Using IBM Bluemix in an Introductory Natural Language Processing Class

Wlodek Zadrozny and Sumanth Charabuddi

Computer Science, UNC Charlotte

wzadrozn@uncc.edu and scharabu@uncc.edu

Abstract: UNC Charlotte has an active research and instruction program in various areas of cognitive computing. They include visual analytics, human-computer interaction, natural language processing and others. Since August of 2015, IBM Bluemix has been part of a hands-on class on Natural Language Processing at UNCC. In addition to learning standard NLP tools such as OpenNLP, students have experimented with Bluemix capabilities in text and speech processing. The teaching objective in using Bluemix was to show the variety of tools supporting dialogue, information extraction (with AlchemyAPI), and typical NLP tasks (tokenization, POS-tagging, classification etc.). The class curriculum was directed towards Master-level students. As an introductory class, it was designed to provide both some understanding of modeling issues involved in natural language processing, as well as and introduction to NLP tools.

This report describes the organization of the class, Bluemix-related class projects, and some of the instructions created for the class.

Table of Contents

1. Introduction: The Natural Language Processing Class

In this report, we describe our experience in using IBM Bluemix as a part of the natural language processing (NLP) class taught in the Fall of 2015 at UNC Charlotte. The first author was the instructor, and was helped by the second author who was the teaching assistant for this class.

Motivation and History of the Class: UNC Charlotte has an active research and instruction program in various areas of cognitive computing. They include visual analytics, human-computer interaction, natural language processing and others. The original motivation for including Bluemix in an NLP curriculum stems from the fact that a lot of software development is migrating to 'the 'cloud', and therefore the students should be aware of existing cloud-based technologies for NLP. The other teaching objective in using Bluemix was to show the variety of tools supporting dialogue, information extraction (with AlchemyAPI), speech processing and typical NLP tasks (tokenization, POS-tagging, classification etc.).

The instructor's original plan was to offer this class in the Spring of 2016. The instructor began looking into offerings in this space in May of 2015, and concluded that at that time IBM Bluemix had the most complete and interesting set of NLP-related technologies. The discussion with IBM, and subsequent support for the authors from an IBM grant, resulted in moving the class to the Fall of 2015.

Class Curriculum: The class curriculum was directed towards Master-level students. As an introductory class, it was designed to provide both some understanding of modeling issues involved in natural language processing, as well as and introduction to NLP tools.

For the latter, in addition to Bluemix, the class used "Natural Language Processing with Java" by Richard M Reese, published by Packt in March 2015. This book was chosen since it contained overviews of major tools such as Stanford NLP and Open NLP, and the basic steps of setting a text processing pipeline. There were also mentions of Mallet and NLTK in the class, and the students were encouraged to use tools of their choice in the projects.

For the linguistic fundamentals, the class used several chapters of "Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax" by Emily M. Bender, published by Morgan & Claypool in 2013.

A few words on IBM Bluemix: IBM Bluemix is a cloud platform developed by IBM to build, run, deploy and manage applications on the cloud. Bluemix is based on Cloud Foundry, an open source platform as a service (PaaS), originally developed by VMware and now owned by Pivotal Software -- a joint venture by EMC, VMware and General Electric.

In the Fall of 2015 there were already multiple examples of Bluemix use in many industries. Thus in the energy sector, there were reports of using it for carbon emission data collection, distribution and utilities analytics. In finance, for micro-donations, gathering customer feedback and real time data mining of data from mobile banking apps. Other examples included healthcare and entertainment.

Our attraction to Bluemix came from a good set of NLP tools, and more broadly, an inspiring and growing set of cognitive computing online tools.

2. Using Bluemix in the classroom

In this section we describe our experience in using Bluemix in the classroom. In the next section we discuss example student projects that involved Bluemix. We envision these notes could also be used as a guide for other instructors.

IBM Bluemix account and setup: To create IBM Bluemix account a student needs to create an IBM ID first, and then register with the IBM Bluemix account. After signing into the Bluemix account they would get 30 days of free trial version, and this trial can be extended by using credit card or promotion code. Our student used promotion codes available through the IBM Academic Initiative which allowed student access for more than the semester.

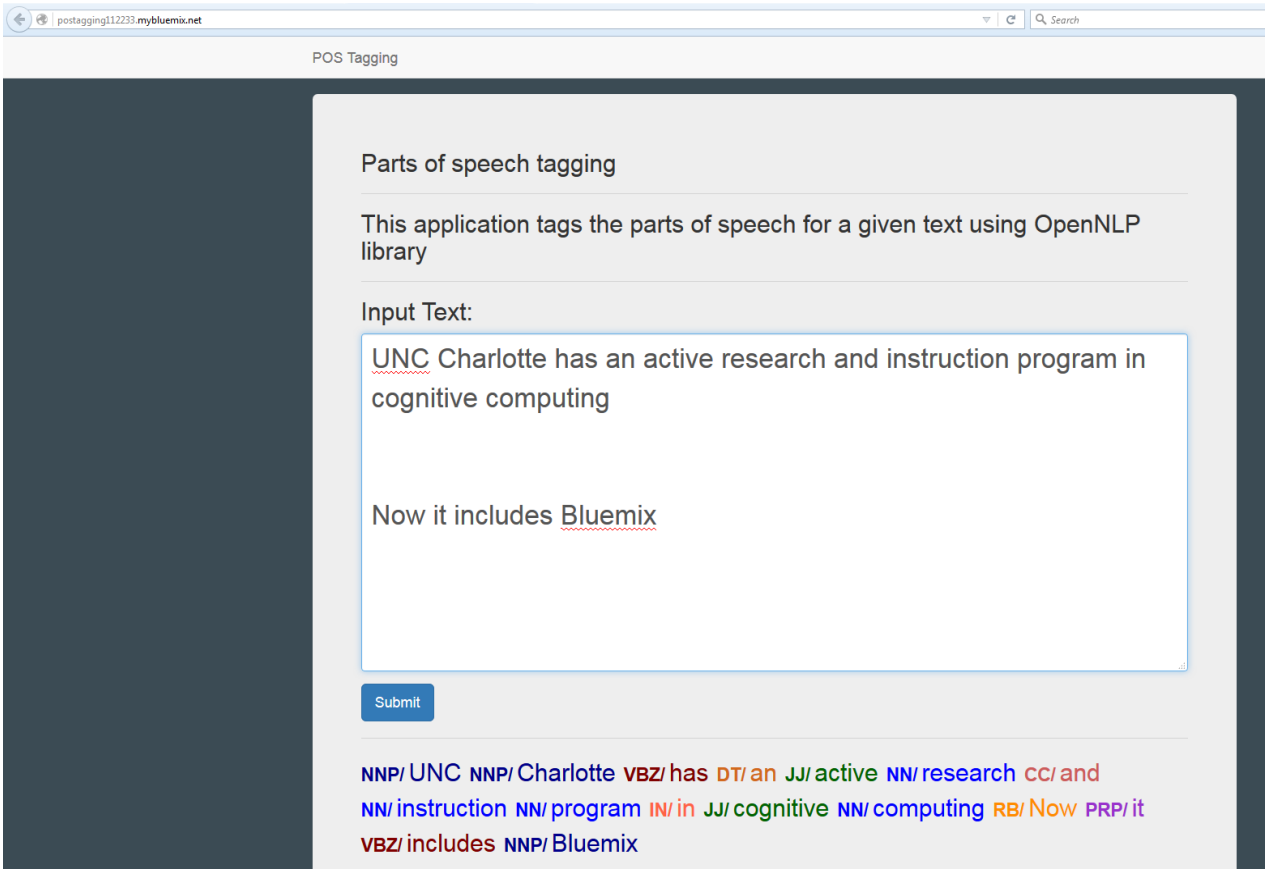
To start coding an application we have three alternatives: using GIT command line interface, Cloud Foundry command line interface, and Eclipse. Using any of these methods we can push our code to IBM Bluemix.

Using Bluemix during the class periods: We originally planned to work in the 'flipped classroom' mode, with plenty of hands-on assignments during the class periods. However, this turned out to be problematic for several reasons: not all students had the same background so the tempo was dictated by the slowest students; in addition when the class was trying to access a particular application, such as Node-Red, for some students was unable to access the application most likely due to limitations of Bluemix to start a collection of identical projects at the same time. Eventually we settled on a mixture of homework prep work and some in class project work and discussion.

Some the access difficulties were the growing pains of Bluemix. During the semester several changes and updates occurred. For example, initially AlchemyAPI was not available through Node-Red. Other features and nodes were introduce or disappeared from Bluemix. Despite these challenges the overall experience was positive, and the students finished several

interesting projects (described in the next section).

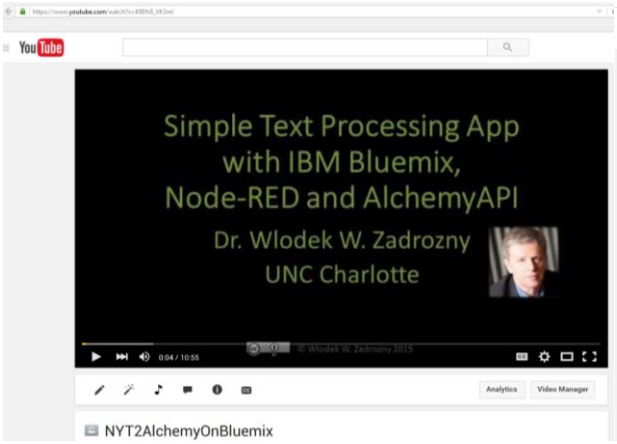
Example class and homework assignments: The assignment started with basic text processing tasks such as part of speech tagging, as shown in the figure below. More complex assignments included extracting semantic information from New York Times news (and storing it in a database), speech to text on a challenging audio, classifying 16th century literary works, comparing capabilities and accuracy of laptop/server and cloud-based NLP tools.

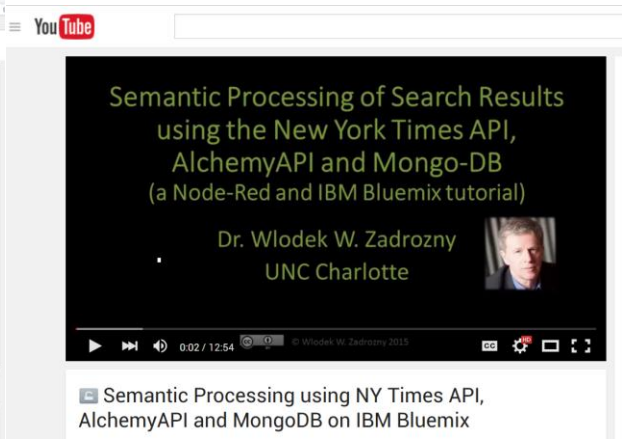


Video and text instructions: before assigning projects, we created instructions. We experimented both with text and video instructions. We usually used as a basis instruction available on IBM developers web sites, which were written from a particular developer or platform perspective, in different formats and were available on different sites. Overall most of these instruction worked fine, as a basis, but they had to be modified to either reflect a particular task or problems we encountered trying to follow them.

Written and video instructions from the class are being shared for others to use. Some of the written instruction are included in the Appendix sections of this report. The link to the videos are below. They cover basic semantic analysis using AlchemyAPI and storing the resulting data in a database on Bluemix.

- https://www.youtube.com/watch?v=K9Dh8_VK3mI
- https://www.youtube.com/watch?v=kDSO2bwMKss





Current work and future opportunities: IBM Bluemix remains, as of this writing, the largest and most interesting collection of cognitive computing apps available. Therefore we (WZ) are planning to use Bluemix in class projects again. We see the opportunity in both NLP-related and data science classes. However, we also feel it would be beneficial to future students to have an updated collection of instructional material available on the IBM website, preferably in somewhat uniform format.

3. Class Projects

Several students decided to use Bluemix for their final projects. Some these projects are described below using, slightly edited, the students' original descriptions.

PROJECT 1: US Elections 2016 Sentiment Analysis:

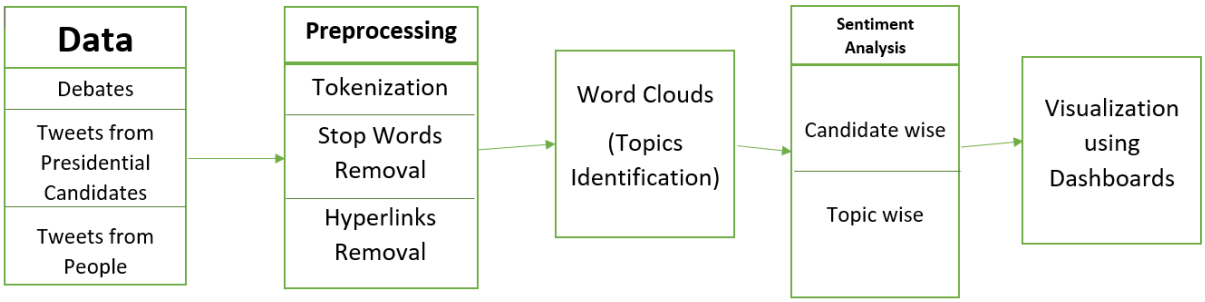
US Elections are one of the biggest events in entire world. The project aimed to analyze the sentiment of people towards the candidates and topics on which they were speaking, and to predict the overall attitude of the general population towards various candidates.

For this project, six most popular candidates from republican and democratic parties (three each) were identified. The candidates selected from Democratic Party were Hillary Clinton, Bernie Sanders and Martin O'Malley while the Republican Party candidates were Donald Trump, Dr. Ben Carson and Jeb Bush. Then the key topics on which these candidates spoke about were found using the data from their respective twitter handle and the debates. After identifying the topics, candidate's tweets were analyzed to find out their sentiment towards them, whether they were supporting those topics or opposing them. Then the general public opinion was found on the topics that candidates were talking about by using twitter data on those topics. Sentiment analysis was again performed on these collected tweets to find general public sentiment on popular topics. A detailed dashboard was then created for each candidate, which included topics on which they are focusing on, people's sentiment about those topics and

the candidate.

1. Data Collection Strategy:

There have been three republican and democratic debates each till now; these debates were used for initial identification of key topics along with the tweets of individual candidates. For analyzing people's sentiment the tweets of people focusing on each of the six chosen candidates, was collected for the time period from Dec, 1 2015 to Dec, 9 2015. There existed several millions of tweets by people focusing on Donald Trump. So US geography information was used for filtering the tweets.



2. Tools and Technologies used

(i) Services from IBM Bluemix:

Insight for Twitter: This service was used to collect data from Twitter using twitter handles of the presidential candidates and hashtags from people's tweets.
AlchemyAPI: This service was used to get the sentiment polarities for the tweets collected from Twitter.
dashDB: This service was used to store the tweets collected from Twitter along with their sentiment polarities.
(ii) Languages: R and Java.

(iii) APIs: Apache OpenNLP and LingPipe.

3. List of problems solved: The above analysis identified common topics of democratic and republican candidate. After identifying the sentiments, they were visualized to show how well this candidates have performed (positive and negative scores).

PROJECT 2: Speech Analyzer Program

"In the current day and age, many people are put to the task of giving speeches on a daily basis. (…). Since many students in college are given the task to present in various classes, it would be useful to write a program that could help individuals analyze their speech giving abilities.

"The first task was to modify the user's speech into a .txt format. The text file would then be used to conduct analysis on. This was done by writing a Ruby program that takes in the user's speech as a .wma format.

"Then, this Ruby program sends the audio file to the IBM's speech recognition software which returns a json file of the user's speech.

"This json file is processed by a java program. Clean.Json method converts the json file into a .txt file. The rest of the java program works with the text file that was produced from the Json file. A tokenizer was used to break the text into tokens (by whitespace). One of the elements that will be analyzed is the top five most used words. Many people have the bad habit of overusing words like: the, like, you, know etc."

Tools and Technologies used:

1. **Services from IBM Bluemix**

Speech to Text: Watson Speech to Text can be used anywhere there is a need to bridge the gap between the spoken word and its written form. This easy-to-use service uses machine intelligence to combine information about grammar and language structure with knowledge of the composition of an audio signal to generate an accurate transcription. It uses IBM's speech recognition capabilities to convert speech in multiple languages into text

2. **Languages**

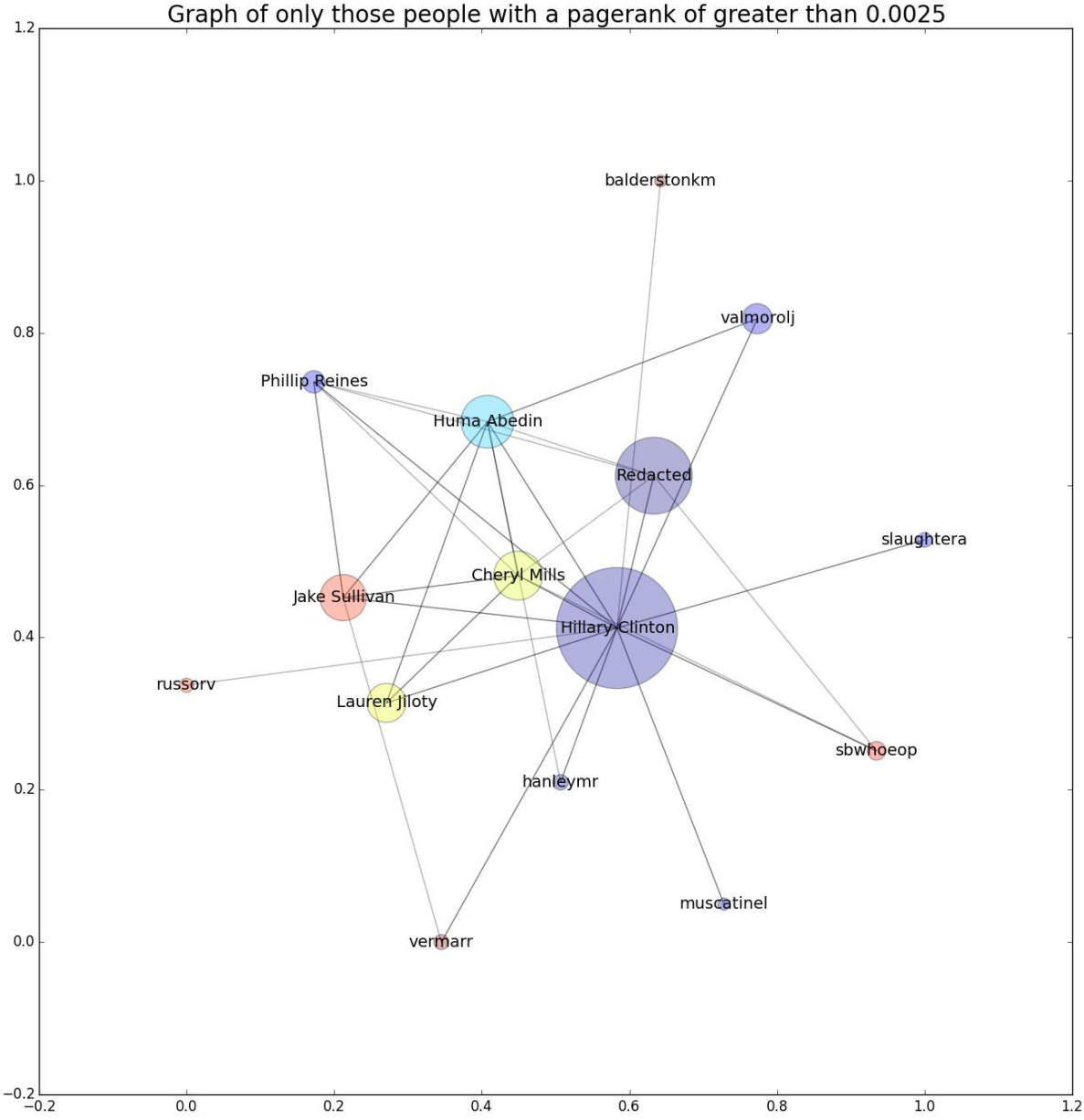
Ruby.

