

# Alchemy Language/News Hands-on Lab

---

## Introduction

IBM Watson offers a broad set of services to enable developers to build cognitive solutions. Cognitive Computing Systems are defined as systems that **understand**, **reason**, and **learn** to assist humans in making better decisions by penetrating the complexity of Big Data.

To help with the task of understanding, Alchemy Language services are designed to extract keywords, entities, relations, sentiment, and concepts expressed in text. In this lab, we will step through the process of building a [Node.js](#) app to extract keywords and entities from a web page and displaying the results in a word cloud using a [d3.js](#) library from Jason Davies [d3 cloud](#).

Using that you will then expand your solution to leverage Alchemy News to identify most relevant News articles to a given query and display extracted keywords and entities.

## Bluemix App

To start, we will build a simple app using node.js runtime and host it on Bluemix, IBM's platform-as-a-service solution. You should have signed up for Bluemix account already. If not, go to <http://bluemix.net> and sign up for a Bluemix account.

- 1- Log into your Bluemix account using username and password you created when you registered for your account.
- 2- Go to Bluemix Dashboard
- 3- Create App
- 4- Select WEB
- 5- SDK for Node.js
- 6- Hit Continue
- 7- Give your app a unique name (for example, {your\_initials}-alchemy)
  - ➔ You should see this message: "Your application is staging: <http://jk-alchemy.mybluemix.net>"
  - ➔ At this point you have a running Node.js app on Bluemix (app doesn't do much yet) and next we need to decide how we'd like to contribute code to this app. In this lab, we will use the CF command line interface.
- 8- Select CF command line interface
  - ➔ To manage the application using CF command line, need to download the Bluemix Command Line Interface and the CF command line interface.
- 9- Follow the instructions to download Bluemix command line interface and the CF command line interface.

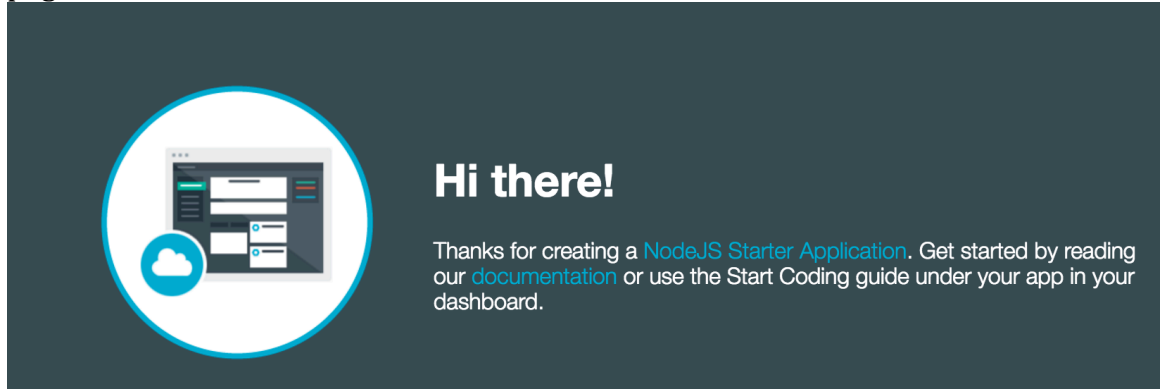
⇒ After installation completes, you can verify installation by issuing these commands in a terminal:

Open a terminal

bluemix -v

cf -v

10-You can verify that your app is running by directing your browser to:  
<https://jk-alchemy.mybluemix.net> You should see the following landing page:



11-To make edits and add code to your app, you can download the starter code, make edits and then push the updates to Bluemix.

- a. Press Download Starter Code button and this should download a zip file to your Downloads directory.
- b. On your local computer, create a directory where you'll be working. For the rest of this lab, we'll reference that directory as DIR.
- c. `Cd $DIR`
- d. `Mv ~/Downloads/jk-alchemy.zip .`
- e. `Unzip jk-alchemy.zip`
- f. Review the manifest.yml file which includes application information such as application name, host, what services it uses and what are memory and disk requirements. To change the name or host of the application, you can change it in manifest.yml and then push the changes to Bluemix.

12-Lastly, we'll make a simple change to the app and push the updates to Bluemix.

