MIRACLE
SOFTWARE SYSTEMS

Going Hands-On with IBM Bluemix Integration Services

Cloud Developer Lab | Miracle Innovation Labs

Manasa Sutapalli
Lead Researcher API M, MIL
Miracle Software Systems, Inc.

February 02nd, 2016

Going Hands-On with IBM Bluemix Integration Services

Goal

In this lab, the user will be creating an Enterprise application using IBM Bluemix where you will be creating a REST API Proxies for existing SOAP Services using IBM API Management; this REST API can be consumed into Bluemix Application.

Pre-Requisites

The following are required for this lab to run successfully,

- Browser for accessing IBM Bluemix
- Active Email ID for registering to IBM Bluemix
- Cloud Foundry
- Public WSDL for testing
- SOAP/REST Formats
- SOAPUI 5.0

Technologies Involved

- IBM Bluemix
- Node.js
- IBM API Management (Cloud Version)
- Cloud Foundry

Lab Steps

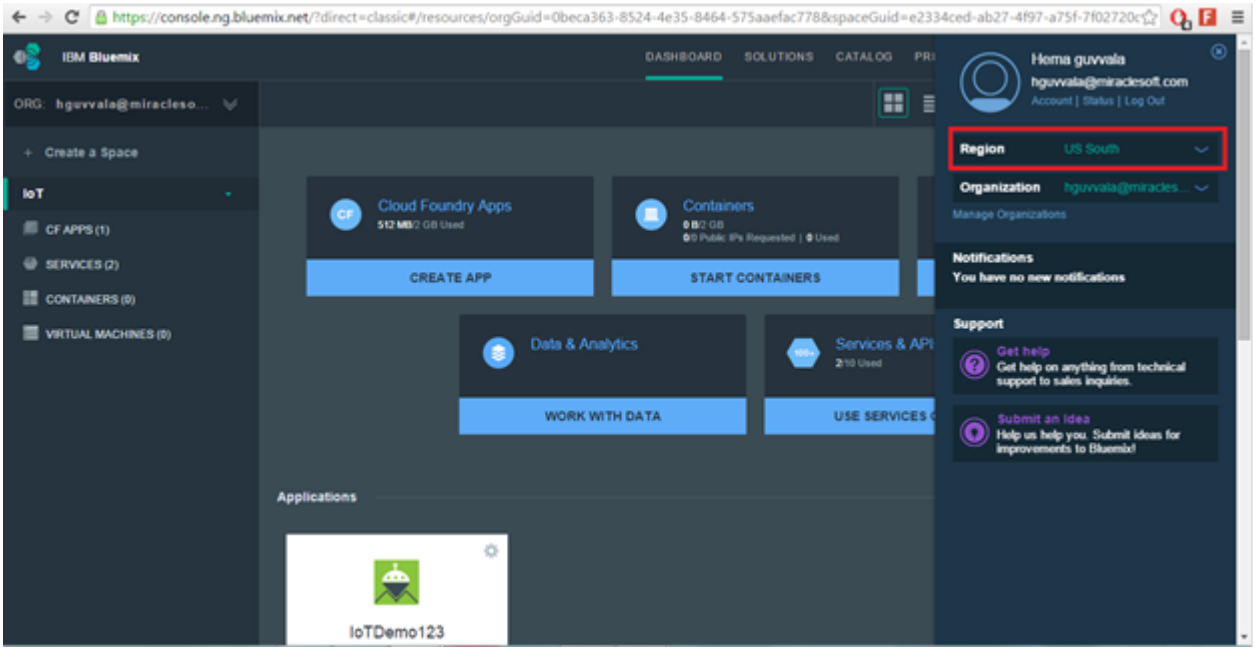
So, let us get started with the lab!

#1 | Access Bluemix

The first step will be to make sure that you have access to the IBM Bluemix Console with either the free trail option (or) the paid subscription option.