List of problems solved:

The students went through a process of trial and error. They started out with the idea of using a Java tool known as Sphinx to convert from speech to text. Although they were able to use it for basic commands, it was giving us a lot of trouble which large phrases: when the inputs size where multiple sentences, the accuracy of the speech recognition dropped dramatically. Then they have decided to use C# and Microsoft's speech recognition software. Although this proved to be more effective that using Sphinx, they still wanted to try another alternative. Finally, they decided to use the IBM speech recognition software. They observed that Watson was in general more accurate at speech to text conversion. To find a way to increase the accuracy of speech to text, the decided to increase the audio recordings quality: they used Audacity to record the audio files, and to remove background noise.

PROJECT 3: Hillary Clinton's Email Analysis

A controversy arose in 2015 when it was revealed publicly that Hillary Clinton had used for government business her personal email accounts on a privately maintained server. Some of the politicians and her opponents said that this is a violation of protocols and federal laws.



The students involved in this project analyzed the data set of Clinton's emails using NLP techniques; in particular to find the sentiments, word frequencies, and topic models.

They have gathered the data of email conversations of Hilary Clinton that have been released by FOIA. After cleaning the data, tokenizing, POS tagging and parsing, the data was converted into tabular format and inserted into a MySQL database.

With the help of NODE-RED, Java and Python the students tried to perform various functions like sentiment analysis, text mining, email graph, etc., to get insights. They visualized the results of the analysis using R, Python, D3 and Tableau.

Tools and Technologies used:

1. Services from IBM Bluemix

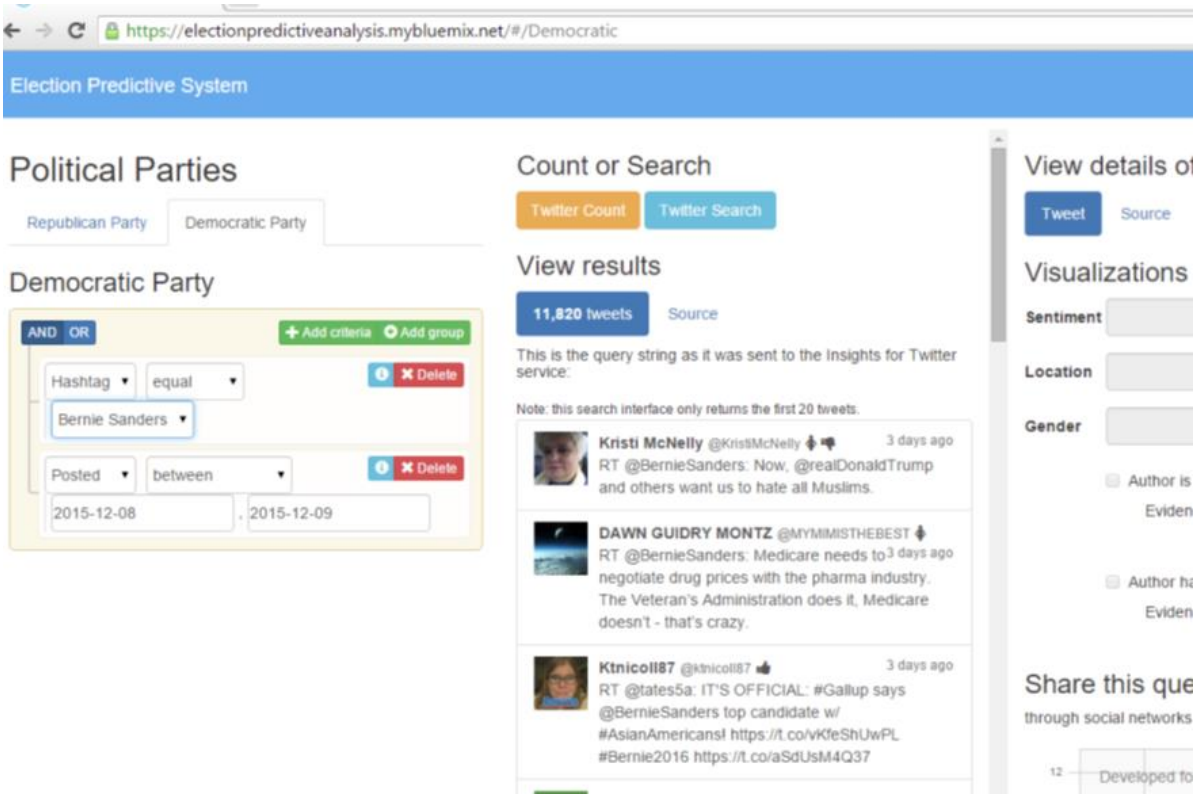
AlchemyAPI: This service was used to get the sentiment polarities for the text collected from Twitter.

Node-Red: Node-RED provides a browser-based flow editor that makes it easy to wire together devices, APIs, and online services by using the wide range of nodes in the palette. Flows can be then deployed to the Node.js runtime with a single click

PROJECT 4: Election predictive analysis using Twitter Feeds

In this project, students used real time Tweeter data to predict the likely candidates of 2016 elections from both the Democrats and the Republicans by taking the sentiment of the tweets expressed during the presidential debates on Twitter. Sentiment Analysis was performed on the tweets to determine the polarity of the tweets which maybe positive, negative or neutral.

The main idea of the project was the extract tweets from Twitter for a particular candidate using hash tags and keywords, perform sentiment analysis, and visualize it. Apart from visualizations the students also identified the candidate mentioned in the tweet using Named Entity Recognition (NER).



The dataset was obtained from Twitter using Twitter Insight API. The tweets were extracted with the help of keywords and hash tags. The extracted tweets needed to be preprocessed: the URLs, emojis and gifs were all removed from the tweets.

Tools and Technologies used:

1. **Services from IBM Bluemix**

Node.js: The IBM Node.js build pack stages the Node.js application to run in IBM Bluemix. The build pack reads the runtime and dependencies information that is defined in the package.json file of a Node.js application. The build pack also packages the runtime and dependencies of the Node.js application into a droplet that can be deployed to the cloud.

Twitter Insight API: Twitter streams can be incorporated into IBM Bluemix applications. The Twitter content store is refreshed and indexed in real-time. The Insights API service enriches Tweets with sentiments

Dash DB: Dash DB stores relational data.

Insight for Twitter: This service was used to collect data from Twitter using twitter handles of the presidential candidates and hashtags from people's tweets. . Insights for Twitter has RESTful APIs for customizing the searches; it returns Tweets and enrichments in JSON format.

Appendices containing various sets of instructions.

Below we present some of our IBM Bluemix instructions given to students. Most of the material is based on the IBM Bluemix instructions available at various IBM web sites. In some cases we added details. In other cases made minor tweaks and changes to make the programs work for us. Parts of these might be obsolete by now, and we make no claims that they should work for the reader. They are provided to illustrate the type of preparation that was required to make the class work. We also plan to use these later this year as a basis for a revised set of instructions.

In addition, as mentioned earlier, we created two sets of video instructions. They cover basic semantic analysis using AlchemyAPI and storing the resulting data in a database on Bluemix.

- https://www.youtube.com/watch?v=K9Dh8_VK3mI
- https://www.youtube.com/watch?v=kDSO2bwMKss

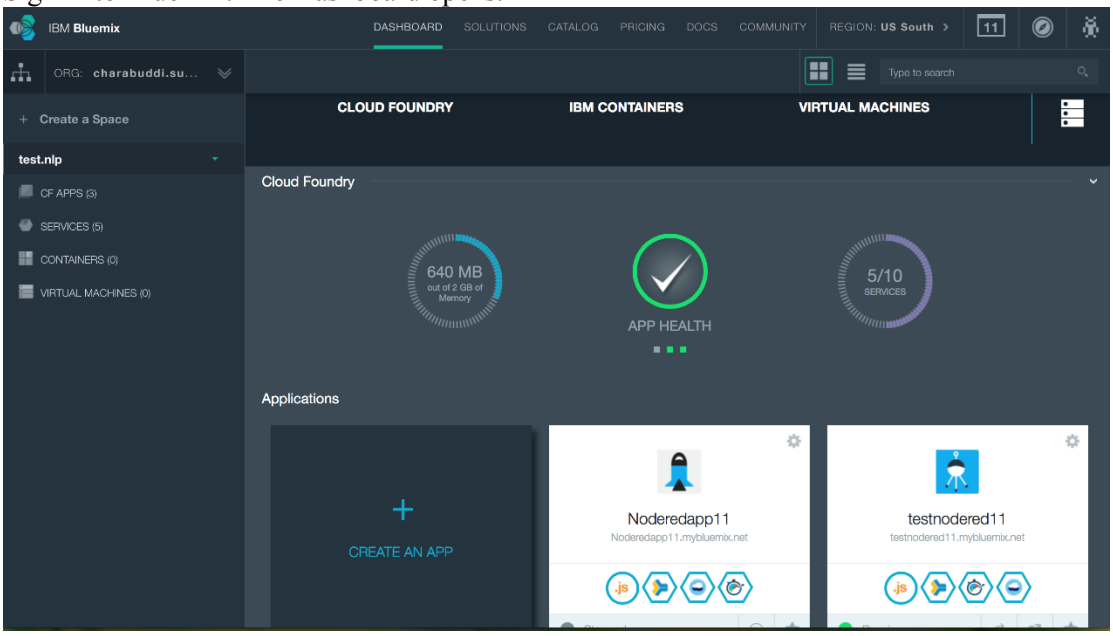
Appendix 1. Initial Instructions

How to create an IBM Bluemix account? https://console.ng.Bluemix.net/registration/

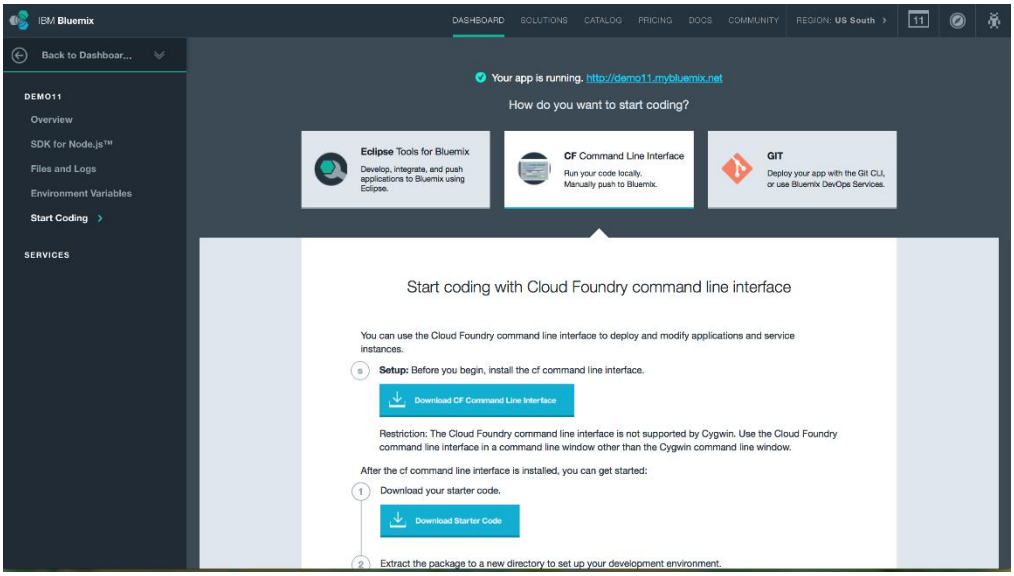
Demonstrating a simple application

To get started now, try this tutorial. You'll create an app and deploy it.

1. Sign in to Bluemix. The Dashboard opens:



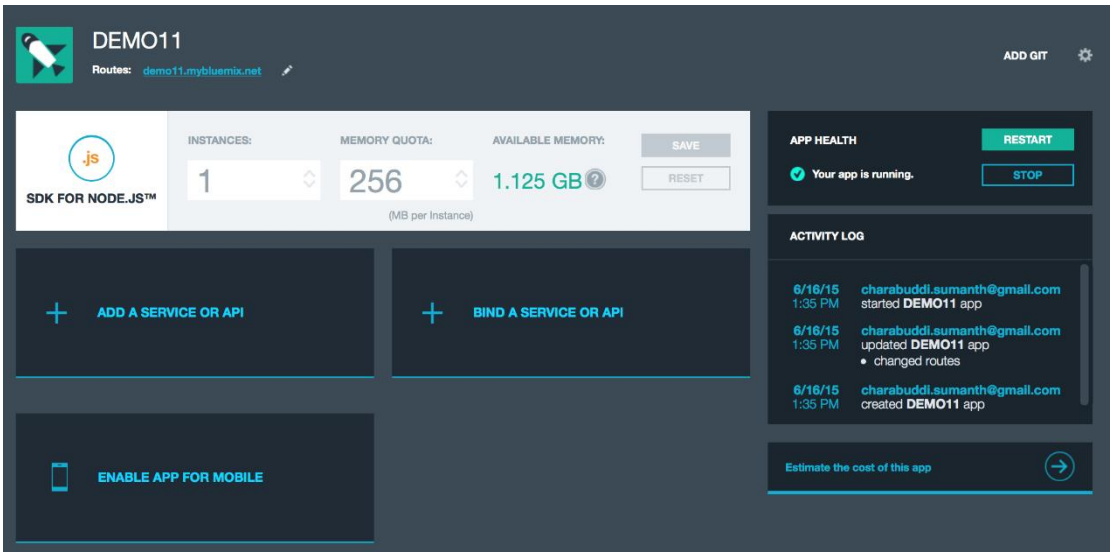
2. The Dashboard shows an overview of the active Bluemix space for your organization. By default, the space is dev and the organization is the user name of the person who created the project
3. Click **CREATE AN APP**. For the kind of app that you are creating, click **WEB**.
4. For your starter, click **SDK for Node.js**. Review the docs and details, and then click **CONTINUE**.
5. Name your app and then click **FINISH**. The name is a unique URL where you access your app. After a moment, the app starts. The app's Overview page opens and shows that the app is running.



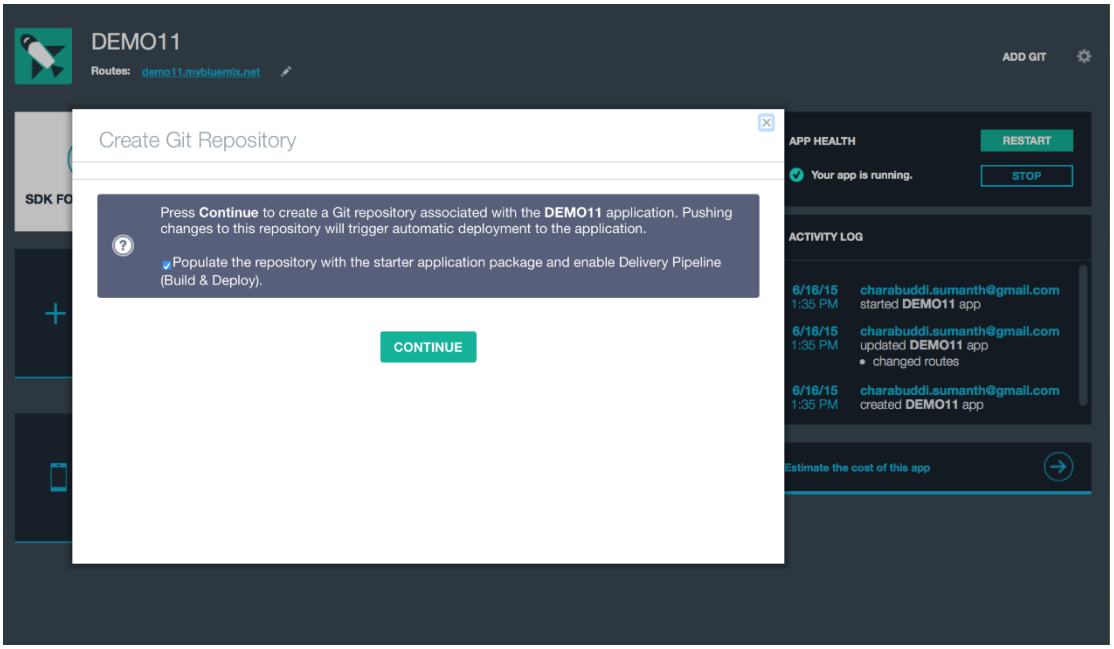
6. To make any modifications in the code you can do it through eclipse, cf command line or GIT CLI.