- a. Edit index.html file using your favorite editor.  
`Cd $DIR`  
`vi public/index.html`
- b. Change the "Hi there" message to "Word Cloud Visualization of Alchemy Language Results"  
`<h1>Hi there!</h1>`

➔ <h1>Word Cloud Visualization of Alchemy Language Results</h1>

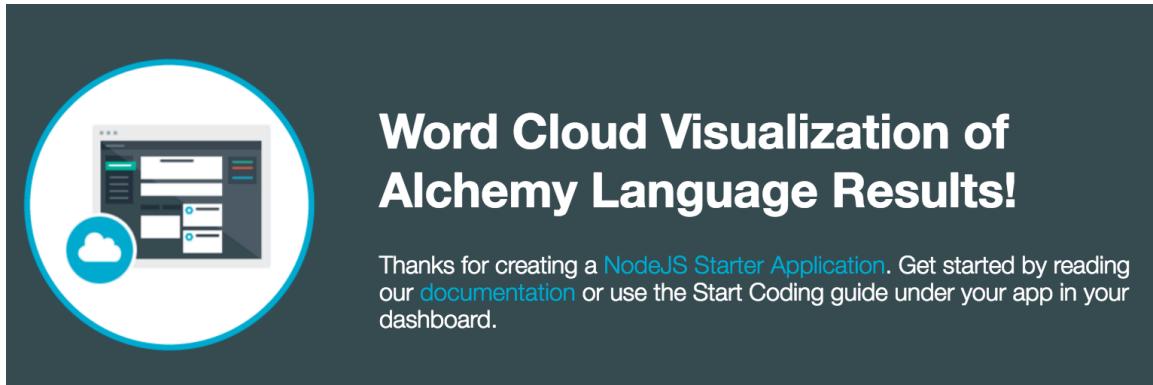
- c. Save your file
- d. Push changes to Bluemix

cd \$DIR

cf push

➔ It will take a minute or two to push the changes to Bluemix.

➔ Verify the change is reflected by pointing your browser to  
<https://jk-alchemy.mybluemix.net>



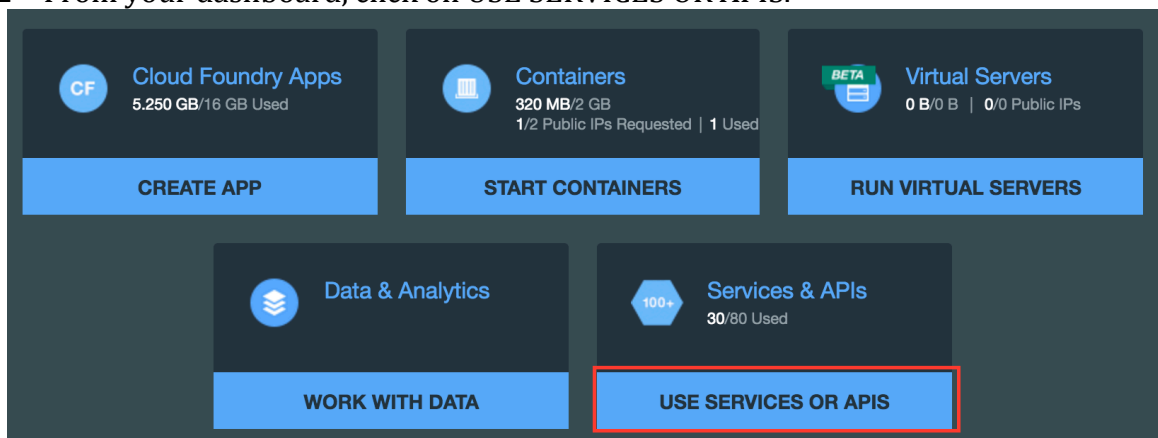
The banner features a dark blue background. On the left, there is a circular icon containing a stylized illustration of a computer monitor displaying a web application, with a small cloud icon below it. To the right of the icon, the title "Word Cloud Visualization of Alchemy Language Results!" is written in large, bold, white text. Below the title, a smaller line of white text reads: "Thanks for creating a [NodeJS Starter Application](#). Get started by reading our [documentation](#) or use the Start Coding guide under your app in your dashboard."

## Alchemy Language Keyword Extraction

[Alchemy Language](#) offers several APIs for semantic text analytics which use sophisticated natural language processing techniques to analyze textual content and extract keywords, entities, and concepts.

