institute

ERP Software Market 2017: SAP, Oracle, Sage, Infosys, Microsoft, Fujitsu, KPMG, Cerner (SAP), IBM, Tora, UNIT4, openPR

IBM PowerVM CVE-2016-7076 Local Command Execution Vulnerability

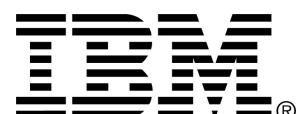
IBM Forms Experience Builder CVE-2016-4001 Server Side Request Forgery Security Bypass Vulnerability

IBM's Going On With IBM's Tax Rate?

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**.

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a sidebar with categories like All Categories, Infrastructure, Compute, Storage, Network, Security, Apps (with sub-options like Boilerplates, Cloud Foundry Apps, Containers, OpenWhisk, Mobile), Services (Data & Analytics, Watson, Internet of Things, APIs, Network, Storage, Security, DevOps, Application Services), and a Filter button. The main area displays various runtime options: Liberty for Java™, SDK for Node.js™ (which is selected and highlighted with a blue border), XPages, ASP.NET Core, Runtime for Swift, PHP, Python, Go, Ruby, and Tomcat. Each option has a brief description and an IBM logo.

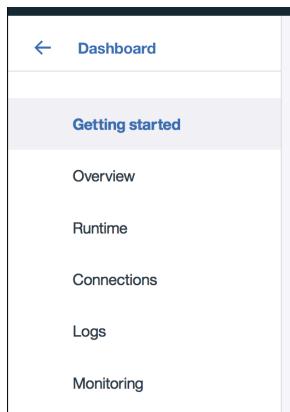
2. On the next screen, enter a name for your application in the field labeled **App name**. This name is for your convenience. Enter a host name, which will be the first part of the URL for your application. If you choose myapp, the application URL will be `myapp.mybluemix.net`. There can only be one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

The screenshot shows the 'Create a Cloud Foundry App' page. At the top, it says 'View all' and 'Create a Cloud Foundry App'. Below that, there's a section for 'SDK for Node.js™' with a brief description and an IBM logo. To the right, there are fields for 'App name' (set to 'myapp'), 'Host name' (set to 'myapp'), and 'Domain' (set to 'mybluemix.net'). Further down, there's a 'View Docs' section with details about the version (3.x), type (Application), and region (US South). On the right, there's a 'Pricing Plans' section with a table:

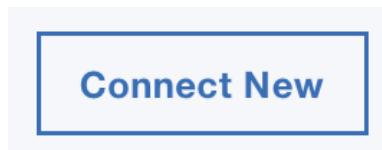
PLAN	FEATURES	PRICING
Default	Run one or more apps free for 30 days (375 GB-hours free).	\$0.07 USD/GB-Hour

Below the table, it says 'This is a service plan for the IBM Bluemix Platform runtime.' At the bottom, there are links for 'Need Help?', 'Contact Bluemix Sales', 'Estimate Monthly Cost', 'Cost Calculator', '\$0.07 USD/GB-Hour', and a 'Create' button.

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.

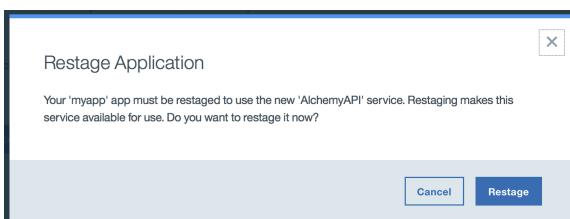


5. Click on the **AlchemyAPI** tile under the Watson section. You can leave the fields as the default values. Click on **Create**.

A screenshot of the Watson Services catalog. On the left, there's a sidebar with categories: All Categories, Apps, Mobile, Services, Data & Analytics, Watson (which is selected and highlighted in blue), Internet of Things, APIs, Storage, Security, DevOps, Application Services, and Integrate. The main area shows a grid of service tiles:

- AlchemyAPI**: An AlchemyAPI service that analyzes your unstructured text and image content. Status: IBM
- Concept Insights**: Explore the concepts behind your input, identifying associations beyond traditional text matching. Status: IBM | Deprecated
- Conversation**: Add a natural language interface to your application to automate interactions with your end users. Status: IBM
- Dialog**: Enable your application to use natural language to converse with users. Status: IBM | Deprecated
- Document Conversion**: Converts a HTML, PDF, or Microsoft Word™ document into a normalized HTML, plain text, or JSON. Status: IBM
- Language Translation**: Translate text from one language to another for specific domains. Status: IBM | Deprecated
- Language Translator**: Translate text from one language to another for specific domains. Status: IBM
- Natural Language Classifier**: Natural Language Classifier performs natural language classification on question texts. A user... Status: IBM
- Personality Insights**: The Watson Personality Insights derives insights from transactional and social media data to identify... Status: IBM
- Retrieve and Rank**: Add machine learning enhanced search capabilities to your application. Status: IBM
- Speech to Text**: Low-latency, streaming transcription. Status: IBM
- Text to Speech**: Synthesizes natural-sounding speech from text. Status: IBM
- Tone Analyzer**: Tone Analyzer uses linguistic analysis to detect three types of tones from communications: emotion, language, and social context. Status: IBM
- Tradeoff Analytics**: Helps make better choices under multiple conflicting goals. Combines smart visualization and... Status: IBM
- Visual Recognition**: Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose... Status: IBM

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

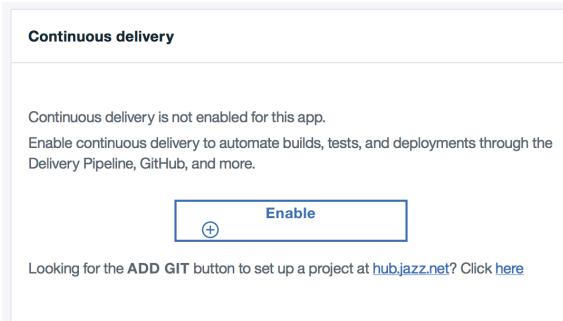


7. When the application has finished restaging, the Node.js application environment will contain the Watson service credentials.

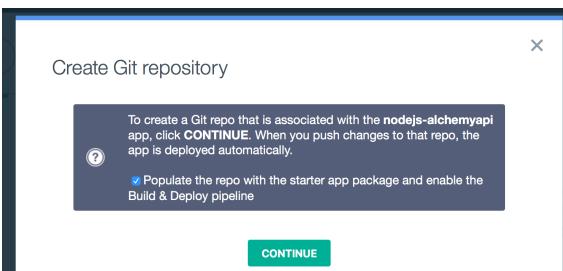
Create Git Repository/Import Project Files

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**.

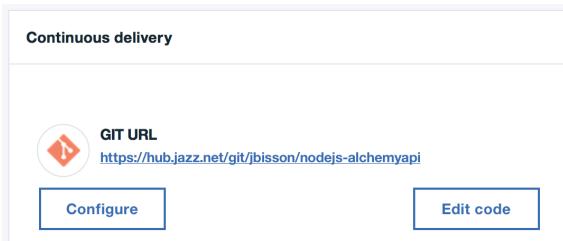
1. Open the application overview for the Node.js application. In the bottom right corner in the box titled **Continuous delivery**, click on the last link, **Click here**, to create a project at hub.jazz.net.



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



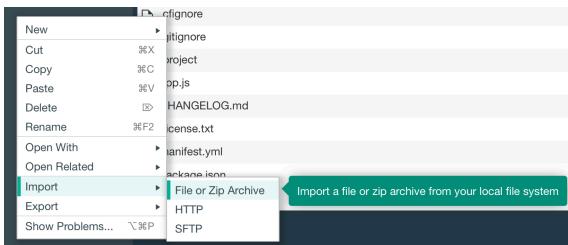
3. You will be returned to the application overview. Click on **Edit code**.



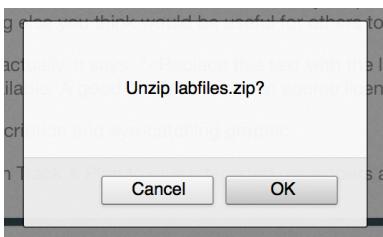
4. You will be taken into the IBM Bluemix DevOps Services tooling. You can edit code in the browser, commit changes to the Git repository, and setup testing and deployment pipeline options when code changes are made. The default pipeline automatically deploys new changes that are committed to the Git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

The screenshot shows the IBM Bluemix DevOps Services interface. On the left, there's a sidebar with a tree view of the project structure. The main area shows a list of files with their details. The files listed include .gitignore, package.json, README.md, .gitignore, public, .gitignore, project, app.js, CHANGELOG.md, License.txt, manifest.yml, package.json, and README.md. Each file entry includes a timestamp (e.g., 1/24/2017, 12:46:07 PM), size (e.g., 1 KB), and a download icon.