GIT CLI:

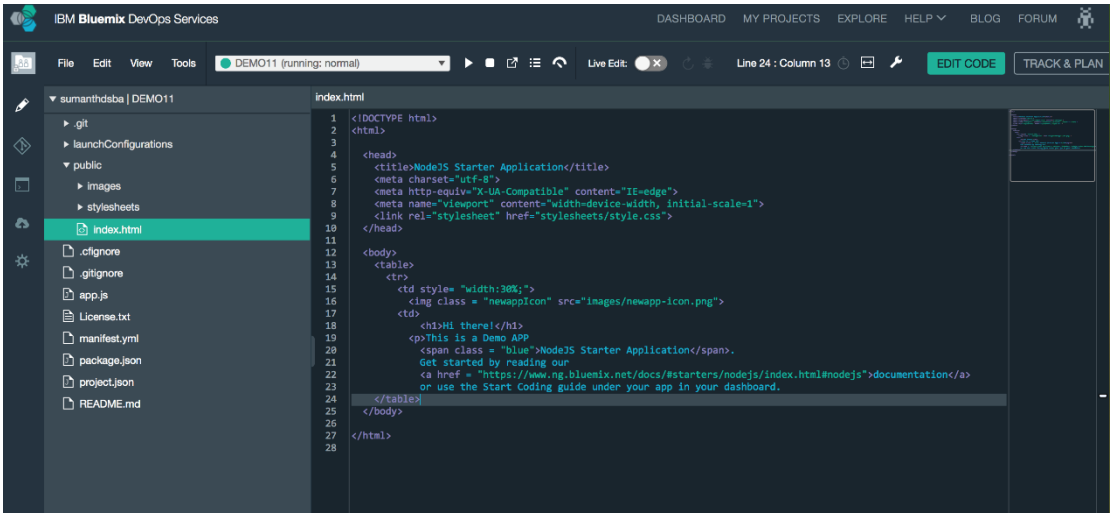
1. To enable auto-deploy, go to your app's Overview page on the Bluemix Dashboard and click **ADD GIT**. A Git repository is created and is populated with example code and a deployed app. You might be prompted to enter your password to verify that DevOps Services can act on your behalf.



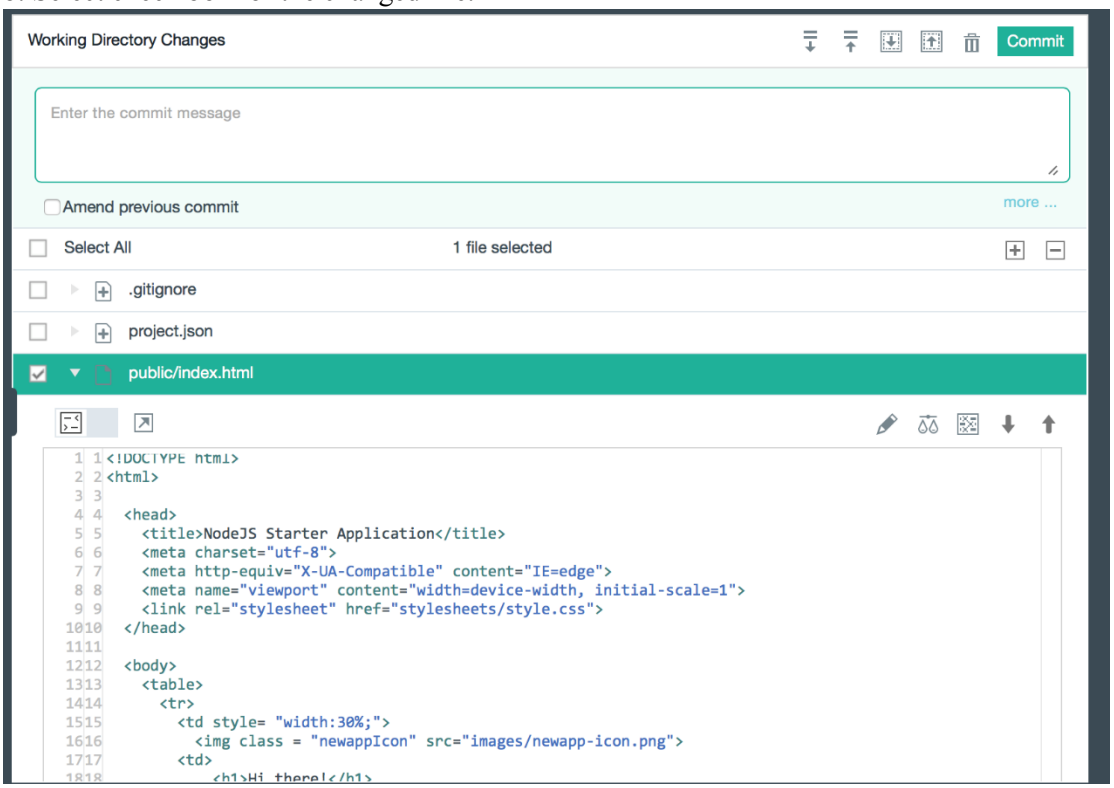
2. Make sure that the **Populate the repository with the starter application package and enable build and deploy** check box is selected.



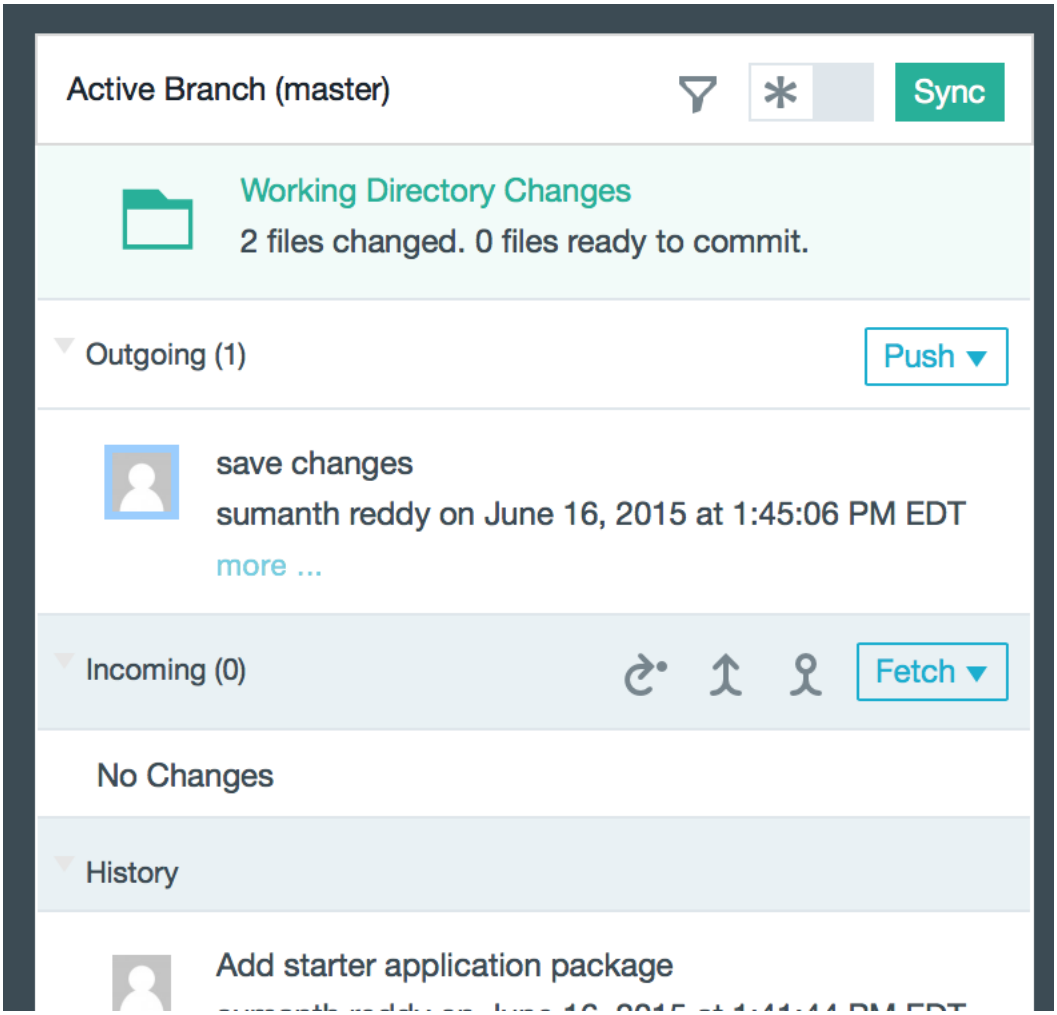
3. You created a Git repository, populated it with the example code, and deployed the app
4. Go to the app's Overview page and click **EDIT CODE**. Your new project opens in the web integrated development environment (Web IDE).
5. In the directory, find a file to modify; for example, public/index.html.
6. Edit the file in the editor



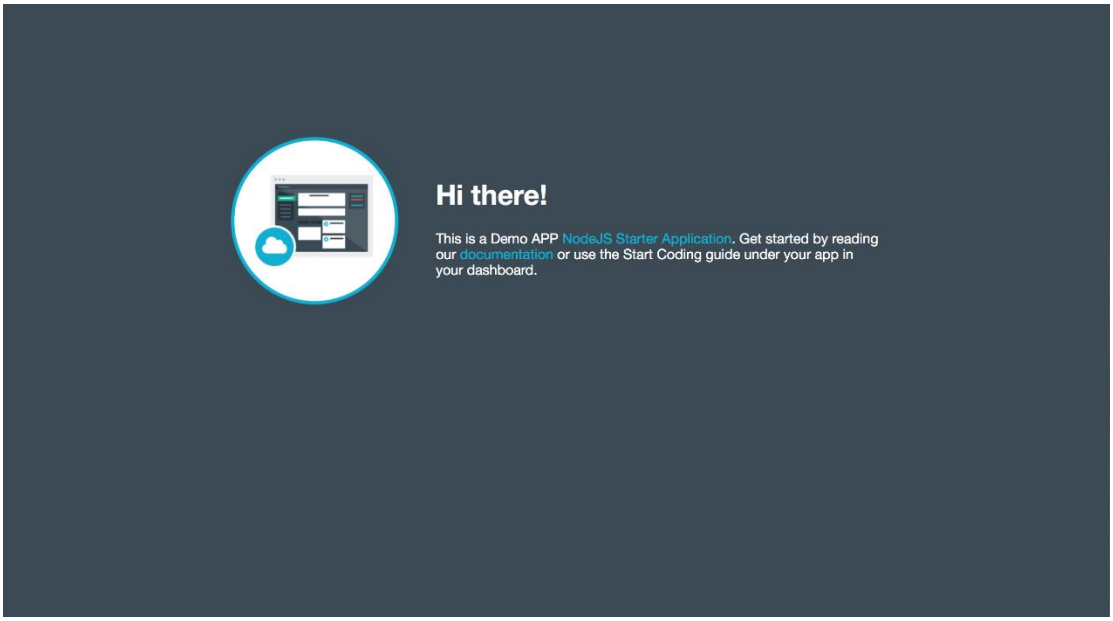
7. Push the change by using the integrated Git support.
   a. From the leftmost menu, click the **Git Repository** icon .
   b. Select check box for the changed file.



8. Type a commit message, and then click **Commit**.
9. In the Outgoing section on the left, click **Push**.

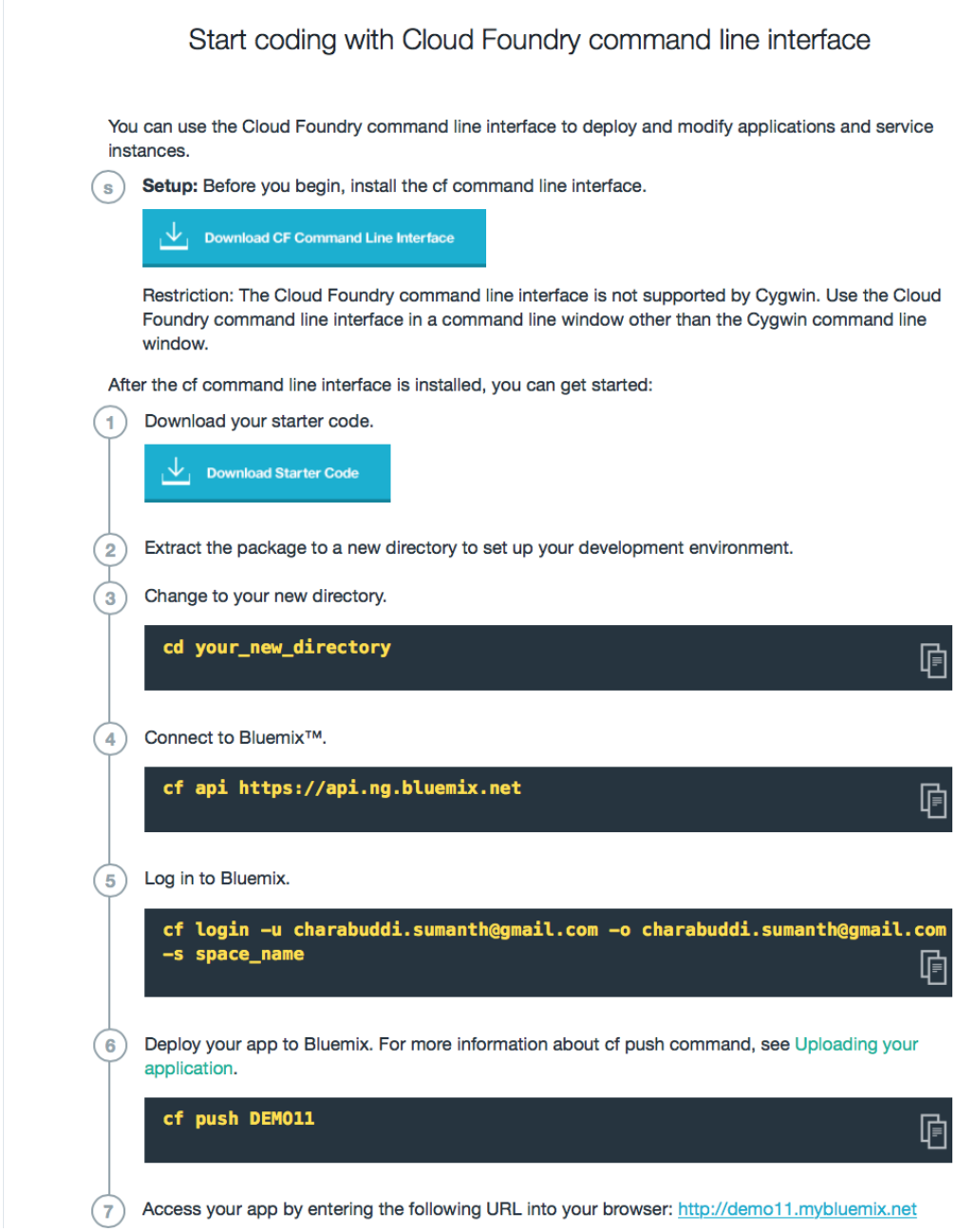


10. Your update is shown



CF Command line:

1. We will be using the CF command line interface for this project. You can download the starter code given in start coding.
2. And then you can make the modifications to the code and then push the code into blue mix.



From Eclipse Tools

Learn to install the IBM Eclipse Tools for Bluemix. A Java™ 1.7 or later Execution Environment is required.

There are multiple ways to install the IBM Eclipse Tools for Bluemix, including:

- Installing from the Marketplace.

- Installing from WASDev.
- Downloading from the IBM WebSphere Application Server Developer Tools (WDT) Installer.

Installing from the Marketplace

This requires Eclipse Luna for Java EE Developers (4.4.1), Eclipse Kepler for Java EE Developers (4.3.2), orEclipse Mars (4.5).

1. Open Help > Eclipse Marketplace. Search for Bluemix.
2. Select the IBM Eclipse Tools for Bluemix entry and click **Install**.
3. By default, there are features selected for you. Click **Confirm**.
4. Accept the license agreement and click **Finish**.

Installing from WASDev

1. Open the Download page in WASDev.
2. Drag the Install icon to the toolbar in Eclipse.
3. By default, there are features selected for you. Click **Confirm**.
4. Accept the license agreement and click **Finish**.

From the IBM WebSphere Application Server Developer Tools (WDT) Installer

Downloading from the IBM WebSphere Application Server Developer Tools (WDT) Installer:

1. Open Help > Install WebSphere Software. Search for Bluemix
2. Select the IBM Eclipse Tools for Bluemix entry and click **Install**.
3. By default, there are features selected for you. Click **Confirm**.

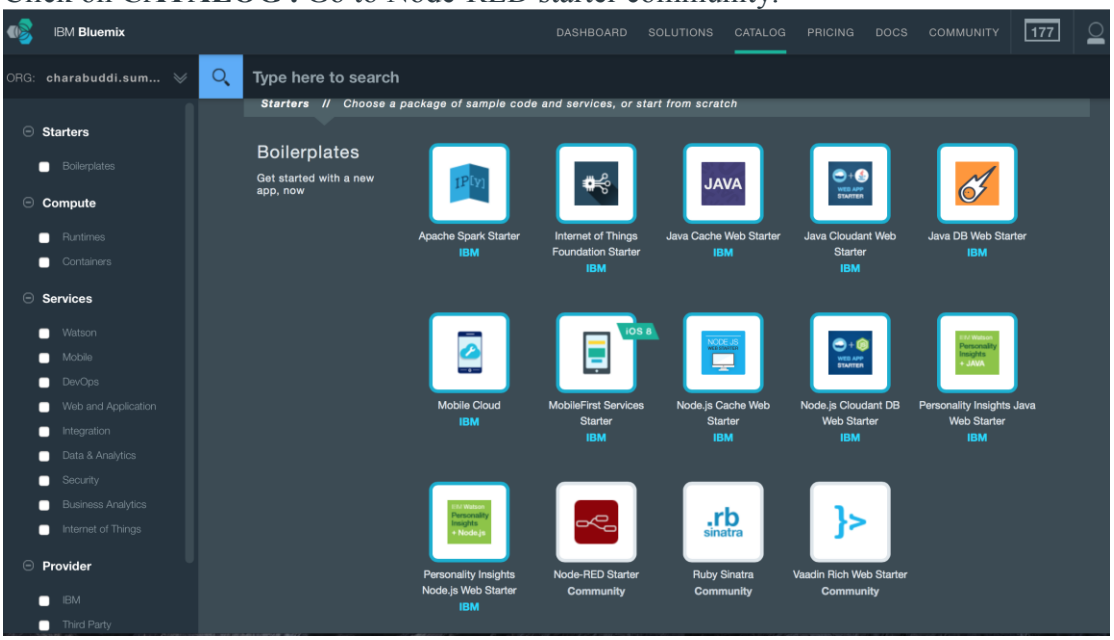
4. Accept the license agreement and click **Finish**.