To leverage any of the Watson Developer Cloud services (including Alchemy Language), you need to create a service instance in Bluemix which, upon success, returns credentials that you can use in accessing the service.

- 1- Log in to Bluemix using your username and password.
- 2- From your dashboard, click on USE SERVICES OR APIS.



- 3- From the catalog of Watson services, select AlchemyAPI service.



- 4- Accept all the default fields and hit CREATE.  
Space: dev  
App: Leave unbound  
Service name: AlchemyAPI-iu (by default the system creates a unique service name)  
Credential name: Credentials-1 (by default, the system creates a unique credential name).  
Selected Plan: free  
➔ When complete, an AlchemyAPI service is created.
- 5- Click the Service Credentials in the left navigation bar and capture the apikey which we will need in subsequent steps in the lab. This apikey is needed whenever you make a call to Alchemy Language service.

The screenshot shows the AlchemyAPI-iu dashboard. On the left, a sidebar contains links: 'Back to Dashboard...', 'Manage', 'Service Credentials' (highlighted with a red arrow), 'Service Access Authorization', and 'APPS USING SERVICE'. The main content area is titled 'Service Credentials' and includes a green 'ADD CREDENTIALS' button. Below this, a table lists credentials with columns 'NAME' and 'Credentials-1'. A 'DELETE' button is next to the entry. The 'SERVICE CREDENTIALS' section displays a JSON snippet with the API key and URL highlighted by red boxes.

```
{
  "credentials": {
    "url": "https://gateway-a.watsonplatform.net/calls",
    "apikey": "2f945cf36a8cee6879c1875cb62aa0f81e29db64",
    "note": "It may take up to 5 minutes for this key to become active"
  }
}
```

After obtaining the AlchemyAPI credentials, experiment by running the following commands using curl (or your preferred REST client):

***apikey=YOUR\_API\_KEY;*** ➔ your AlchemyAPI credentials  
***url='http://www.cnn.com/2015/07/15/africa/obama-kenya-trip/index.html';***

#### ***Keywords (words of interest)***

curl [http://gateway-a.watsonplatform.net/calls/url/URLGetRankedKeywords?apikey=\\${apikey}&url=\\${url}&outputMode=json](http://gateway-a.watsonplatform.net/calls/url/URLGetRankedKeywords?apikey=${apikey}&url=${url}&outputMode=json)

➔ Review the response

#### ***Entities (semantic clusters of keywords)***

curl [http://gateway-a.watsonplatform.net/calls/url/URLGetRankedNamedEntities?apikey=\\${apikey}&url=\\${url}&outputMode=json](http://gateway-a.watsonplatform.net/calls/url/URLGetRankedNamedEntities?apikey=${apikey}&url=${url}&outputMode=json)

➔ Review the results

To get a complete list of all entities supported by Alchemy Language, review [the full list of supported entities](#)

### *Concepts (abstractions of keywords and entities)*

```
curl http://gateway-a.watsonplatform.net/calls/url/URLGetRankedConcepts?apikey=${apikey}&url=${url}&outputMode=json
```

➔ Review the results

### *Sentiment (how is someone expressing themselves)*

```
apikey=YOUR_API_KEY; ➔ your AlchemyAPI credentials
url= 'http://www.cnn.com/2016/04/29/politics/donald-trump-california-protest/index.html';
phrase='At least one police car was damaged and several scuffles broke out amid the hectic scene';
```

### **Overall Text Sentiment**

```
curl http://gateway-a.watsonplatform.net/calls/url/URLGetTextSentiment?apikey=${apikey}&url=${url}&outputMode=json
```

### **Targeted Sentiment**

```
curl -G "http://gateway-a.watsonplatform.net/calls/url/URLGetTargetedSentiment" --data-urlencode "targets=$phrase1" --data-urlencode "apikey=$apikey" --data-urlencode "url=$url" --data-urlencode "outputMode=json"
```

## **Node.js App using Alchemy Language Keyword Extraction**

In this section, we will explore building a Node.js app that leverages Alchemy Language keyword extraction service. You can proceed to next steps by downloading the sample code from github.

In what follows, we'll assume WORKDIR is the working directory.