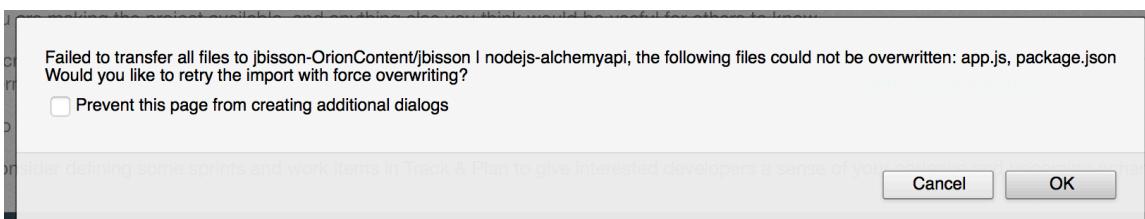
5. Let's import a couple of files that the following sections need. Download the Zip file from ibm.biz/nodejs-alchemyapi-labfiles
6. Under the file listing in the left sidebar, right click to display the context menu. Select **Import -> File or Zip Archive**. Select the downloaded Zip file.



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.

```

8   "dependencies": {
9     "express": "4.13.x",
10    "cfenv": "1.0.x",
11    "mustache": "2.3.x",
12    "watson-developer-cloud": "2.15.x"
13  },
14  ...
15 }

```

```

"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

```
1. app.get("/story", function (req, res) {
2.   var alchemyLanguage = new watson.AlchemyLanguageV1({
3.     // api_key: '<api-key>'
4.   });
5.
6.   var features = [
7.     "entities",
8.     "keywords",
9.     "title",
10.    "authors",
11.    "taxonomy",
12.    "concepts",
13.    "relations",
14.    "pub-date",
15.    "doc-sentiment",
16.    "doc-emotion",
17.    "page-image",
18.    "feeds"
19.  ];
20.
21.  var parameters = {
22.    extract: features.join(","),
23.    sentiment: 1,
24.    emotion: 1,
25.    maxRetrieve: 1,
26.    url: req.query.url
27.  };
28.
29.  alchemyLanguage.combined(parameters, function (err, response) {
30.    if (err) {
31.      res.send(err);
32.    } else {
33.      if (req.query.format == "json") {
34.        res.send(response);
35.      } else {
36.        response.url = req.query.url;
37.        res.send(mustache.render(templates.story, response));
38.      }
39.    }
40.  });
41.});
```

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

```
1. <h1>{{title}}</h1>
2. <table>
3.   <tr>
4.     <td>Author</td>
5.     <td>
6.       <ul>
7.         {{#authors.names}}
8.           <li>{{.}}</li>
9.         {{/authors.names}}
10.      </ul>
11.    </td>
12.  </tr>
13.  <tr>
14.    <td>Document Emotion</td>
15.    <td>
16.      <ul>
17.        <li>Anger (Score: {{docEmotions.anger}})</li>
18.        <li>Disgust (Score: {{docEmotions.disgust}})</li>
19.        <li>Fear (Score: {{docEmotions.fear}})</li>
```