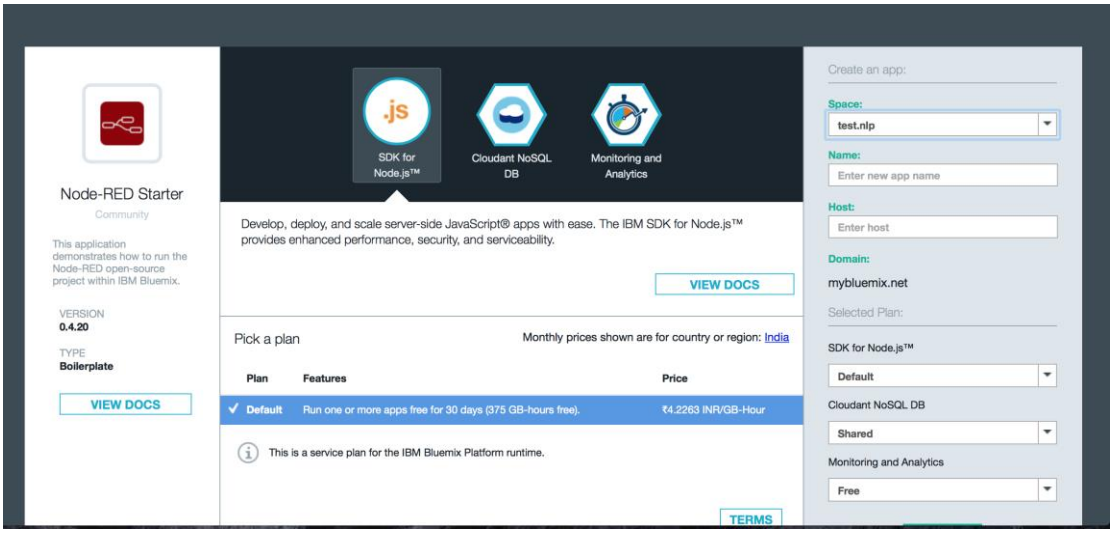
Appendix 2: Instructions: Extraction from NYT using Alchemy and Node-RED

In this tutorial, you'll create the title app and deploy it.

1. Sign in to IBM Bluemix. The Dashboard opens.

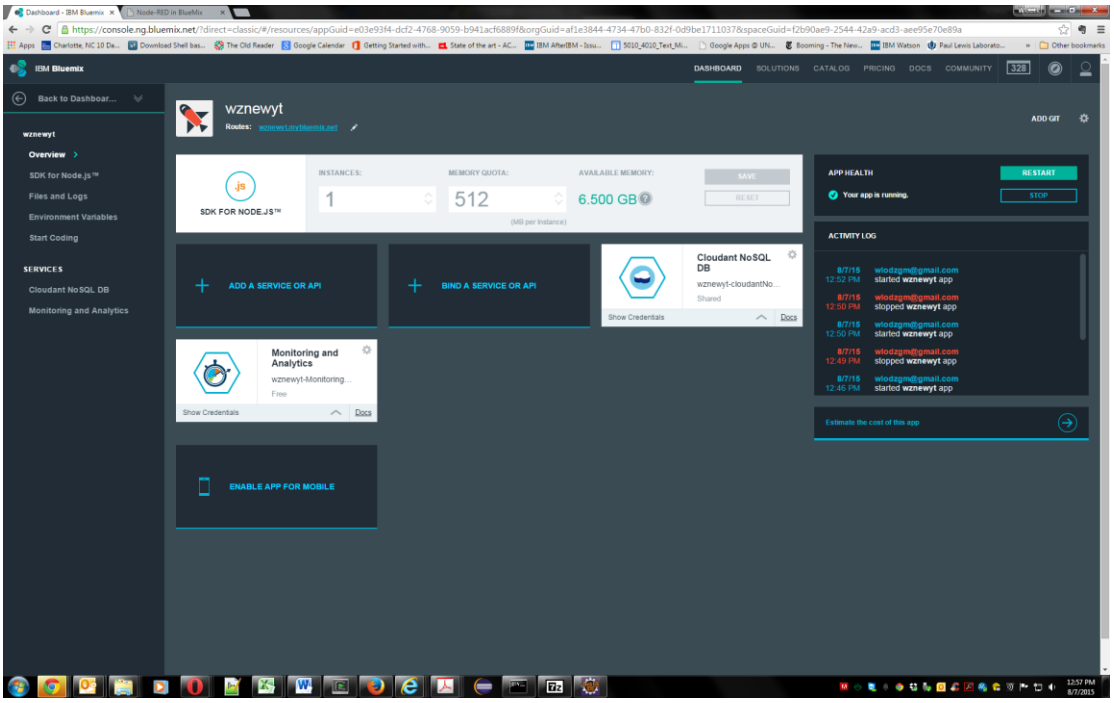
2. The Dashboard shows an overview of your active Bluemix. By default, the space is "dev".
3. Click on **CATALOG .** Go to Node-RED starter community.



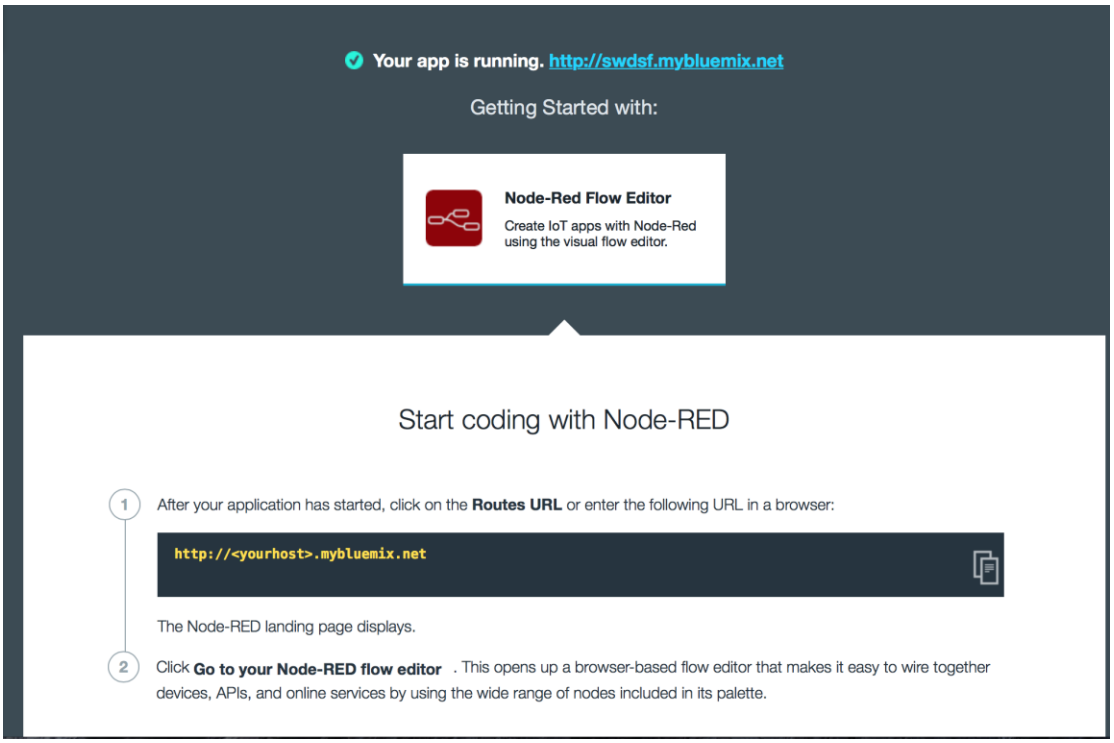


4. Name your app (we use “swdsf” ‘wznewyt’ in this tutorial) then click **FINISH**. The name is a unique URL where you access your app. After a while, the app starts. The app's Overview page opens and shows that the app is running.

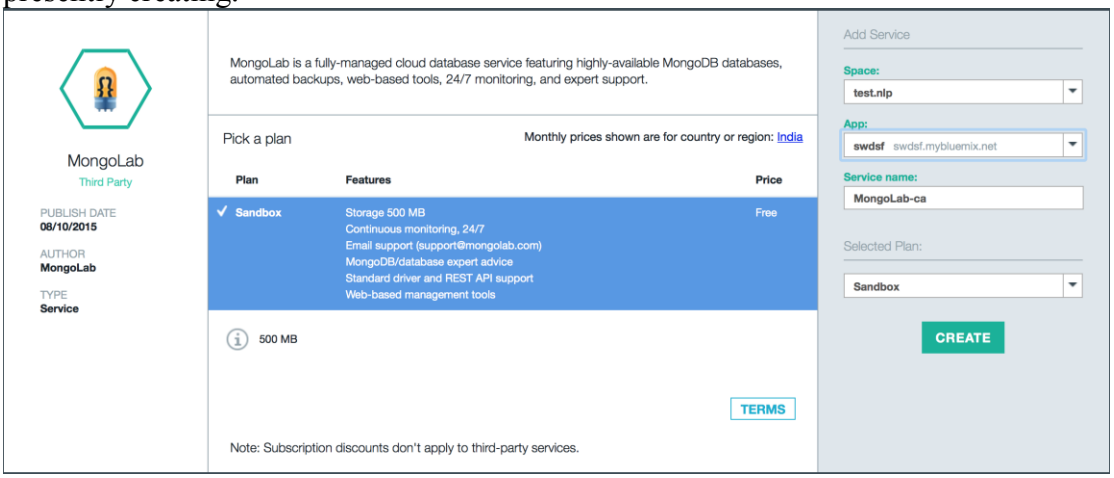
   Comment: It might take about 2-10 min to open the overview page. Sometimes “restart” helps. But it still might take 5-7 min from pushing “restart” to getting the message “Your app is running”



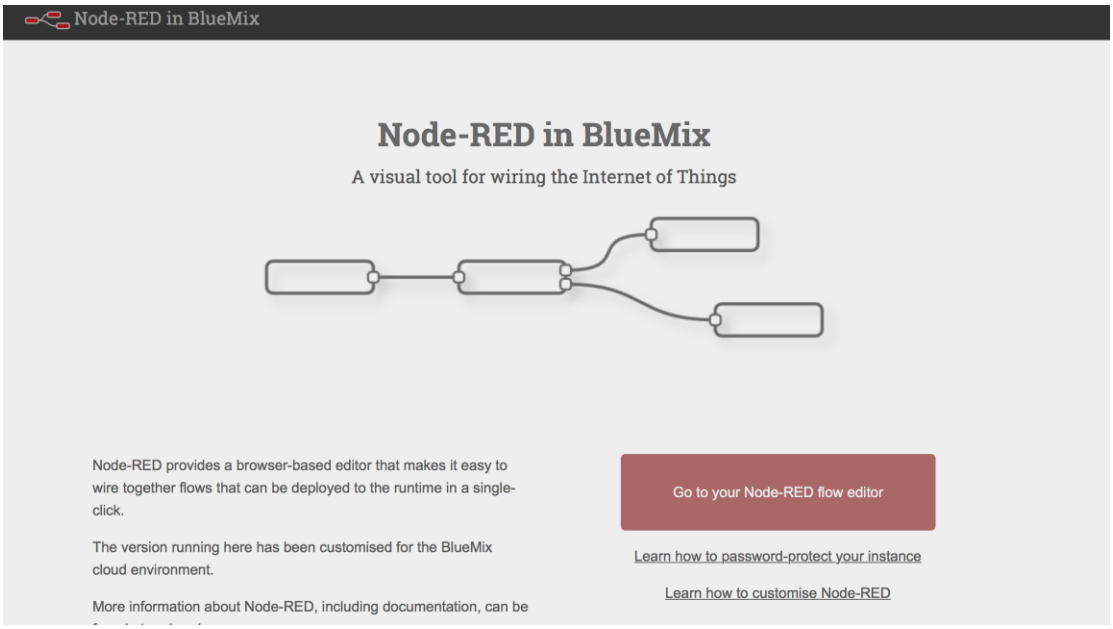
Note: From the above screen you can to click on the name of your app (in azure color under the short name) to get to Node-Red Flow Editor



5. Go to CATALOG and then add the service “MongoLab” to the application that you are presently creating.



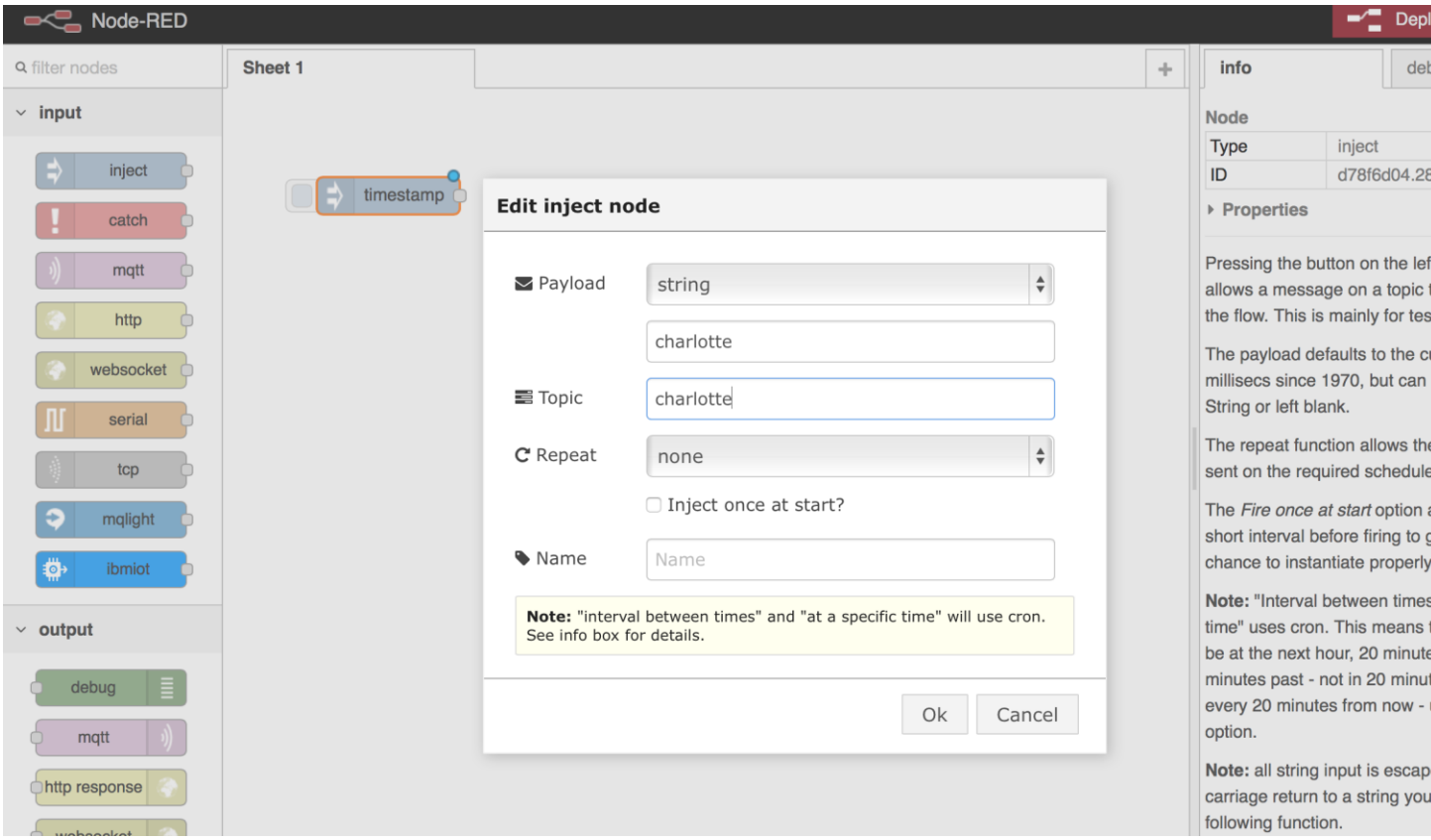
6. Click on the link where the app is running.



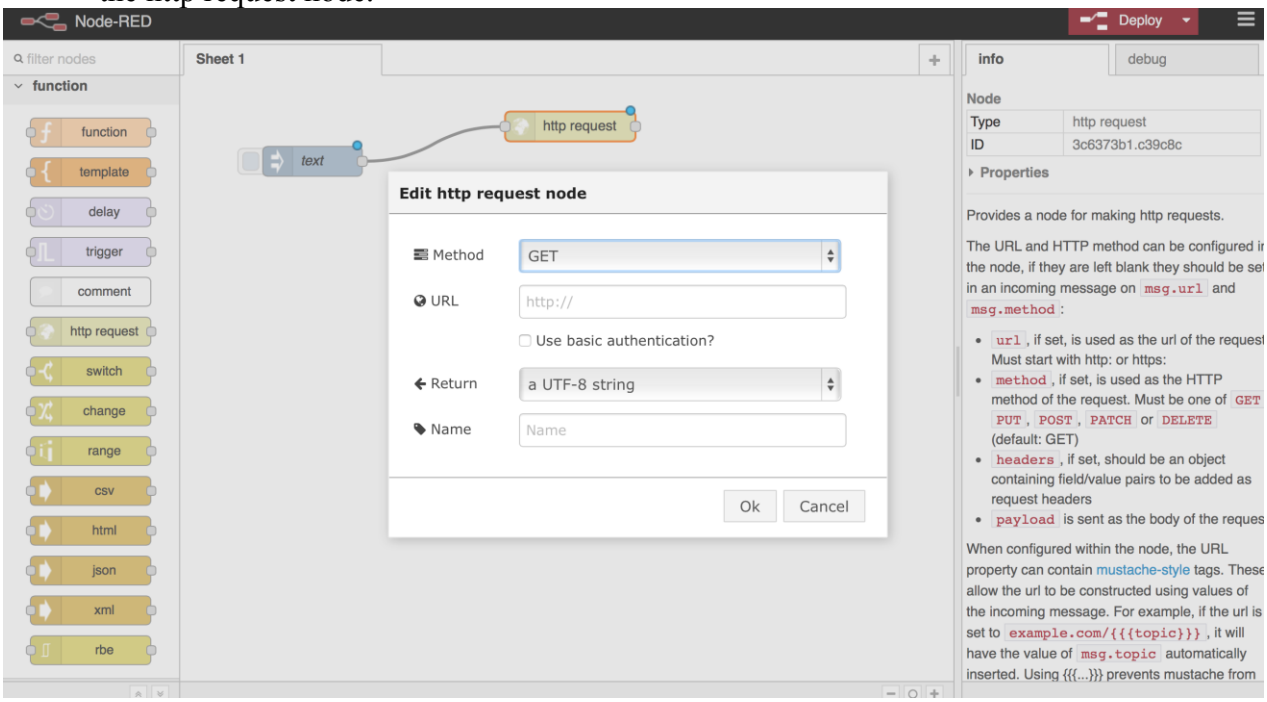
7. Click on Go to your Node-Red flow editor.