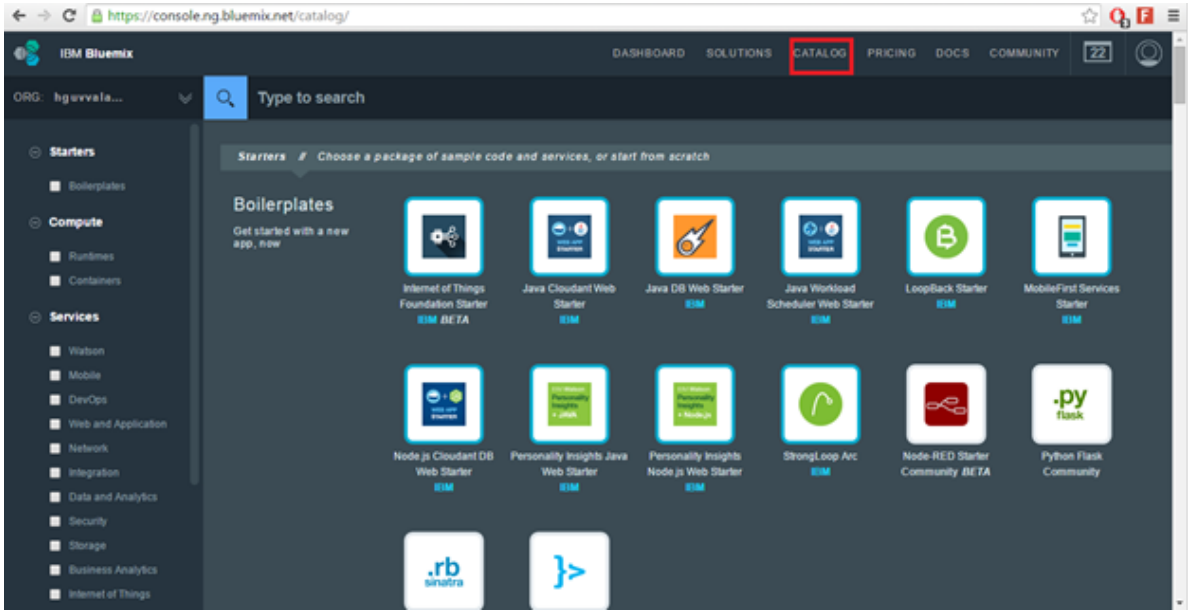
Login to Bluemix at http://bluemix.net
(or)
Register today at https://console.ng.bluemix.net/registration/

After you login, you can see the dashboard where you can take a look at your applications and services. You can also go to the profile icon at the top and change which Cloud Availability Region you are working in.