```

20.      <li>Joy (Score: {{docEmotions.joy}})</li>
21.      <li>Sadness (Score: {{docEmotions.sadness}})</li>
22.    </ul>
23.  </td>
24. </tr>
25. <tr>
26.   <td>URL</td>
27.   <td><a href="{{href}}" target="_blank">{{href}}</a></td>
28. </tr>
29. <tr>
30.   <td>Published</td>
31.   <td>{{publicationDate.date}}</td>
32. </tr>
33. <tr>
34.   <td></td>
35.   <td><a href="/story?format=json&url={{href}}">View JSON</a></td>
36. </tr>
37. </table>
38.
39. <h2>Entities</h2>
40. <table border="1">
41.   <tr>
42.     <th>Entity</th>
43.     <th>Type</th>
44.     <th>Relevance</th>
45.     <th># Occurrences</th>
46.   </tr>
47. {{#entities}}
48.   <tr>
49.     <td>{{text}}</td>
50.     <td>{{type}}</td>
51.     <td>{{relevance}}</td>
52.     <td>{{count}}</td>
53.   </tr>
54. {{/entities}}
55. </table>
56.
57. <h2>Keywords</h2>
58. <table border="1">
59.   <tr>
60.     <th>Keyword</th>
61.     <th>Relevance</th>
62.   </tr>
63. {{#keywords}}
64.   <tr>
65.     <td>{{text}}</td>
66.     <td>{{relevance}}</td>
67.   </tr>
68. {{/keywords}}
69. </table>
70.
71. <h2>Concepts</h2>
72. <table border="1">
73.   <tr>
74.     <th>Concept</th>
75.     <th>Relevance</th>
76.     <th>DBpedia</th>
77.     <th>Freebase</th>
78.     <th>OpenCyc</th>
79.   </tr>
80. {{#concepts}}
81.   <tr>
82.     <td>{{text}}</td>
83.     <td>{{relevance}}</td>
84.     <td><a href="{{dbpedia}}" target="_blank">{{dbpedia}}</a></td>
85.     <td><a href="{{freebase}}" target="_blank">{{freebase}}</a></td>
86.     <td><a href="{{opencyc}}" target="_blank">{{opencyc}}</a></td>
87.   </tr>
88. {{/concepts}}
89. </table>
90.
91. <h2>Taxonomy</h2>
92. <table border="1">
93. <tr>
94.   <th>Label</th>
95.   <th>Score</th>
96.   <th>Confident</th>
97. </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. 

```

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: 0.20095) • Anger (Score: 0.071551)
• Disgust (Score: 0.16664)
• Fear (Score: 0.181507)
• Joy (Score: 0.251675)
• Sadness (Score: 0.493101)

Document Emotion
URL <https://techcrunch.com/2016/05/03/ibm-brings-experimental-quantum-computing-to-the-cloud/>
Published 2016/05/03T000000
[View JSON](#)

Entities

Entity	Type	Relevance	# Occurrences
IBM	Company	0.884866	[1]

Keywords

Keyword	Relevance
quantum computing	0.901214

Concepts

Concept	DBpedia	Freebase	OpenCyc
Computer	http://dbpedia.org/resource/Computer	http://rdf.freebase.com/ns/m.07435	http://www.opencyc.org/concept/MxlvVJXzZwpE8GdnvN3Y29yA

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",
  "url": "http://mybluemix.net/story?format=json&url=https://techcrunch.com/2016/05/03/ibm-brings-experimental-quantum-computing-as-a-cloud-service",
  "sentiment": {
    "polarity": "positive",
    "score": 0.20095
  },
  "emotion": {
    "anger": 0.071551,
    "disgust": 0.16664,
    "fear": 0.181507,
    "joy": 0.251675,
    "sadness": 0.493101
  },
  "document": {
    "title": "IBM launches quantum computing as a cloud service",
    "text": "IBM launches quantum computing as a cloud service",
    "date": "2016/05/03T000000",
    "published": "2016/05/03T000000"
  },
  "entities": [
    {
      "type": "Company",
      "name": "IBM",
      "relevance": 0.884866
    }
  ],
  "keywords": [
    {
      "text": "quantum computing",
      "relevance": 0.901214
    }
  ],
  "concepts": [
    {
      "type": "Computer",
      "uri": "http://dbpedia.org/resource/Computer",
      "label": "Computer"
    },
    {
      "type": "Freebase",
      "uri": "http://rdf.freebase.com/ns/m.07435",
      "label": "Computer"
    },
    {
      "type": "OpenCyc",
      "uri": "http://www.opencyc.org/concept/MxlvVJXzZwpE8GdnvN3Y29yA",
      "label": "Computer"
    }
  ],
  "taxonomy": [
    {
      "label": "Technology and Computing",
      "score": 0.454731
    },
    {
      "label": "Hardware and Computing",
      "score": 0.215981
    },
    {
      "label": "Software and Programming Languages",
      "score": 0.11894
    },
    {
      "label": "Programmers and Developers",
      "score": 0.11894
    },
    {
      "label": "Cloud Computing",
      "score": 0.11894
    },
    {
      "label": "Quantum Computing",
      "score": 0.11894
    }
  ]
}
```