8. The left side of the screen contains all the nodes.

9. Insert the **inject** node into the worksheet. Put string(s) you want to search for in the New York Times.



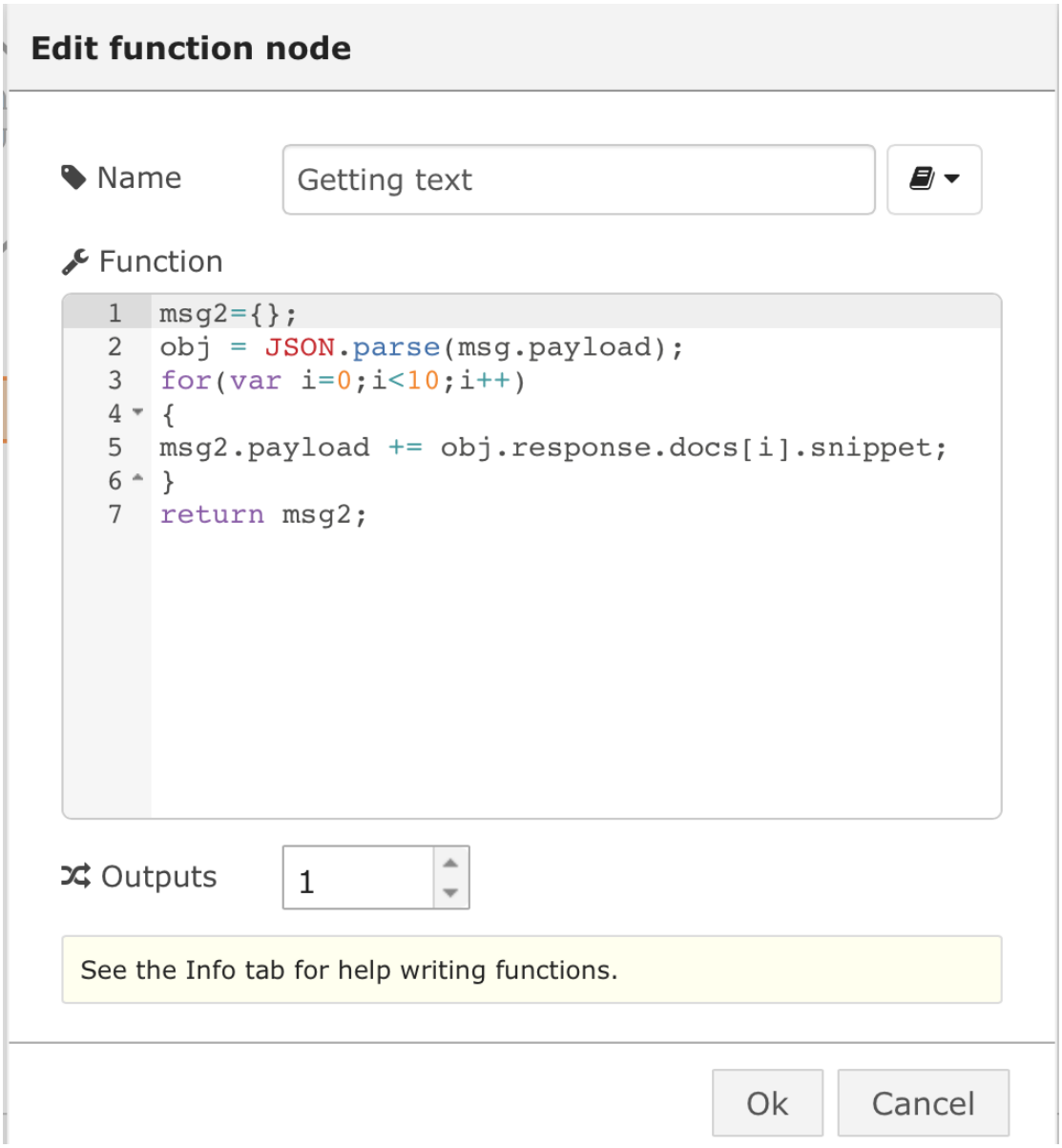
10. Insert the **http request** node (from category “functions”) and connect the inject node to the http request node.



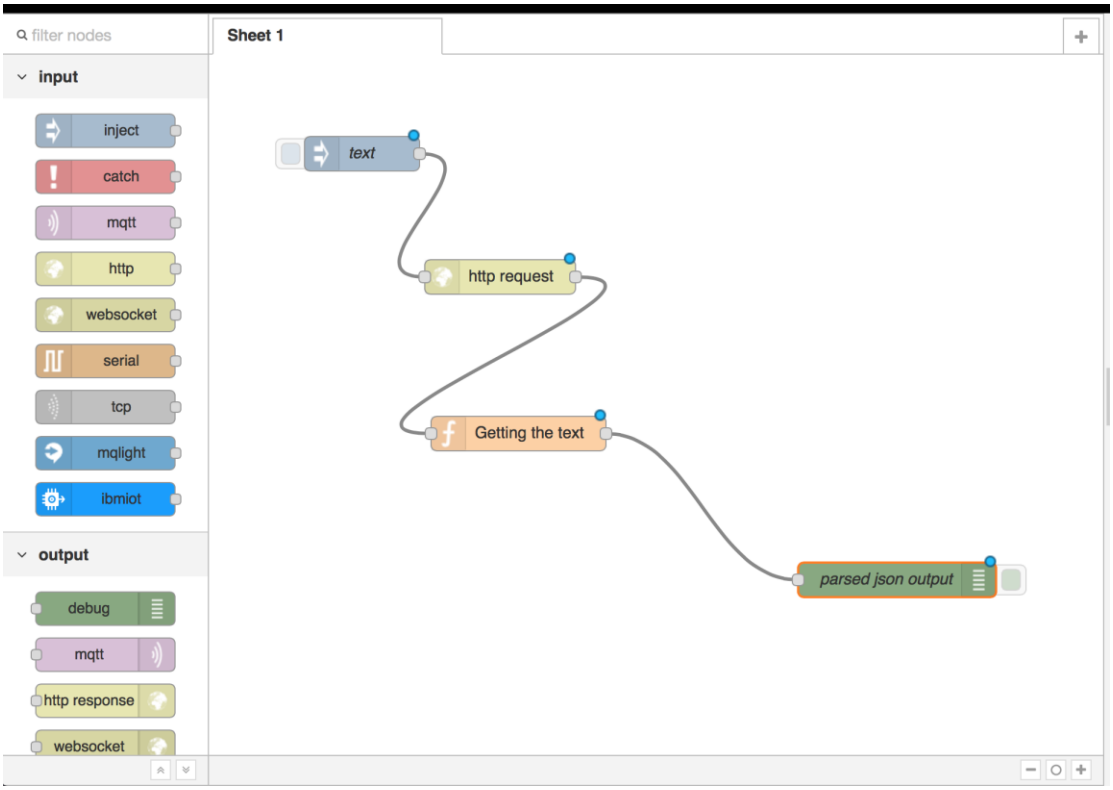
11. Configure the http request node with the following url link:
    http://api.nytimes.com/svc/search/v2/articlesearch.json?q={{payload}}&api-key=sample-key
    Note. You might want to register with the NY Times and get your private key, if you plan to build more than this example app using NYT data.

12. Insert the **function** node into the worksheet, and name it, e.g. “Getting text” or “getText”. Configure the function node with the following code.

Note. We use “10” records as a default, but you can take all available information:
By setting “i < obj.response.docs.length”.



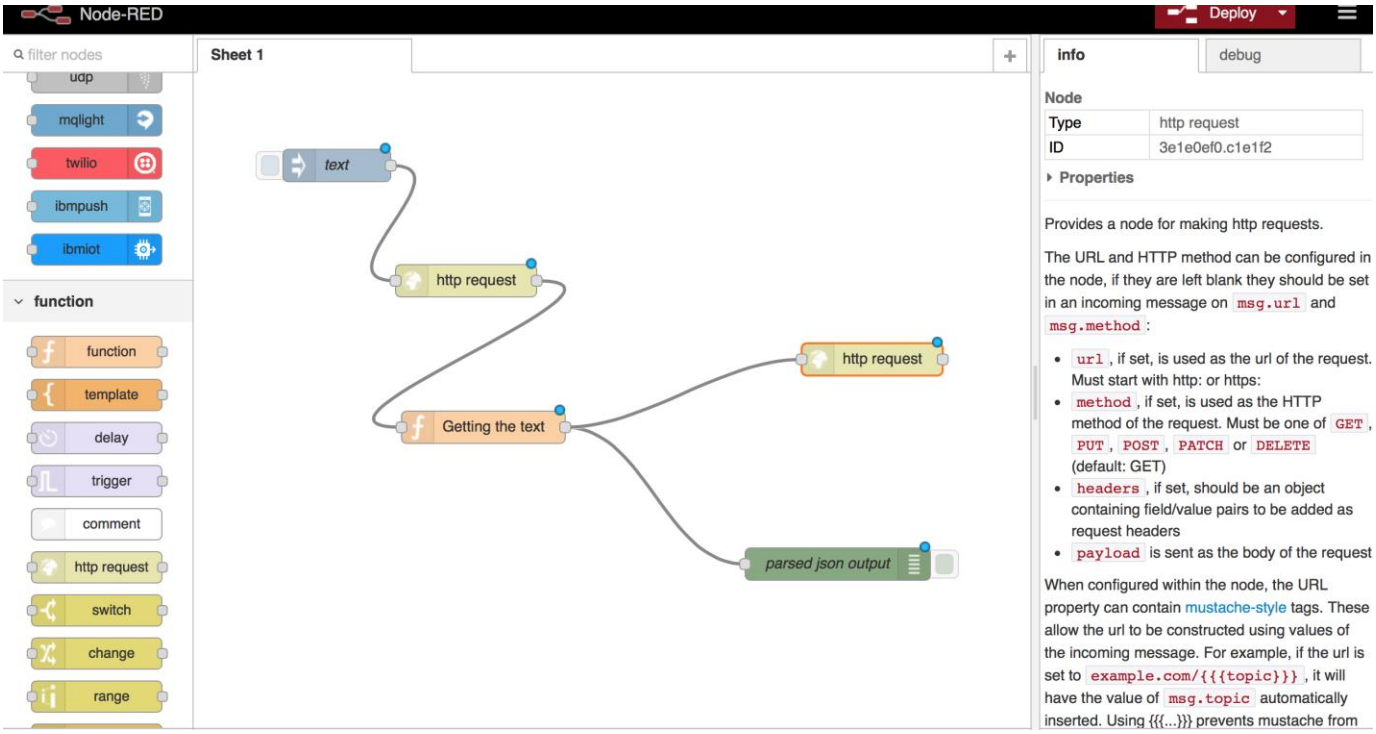
13. To know whether we are getting the output or not. Insert the **debug** node and connect it to the function node.



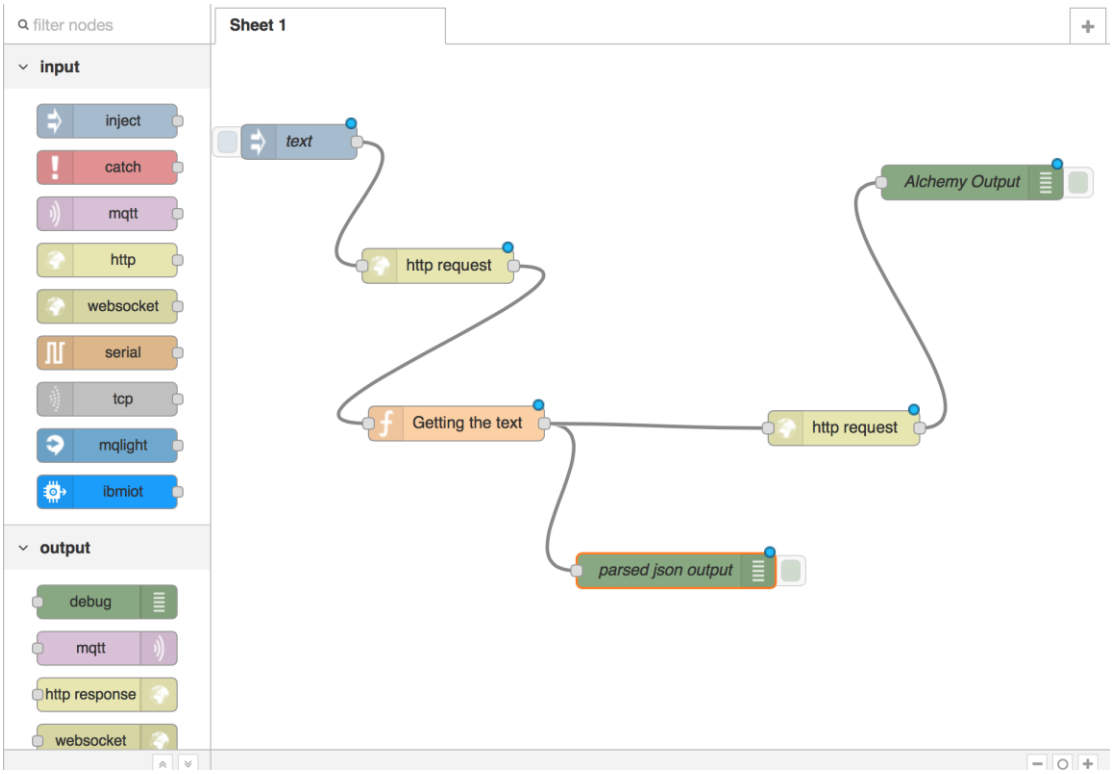
14. Insert another **http request** node and connect it to the function node. We will know get the data from the function node and the send the data as a request to the AlchemyAapi
Note: You can put debug statements after any node. Debug allows you to see the "payload" (property) or the whole object. In this tutorial, usually, you want to see the "complete message object".

15. Get the AlchemyApi key by registering your detail in the alchemy website.
http://www.alchemyapi.com/api/register.html

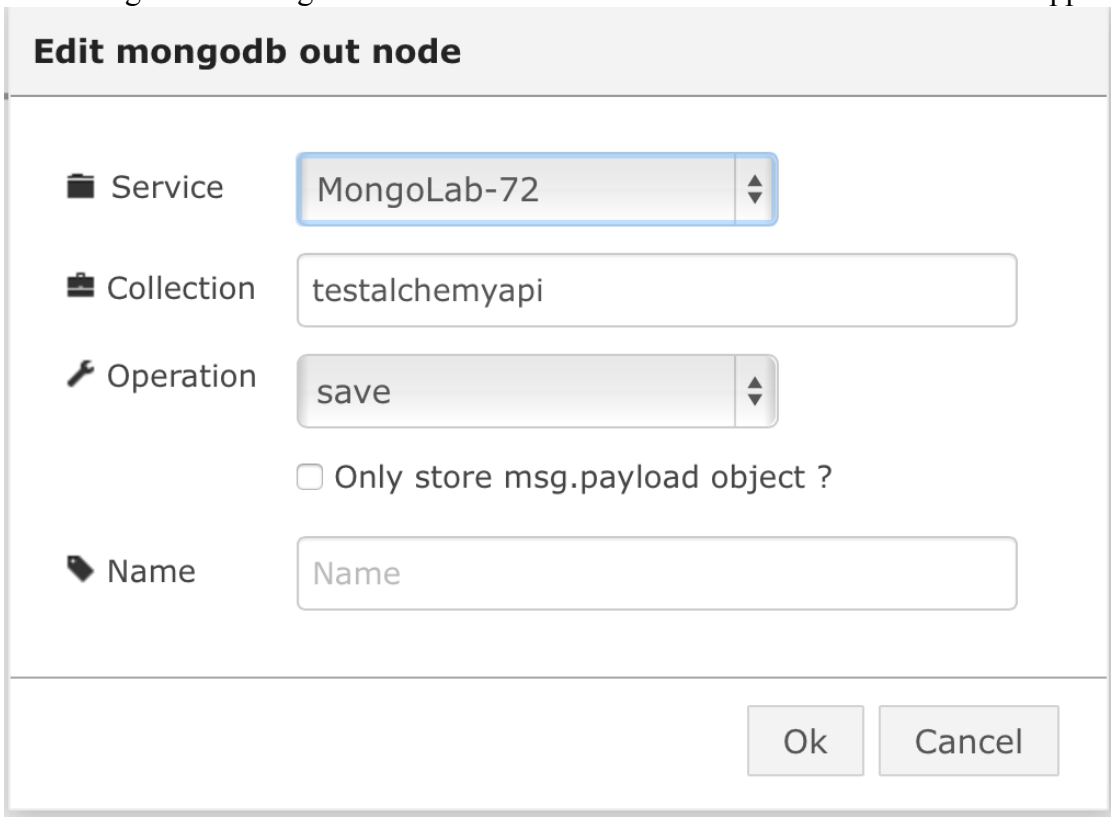
16. Configure the http request node with the following link.
http://access.alchemyapi.com/calls/text/TextGetRankedNamedEntities?apikey=<**apikey**>&text={{payload}}&outputMode=json&showSourceText=1&quotations=1&sentiment=1
Note: Replace the **apikey**, with the key your got.



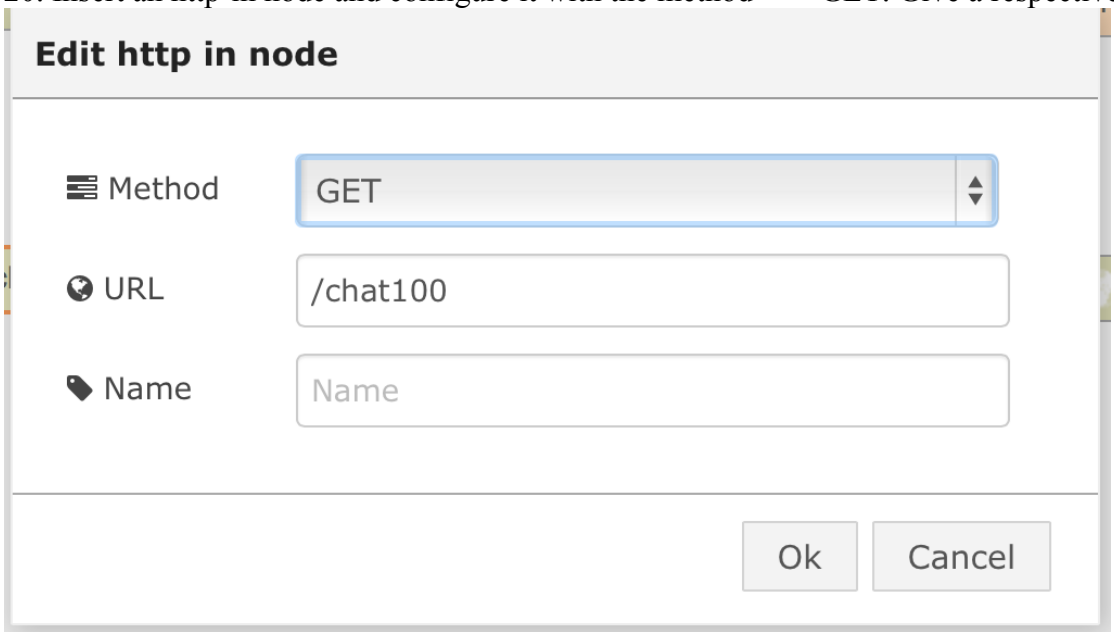
17. Insert the **debug** node and connect it to the http request node.