1. Open a terminal window
2. `mkdir $WORKDIR`
3. `cd $WORKDIR`
4. `git clone https://github.com/joe4k/alchemy-lang.git`
5. `cd alchemy-lang`
6. Edit manifest.yml in your favorite text editor and change the name and host to unique values.
7. Edit watson/config.js and replace apikey with the value you obtained earlier
8. Push to bluemix

```
cf api https://api.ng.bluemix.net
cf login      (provide your bluemix username and password to login, select your bluemix dev space)
cf push      (this will push your app to Bluemix)
```
9. Verify the app is running as expected:
  - a. Point your browser to <https://{yourappname}.mybluemix.net>
  - b. Supply a url for any website in the form field

# Alchemy Language D3 Visualization

- c. Press Analyze which would call Alchemy Language on that url and return a json object with all the extracted keywords.

```
[
  - {
    relevance: "0.909456",
    text: "police officers"
  },
  - {
    relevance: "0.723278",
    text: "Donald Trump campaign"
  },
  - {
    relevance: "0.637682",
    text: "Costa Mesa"
  },
  - {
    relevance: "0.587602",
    text: "26-year-old community organizer"
  },
  - {
    relevance: "0.57565",
    text: "Costa Mesa Police"
  },
  - {
```

In this sample app, we leverage the Node.js **watson-developer-cloud** package released by IBM to simplify integration with the different Watson Developer Cloud services, including AlchemyAPI. You can review the package.json file under \$WORKDIR/alchemy-language and note the dependencies field, specifically the **watson-developer-cloud** package.

```
"dependencies": {
  "body-parser": "*",
  "extend": "*",
  "request": "*",
  "express": "4.13.x",
```

```
    "cfenv": "1.0.x",  
    "watson-developer-cloud": "^1.0.5"  
  },
```

## Visualizing Alchemy Keywords as Word Cloud with D3

In this section, we will expand on the previous section to return keywords and also display them in a word cloud where the size of the word corresponds to its relevance score as returned by Alchemy Language.

Like before, we'll assume WORKDIR is the current working directory (we recommend creating a new directory for this section):

1. Open a terminal window
2. `mkdir $WORKDIR`
3. `cd $WORKDIR`
4. `git clone` <https://github.com/joe4k/alchemy-kwdcloud-d3.git>
5. `cd alchemy-kwdcloud-d3`
6. Edit `manifest.yml` in your favorite text editor and change the name and host to unique values.
7. Edit `watson/config.js` and replace `apikey` with the value you obtained earlier
8. Push to bluemix

**cf api** <https://api.ng.bluemix.net>

**cf login** (provide your bluemix username and password to login, select your bluemix dev space)

**cf push** (this will push your app to Bluemix)

9. Verify the app is running as expected:
  - a. Point your browser to `https://yourappname.mybluemix.net`.
  - b. Supply a url for any website in the form field

Enter the URL you'd like to analyze with Alchemy Language

Analyze

- c. Press Analyze which would call Alchemy Language on that url and return a word-cloud with all the extracted keywords.





If you'd like to run the app locally, you can follow these steps:

1. Open a terminal window
2. `mkdir $WORKDIR`
3. `cd $WORKDIR`
4. `git clone https://github.com/joe4k/alchemy-kwdcloud-d3.git`
5. `cd alchemy-kwdcloud-d3`
6. Edit `manifest.yml` in your favorite text editor and change the name and host to unique values.
7. Edit `watson/config.js` and replace `apikey` with the value you obtained earlier
8. `npm install` → install dependencies
9. `npm start` → start the app (note the port number where server is started)
10. Verify the app is running as expected:
  - a. Point your browser `http://localhost:port`
  - b. Supply a url for any website in the form field
  - c. Press Analyze and you should see the word cloud for the keywords extracted from that url.

## Alchemy Language Entity Extraction

In this section, you will add code to extract both keywords and entities from a given URL and display word clouds for both.

**\*\*Hint\*\***: Working code available on github [alchemy-cloud-d3](#)

## Alchemy News Keyword/Entity Visualization **\*\*Optional\*\***

This section is optional. Time permitting, add code to query [Alchemy News](#) and visualize returned keywords/entities.

As you work with Alchemy News, it'll be helpful to explore the [full list of Alchemy News API fields](#).