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.     return: fields.join(",")
15.   };
16.
17.   // For more search fields, please refer to ibm.biz/nodejs-alchemyapi-parameters
18.   if (req.query.hasOwnProperty("title")) {
19.     params["q.enriched.url.title"] = req.query.title;
20.   }
21.
22.   alchemyDataNews.getNews(params, function (err, news) {
23.     if (err) {
24.       res.send(err);
25.     } else {
26.       if (req.query.format == "json") {
27.         res.send(news);
28.       } else {
29.         res.send(mustache.render(templates.news, news));
30.       }
31.     }
32.   });
33. });


```

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.       
5.     {{/source.enriched.url.image}}
6.     {{^source.enriched.url.image}}
7.       
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 14 and assigned to the parameter labeled `return`.
- The parameter `maxResults` limits the number of results returned, or defaults to 10 items.
- Lines 18-20 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 22-32 execute the search and handle the result. One of three actions occur:
 - i. Displays an error if one occurs (lines 23-25)
 - ii. Displays the raw JSON result from the API (lines 26-28)
 - iii. Displays a rendered HTML page (lines 28-30)
- The last option uses the mustache templating library to render the HTML located in the `templates/news.html` file.

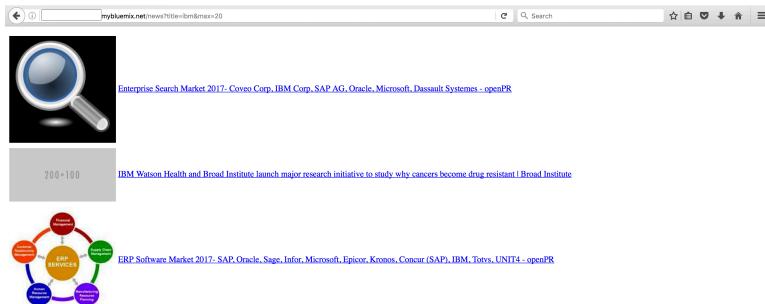
2. Commit and deploy the changes by following the directions in the section titled **IBM Bluemix DevOps Services** on page 12. When finished with instructions in that section, continue onto Step #3.
3. When the application has been deployed, open a browser tab and visit your application's endpoint.

`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

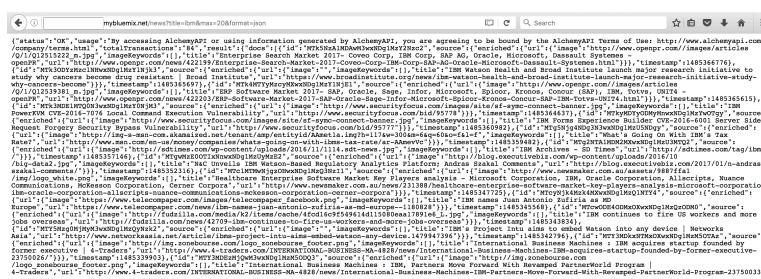
Note: You can also insert `max=<<NUMBER>>` in the query string and limit the number of results returned.

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.

A screenshot of the IBM Bluemix DevOps Services interface. On the left, there's a sidebar with 'Active Branch (master)', 'Working Directory Changes' (listing 6 files ready to commit), 'Outgoing (0)' (No Changes), 'Incoming (0)' (No Changes), and a 'History' section showing an initial commit by JeanCarl Bisson on 1/24/2017. On the right, a 'Working Directory Changes' panel shows a list of selected files: .gitignore, app.js, lib/templates.js, package.json, template/news.html, and template/story.html. A text input field says 'Enter the commit message'. Below it are checkboxes for 'Amend previous commit' and 'Select All', with 'Select All' checked. A 'Commit' button is at the top right of the panel.

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.

A screenshot of the 'Outgoing' list in the left sidebar. It shows one item: 'Added analysis of news content with AlchemyAPI.' by JeanCarl Bisson on 1/24/2017, 3:13:03 PM. To the right of the list is a 'Push' button.

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.

A screenshot of the 'Pipeline: All Stages' interface. It shows two stages: 'Build Stage' and 'Deploy Stage'. The 'Build Stage' is labeled 'STAGE PASSED' with a green bar. It shows 'LAST INPUT' from a Git URL, 'JOBS' with a successful 'Build' job, and 'LAST EXECUTION RESULT' showing 'Build 1'. The 'Deploy Stage' is labeled 'STAGE RUNNING...' with a blue bar. It shows 'LAST INPUT' as 'Stage: Build Stage / Job: B...', 'JOBS' with a 'Deploy to ...' job in 'Running' status, and 'LAST EXECUTION RESULT' showing 'No results'. A '+ ADD STAGE' button is located at the bottom right of the pipeline area.

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