18. Connect the http request node to the MongoDB node. So that we can store the data into the MongoDB.
19. Configure the MongoDB node with the service that is bind at the start of the application.

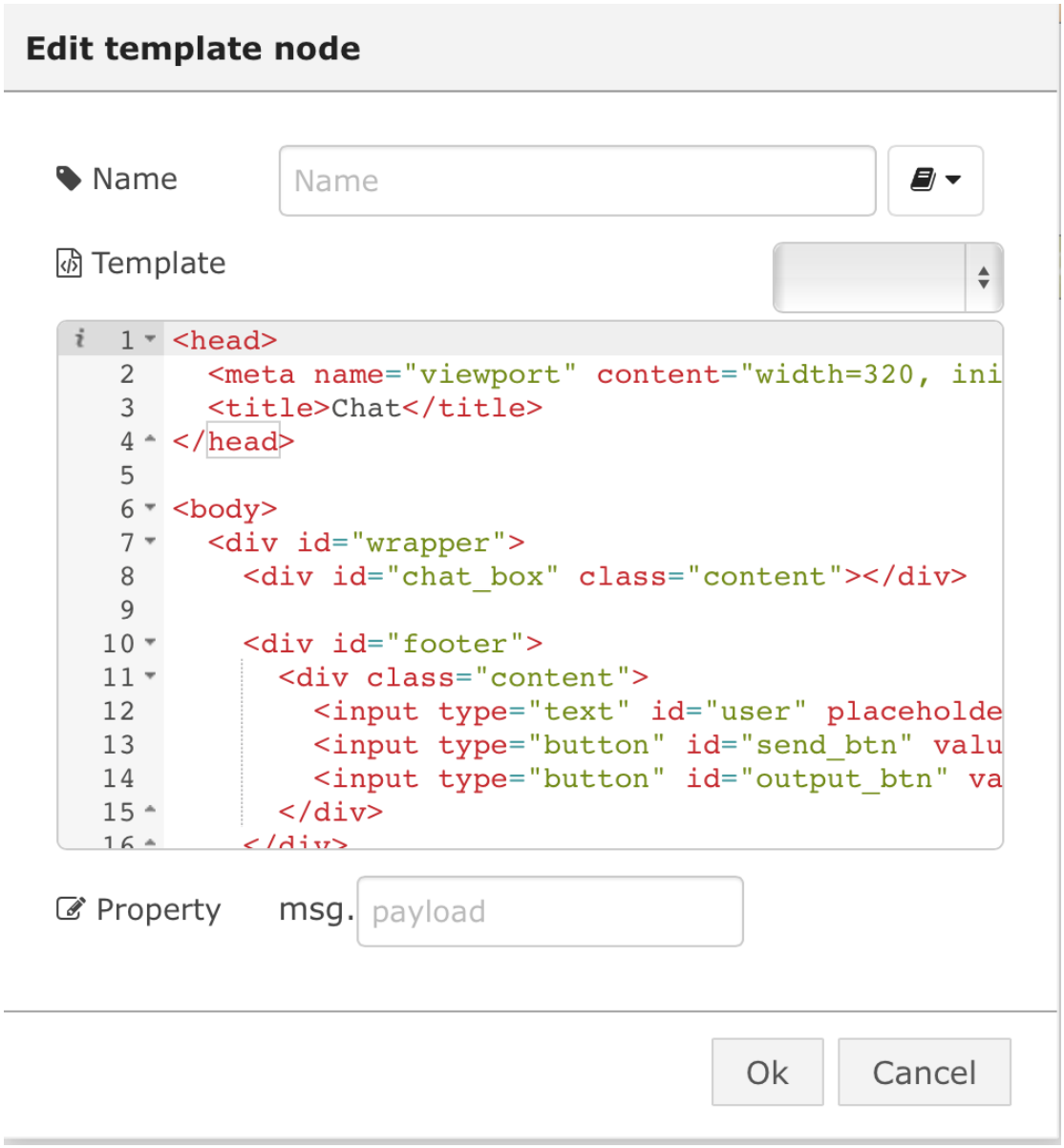


20. Insert an http-in node and configure it with the method GET. Give a respective URL.

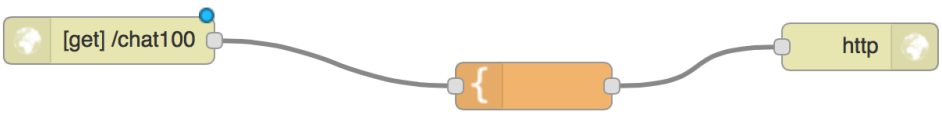


21. Connect a function node to the http-in node, which contains the HTML code in it. Which will be displayed at the URL that is given in the http in node.

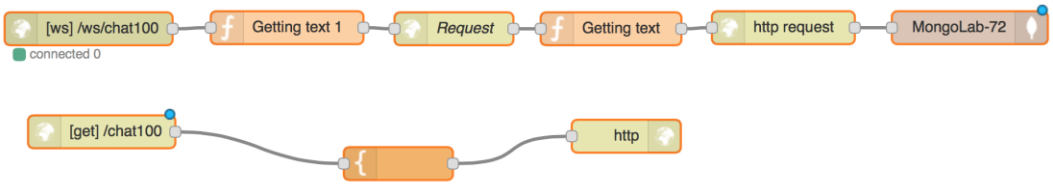
Note: Please find the attached HTML code at the end of the document.



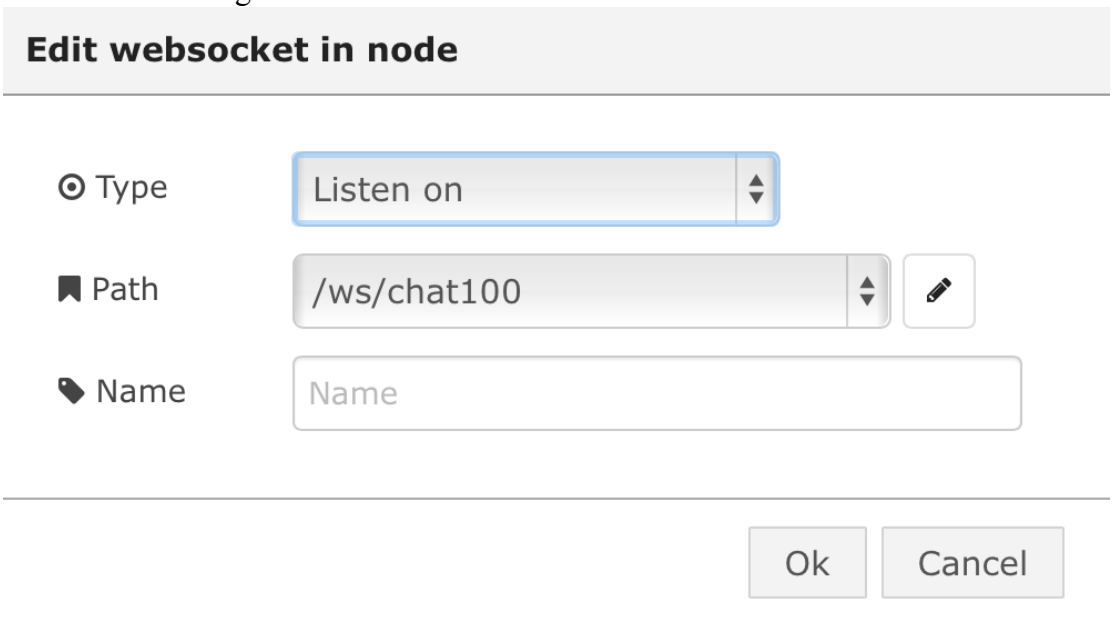
22. Then connect it to an http response node.



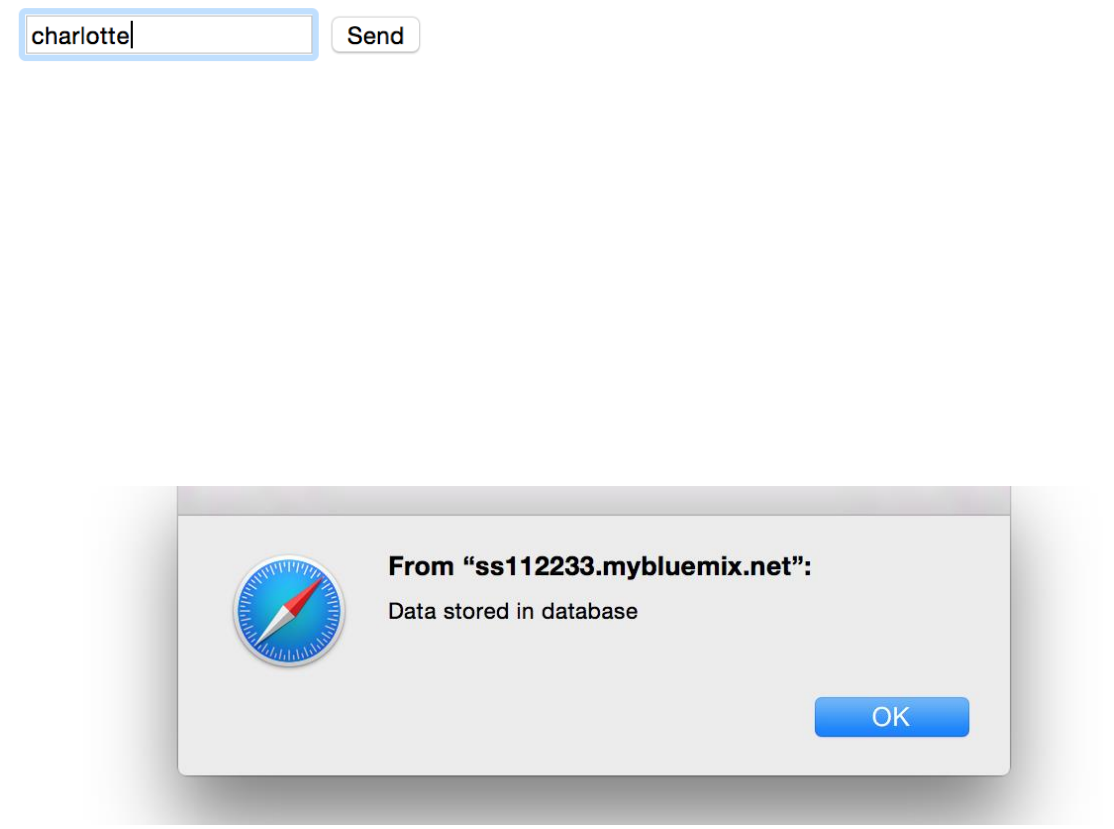
23. Connect web socket to the starting of the node. Remove the Inject node and connect the web socket node to the http request node.



24. The web socket here is "/ws/chat100". It listens to the calls that are made from /chat100.configure this websocket.



25. Click on the **Deploy**, to save the changes. (You can save/"deploy") multiple times during editing.
26. Go to <application_name.Bluemix.net>/chat100 in the browser. As we have mentioned chat100 as the page in http in node.




```
        "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": "",
        "totalTransactions": "2",
        "language": "english",
        "text": "undefinedArticles, photos and video about Charlotte from The New York Times, including hotel, restaurant and attraction information
 with reader reviews and advice on where to stay, where to eat and what to do.Prosecutors challenged the credibility of Officer Randall Kerrick as he
 testified during his trial on voluntary manslaughter charges in the death of an unarmed black man, Jonathan Ferrell.A portion of the dashboard camera
 video released by the Charlotte Police Department on Wednesday shows Jonathan Ferrell moments before he was shot by Officer Randall Kerrick.\u201cWe
 have exhausted every possibility,\u201d said the jury foreman in the trial of Officer Randall Kerrick, accused of using excessive force in the shooting
 of Jonathan Ferrell.The officer, Randall Kerrick, faces up to 11 years in prison if he is found guilty of using excessive force in the death of
 Jonathan Ferrel.A white police officer testified that he repeatedly fired his gun at Jonathan Ferrell, an unarmed black man, because the man charged at
 him repeatedly and he did not think his weapon was working.The video was shown in the trial of the Charlotte officer who fatally shot a black motorist
 who had been seeking help in 2013 after an accident.The fatal shooting of Jonathan Ferrell in September 2013 occurred amid the beginning of the
 national debate over the deaths of young black men that followed the killing of Trayvon Martin in 2012.RICH--Charlotte, a beautiful woman, died
 peacefully at home on Thursday, August 13, the day following her 96th birthday. Beloved wife of the late Sidney. Cherished mother of Judi and Andy
 Marcus and Ronni and the late Mel Rich. Adored grandmother of...News about the Charlotte Bobcats, including commentary and archival articles published
 in The New York Times.",
        "entities": [
            {
                "type": "Person",
                "relevance": "0.832815",
                "sentiment": {
                    "type": "negative",
                    "score": "-0.346733"
                },
                "count": "8",
                "text": "Officer Randall Kerrick."
            },
            {
                "type": "Person",
                "relevance": "0.468088",
                "sentiment": {
                    "type": "negative",
                    "score": "-0.517731"
                },
                "count": "3",
                "text": "Jonathan Ferrell"
            },
            {
                "type": "JobTitle",
                "relevance": "0.384289",
                "sentiment": {
```


27. Go to application in Bluemix and click on MongoDB lab. You can find the database and collection. Click on the collection that you have mentioned while creating the app. You can fin the output as a json file with all the entities.

HTML CODE:

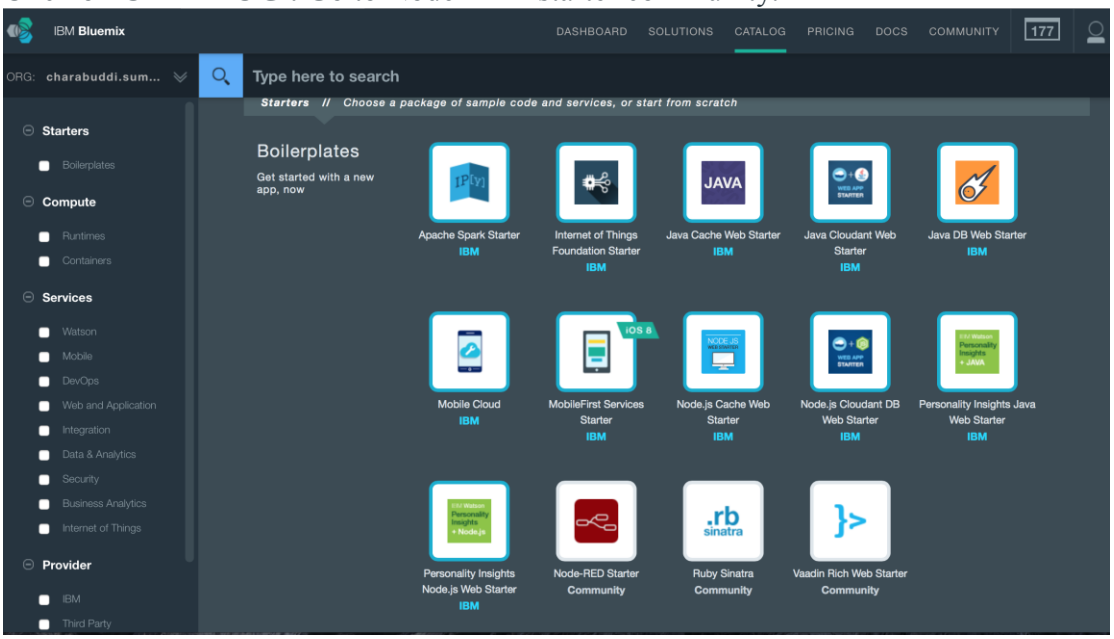
```
<head>
  <meta name="viewport" content="width=320, initial-scale=1">
  <title>Chat</title>
</head>

<body>
  <div id="wrapper">
    <div id="chat_box" class="content"></div>

    <div id="footer">
      <div class="content">
        <input type="text" id="user" placeholder="Type a text" />
        <input type="button" id="send_btn" value="Send"
onclick="sendMessage()">
      </div>
    </div>
  </div>
</body>

<script type="text/javascript">
  var wsUri = "ws://{{req.headers.host}}/ws/chat100";
  var ws = new WebSocket(wsUri);

  function sendMessage() {
    var user = document.getElementById('user');
    var payload = {
        user:user.value,
    };

    ws.send(JSON.stringify(payload));
    alert("Data stored in database");
  };

</script>
```

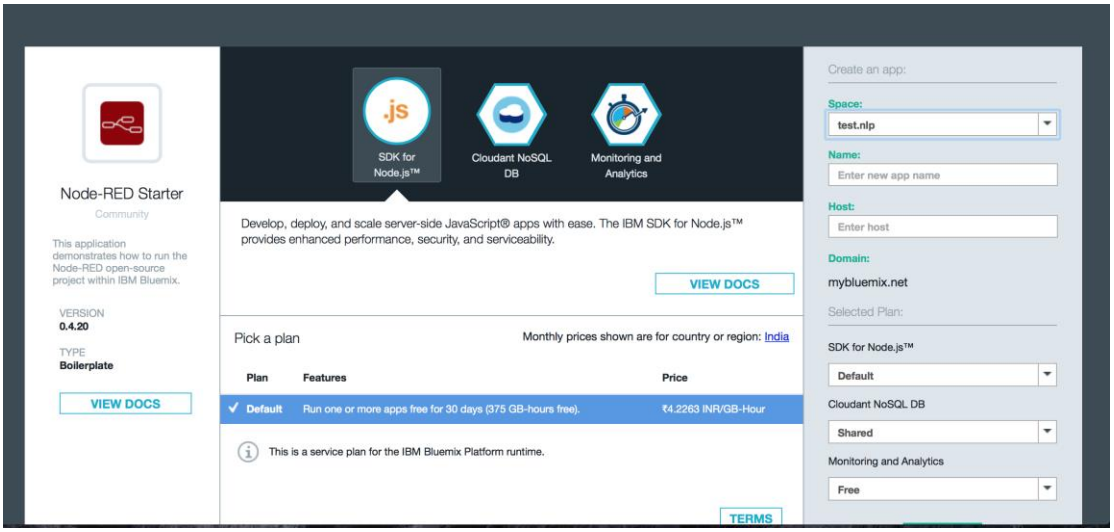
Appendix 3. Creating an Application in IBM Bluemix using Mongo DB

To get started now, try this tutorial. You'll create an app and deploy it.

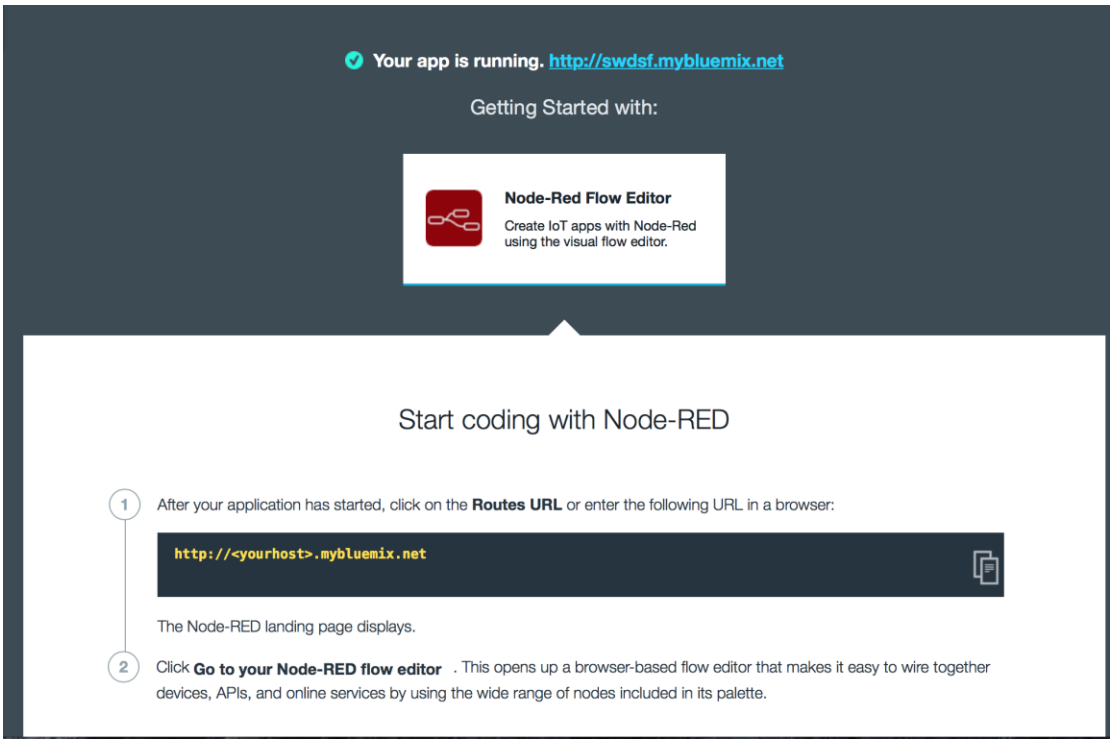
1. Sign in to Bluemix. The Dashboard opens:

2. The Dashboard shows an overview of the active Bluemix space for your organization. By default, the space is dev and the organization is the user name of the person who created the project
3. Click on **CATALOG .** Go to Node-RED starter community.

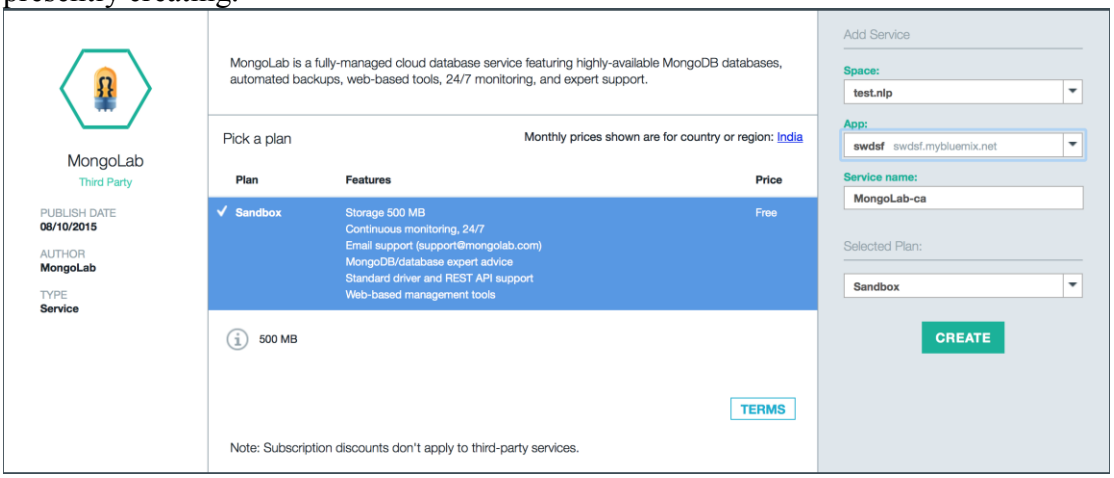




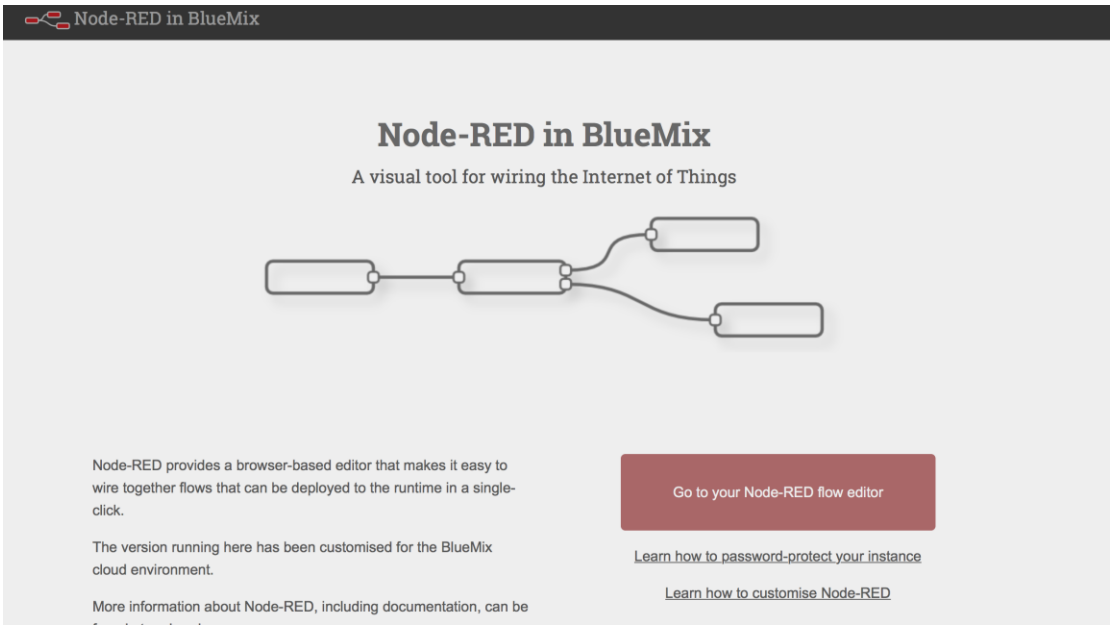
4. Name your app and then click **FINISH**. The name is a unique URL where you access your app. After a moment, the app starts. The app's Overview page opens and shows that the app is running.



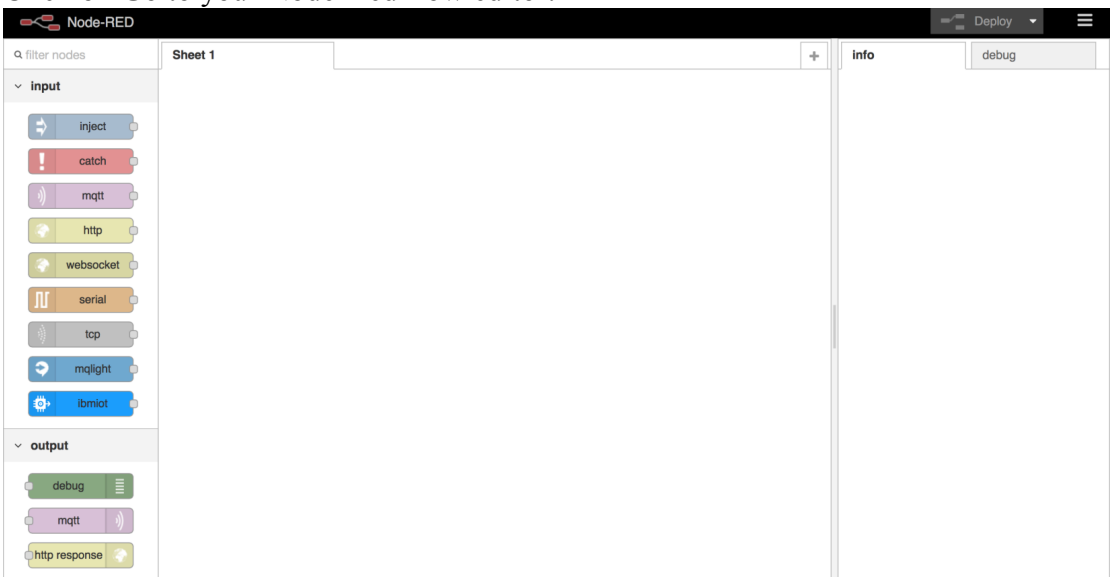
5. Go to CATALOG and then add the service “MongoLab” to the application that you are presently creating.



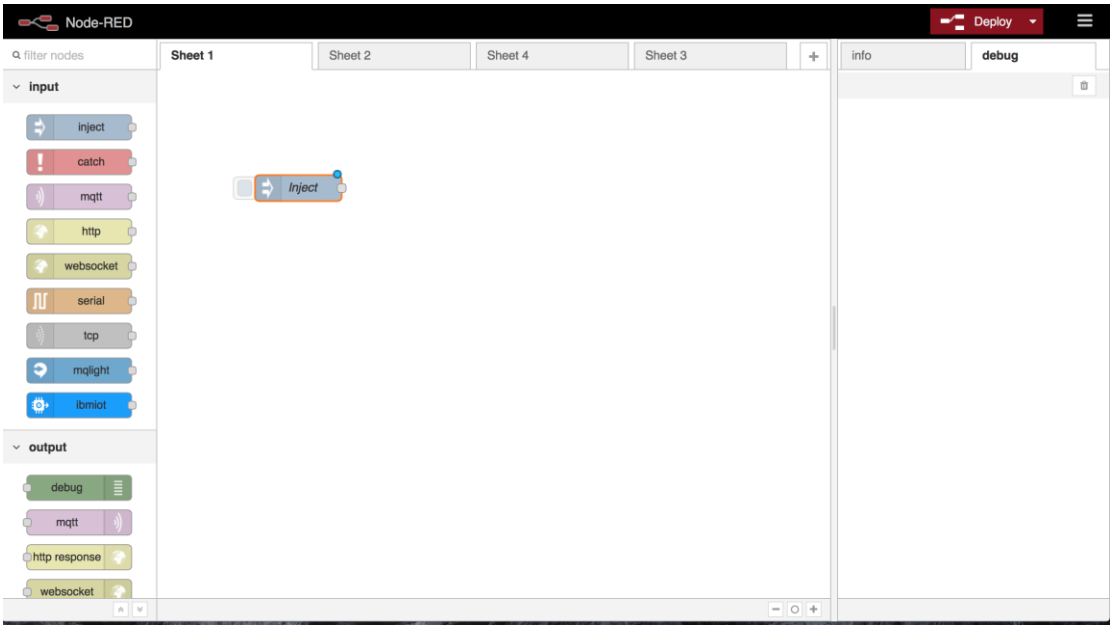
6. Click on the link where the app is running.



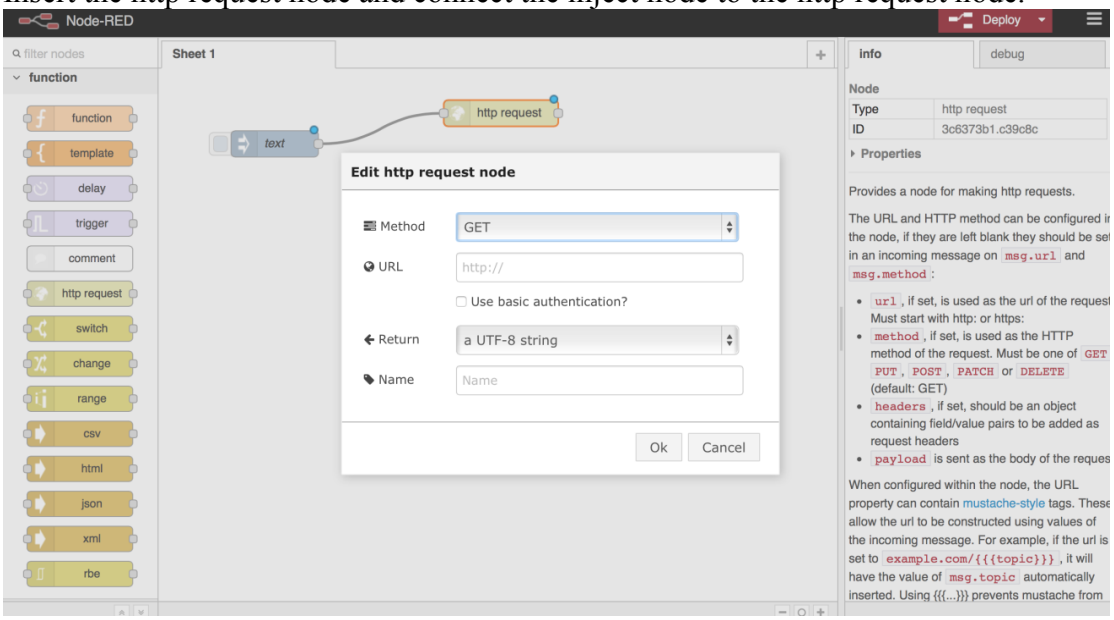
7. Click on Go to your Node-Red flow editor.



8. The left side of the screen contains all the nodes.
9. Insert the inject node into the worksheet.

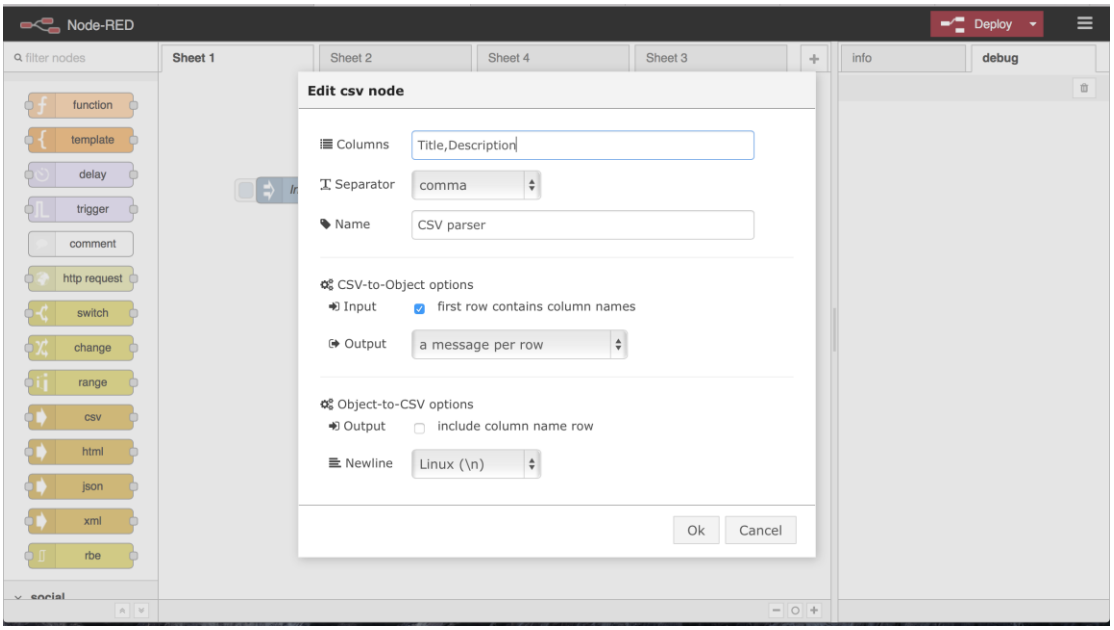


10. Insert the http request node and connect the inject node to the http request node.

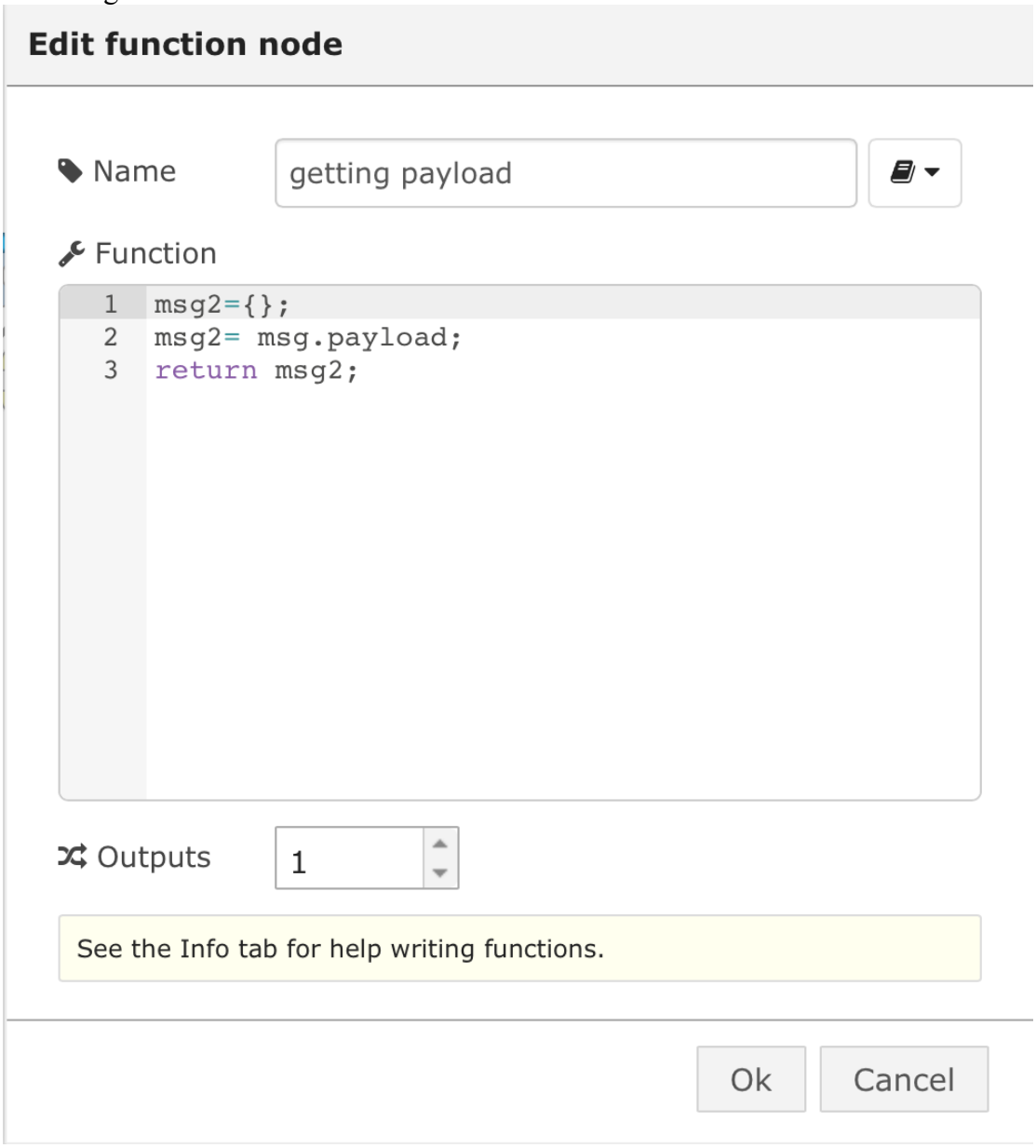


11. Configure the http request node with the following link or any download link of the CSV file.
https://doc-00-bg-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/31lgel6v59hbkfe01h72i9r1c2pal3t6/1439301600000/06884754726730946258/*/0B9vK8aMc6Xk5R1NVUGljcUhhUDQ?e=download

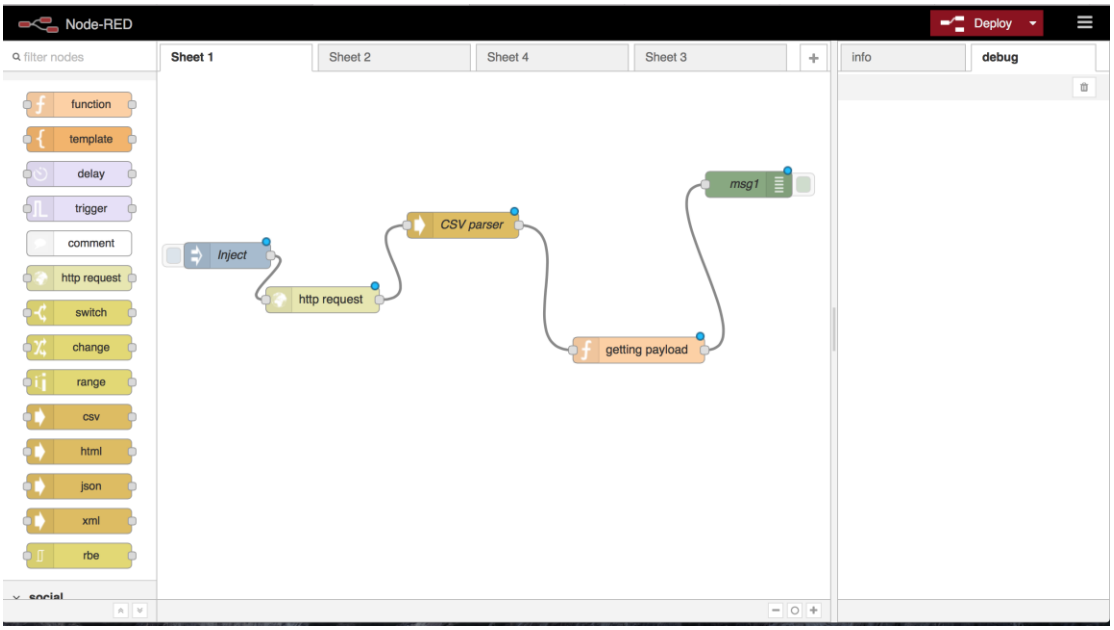
12. Create a CSV parser with a CSV node. And configure the CSV node as given below with proper column names (Note: The column names are according to the CSV file).


Node-RED
Deploy
filter nodes
Sheet 1
Sheet 2
Sheet 4
Sheet 3
info
debug
function
template
delay
trigger
comment
http request
switch
change
range
csv
html
json
xml
rbe
social
Edit csv node
Columns
Title,Description
Separator
comma
Name
CSV parser
CSV-to-Object options
Input
first row contains column names
Output
a message per row
Object-to-CSV options
Output
include column name row
Newline
Linux (\n)
Ok
Cancel

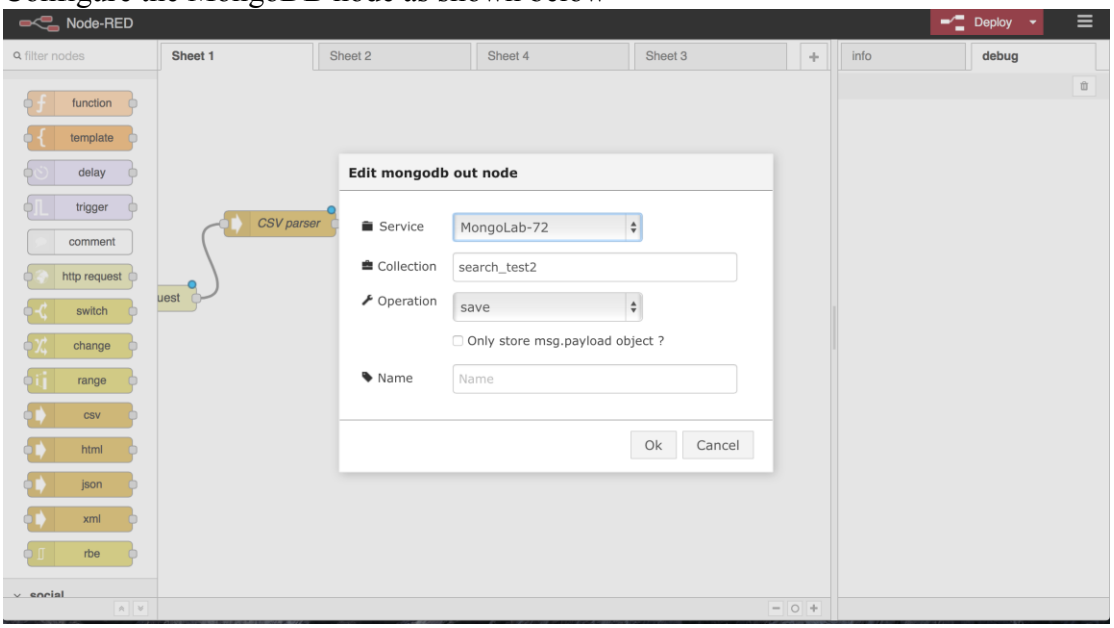
13. Insert the function node into the worksheet. Configure the function node with the following code.



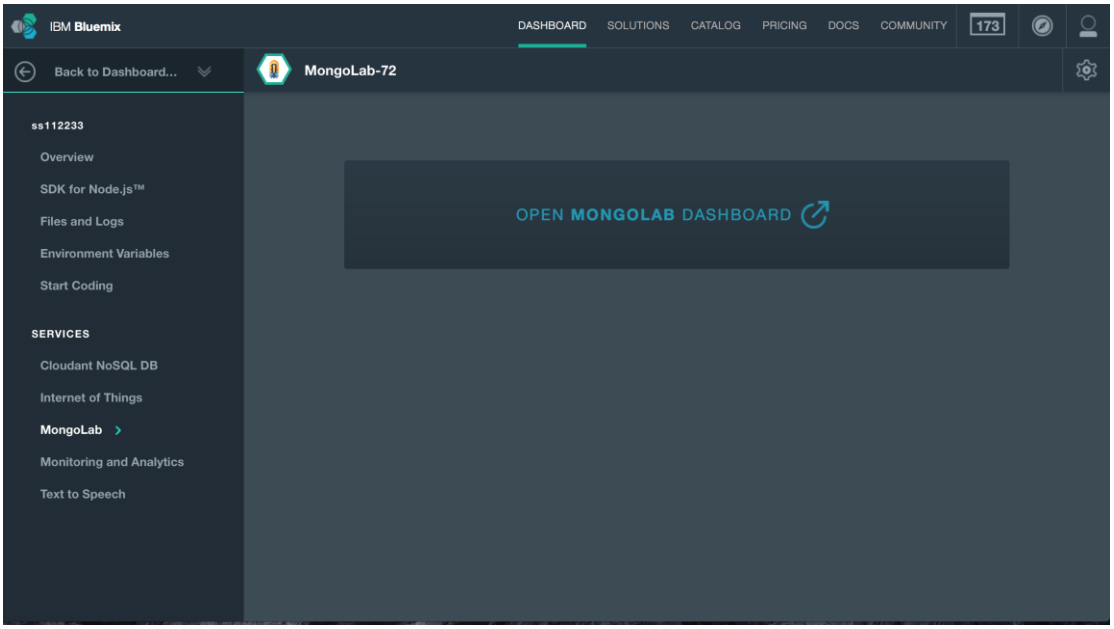
14. To know whether we are getting the output or not. Insert the debug node and connect it to the function node.

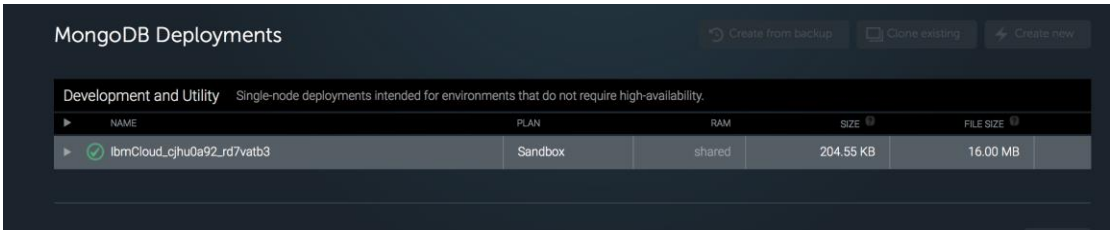


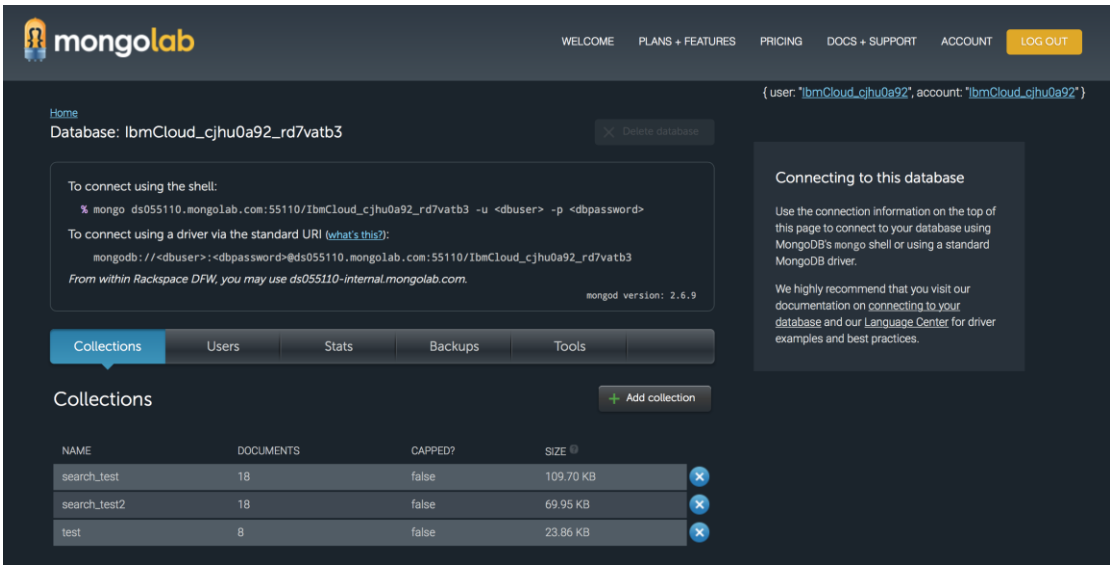
15. Connect the function node to the MongoDB node. So that we can store the data into the MongoDB.
16. Configure the MongoDB node as shown below

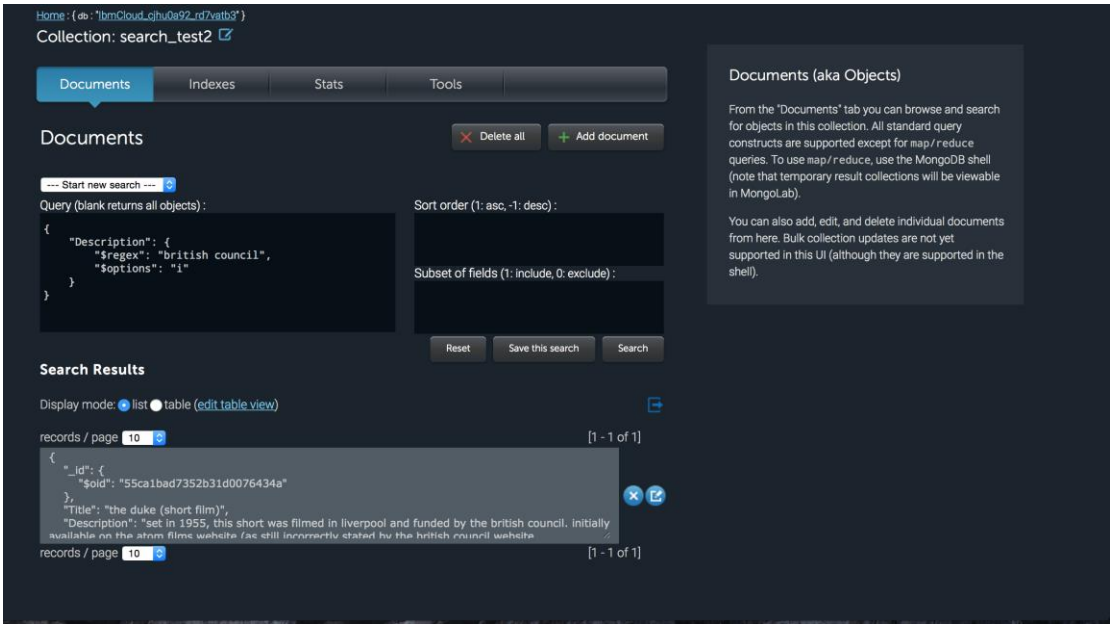


17. Click on the Deploy, to save the changes.
18. Click on the button that is present at the start of inject node to inject the string.
19. The workflow runs and you can see the output in the debug tab that is present on the right side.
20. Go to the IBM Bluemix Dashboard. Click on the application and then go to the MongoDB(MongoLab) service.


IBM Bluemix
DASHBOARD
SOLUTIONS
CATALOG
PRICING
DOCS
COMMUNITY
173
Back to Dashboard...
MongoLab-72
ss112233
Overview
SDK for Node.js™
Files and Logs
Environment Variables
Start Coding
SERVICES
Cloudant NoSQL DB
Internet of Things
MongoLab
Monitoring and Analytics
Text to Speech
OPEN MONGOLAB DASHBOARD


MongoDB Deployments
Create from backup
Clone existing
Create new
Development and Utility
Single-node deployments intended for environments that do not require high-availability.
NAME
PLAN
RAM
SIZE
FILE SIZE
IbmCloud_cjhu0a92_rd7vatb3
Sandbox
shared
204.55 KB
16.00 MB


mongolab
WELCOME
PLANS + FEATURES
PRICING
DOCS + SUPPORT
ACCOUNT
LOG OUT
{ user: "IbmCloud_cjhu0a92", account: "IbmCloud_cjhu0a92" }
Home
Database: IbmCloud_cjhu0a92_rd7vatb3
Delete database
To connect using the shell:
% mongo ds055110.mongolab.com:55110/IbmCloud_cjhu0a92_rd7vatb3 -u <dbuser> -p <dbpassword>
To connect using a driver via the standard URI (what's this?):
mongodb://<dbuser>:<dbpassword>@ds055110.mongolab.com:55110/IbmCloud_cjhu0a92_rd7vatb3
From within Rackspace DFW, you may use ds055110-internal.mongolab.com.
mongod version: 2.6.9
Connecting to this database
Use the connection information on the top of this page to connect to your database using MongoDB's mongo shell or using a standard MongoDB driver.
We highly recommend that you visit our documentation on connecting to your database and our Language Center for driver examples and best practices.
Collections
Users
Stats
Backups
Tools
Collections
Add collection
NAME
DOCUMENTS
CAPPED?
SIZE
search_test
18
false
109.70 KB
search_test2
18
false
69.95 KB
test
8
false
23.86 KB


Home : { db : "ibmCloud_cjhu0a92_rd7vatb3" }
Collection: search_test2
Documents
Indexes
Stats
Tools
Documents
Delete all
Add document
--- Start new search ---
Query (blank returns all objects) :
{
"Description": {
"$regex": "british council",
"$options": "i"
}
}
Sort order (1: asc, -1: desc) :
Subset of fields (1: include, 0: exclude) :
Reset
Save this search
Search
Search Results
Display mode: list table (edit table view)
records / page 10
[1 - 1 of 1]
{
"_id": {
"$oid": "55ca1bad7352b31d0076434a"
},
"Title": "the duke (short film)",
"Description": "set in 1955, this short was filmed in liverpool and funded by the british council. initially
records / page 10
[1 - 1 of 1]
Documents (aka Objects)
From the "Documents" tab you can browse and search for objects in this collection. All standard query constructs are supported except for map/reduce queries. To use map/reduce, use the MongoDB shell (note that temporary result collections will be viewable in MongoLab).
You can also add, edit, and delete individual documents from here. Bulk collection updates are not yet supported in this UI (although they are supported in the shell).