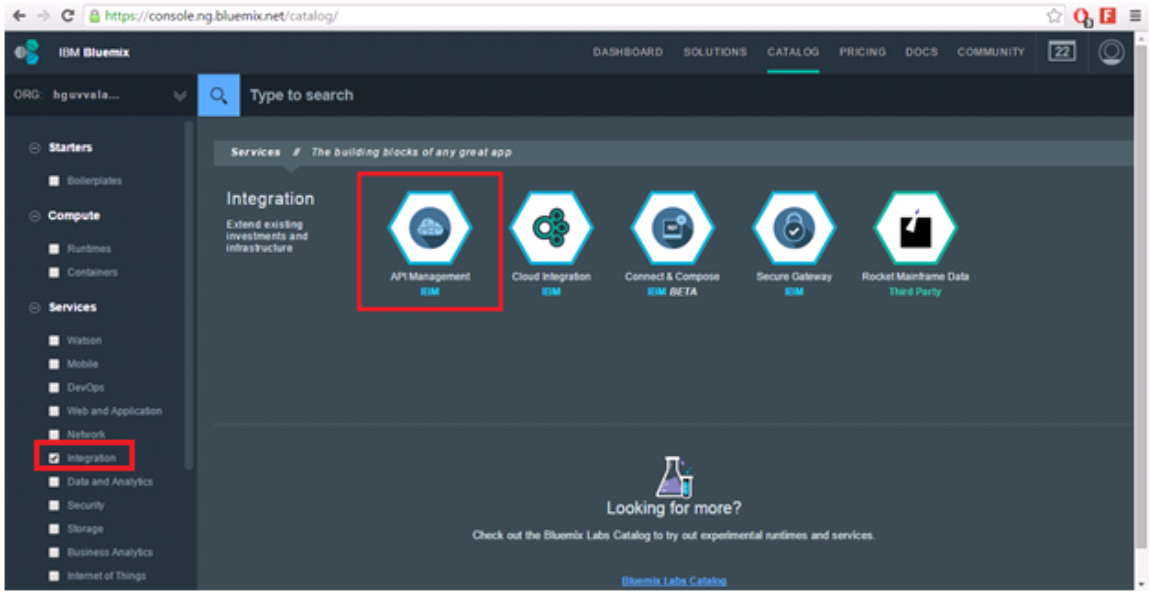


#2 | Bind API Management Service

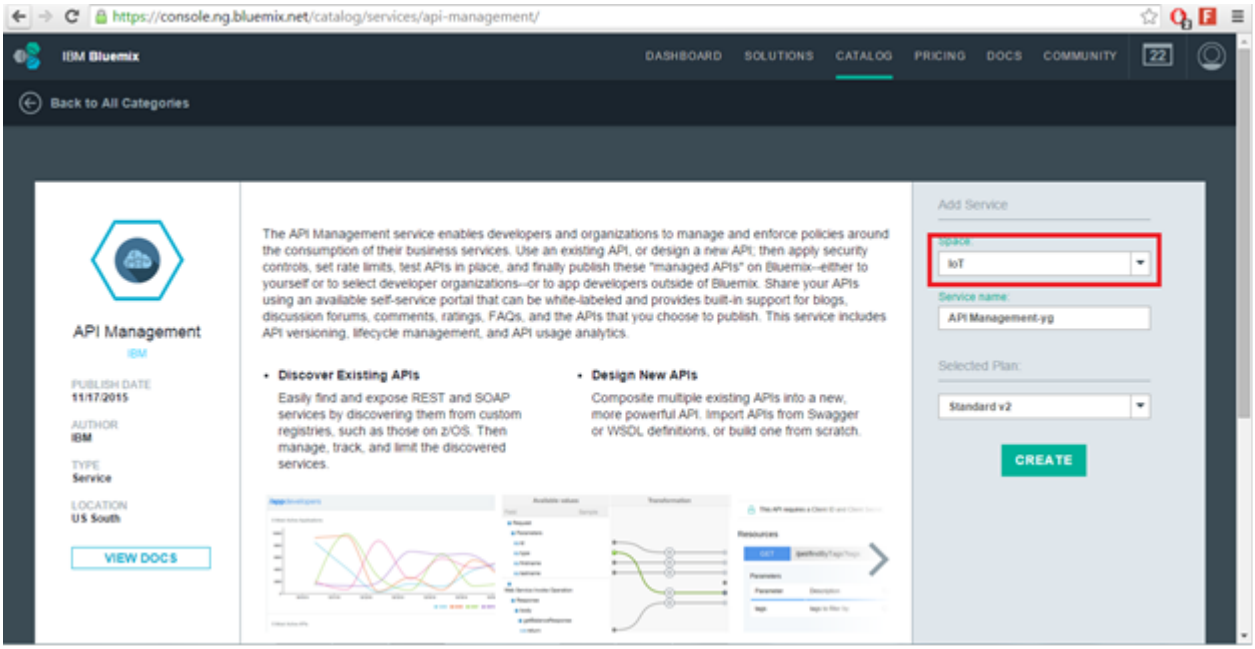
Go to Catalog, it will show all the services that are provided by IBM Bluemix.



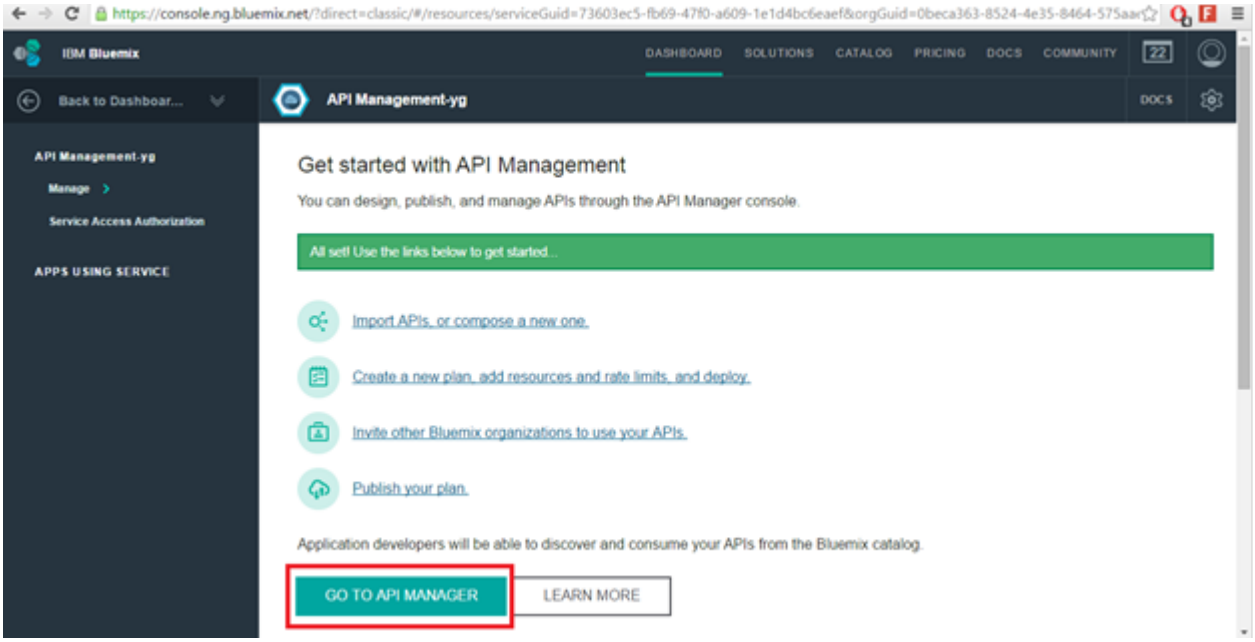
In Integration Services, select **API Management Service** for creating API's and publishing them on to Bluemix.



Click on API Management Service. Choose your space in “**Space**” field and click on “**Create**”.

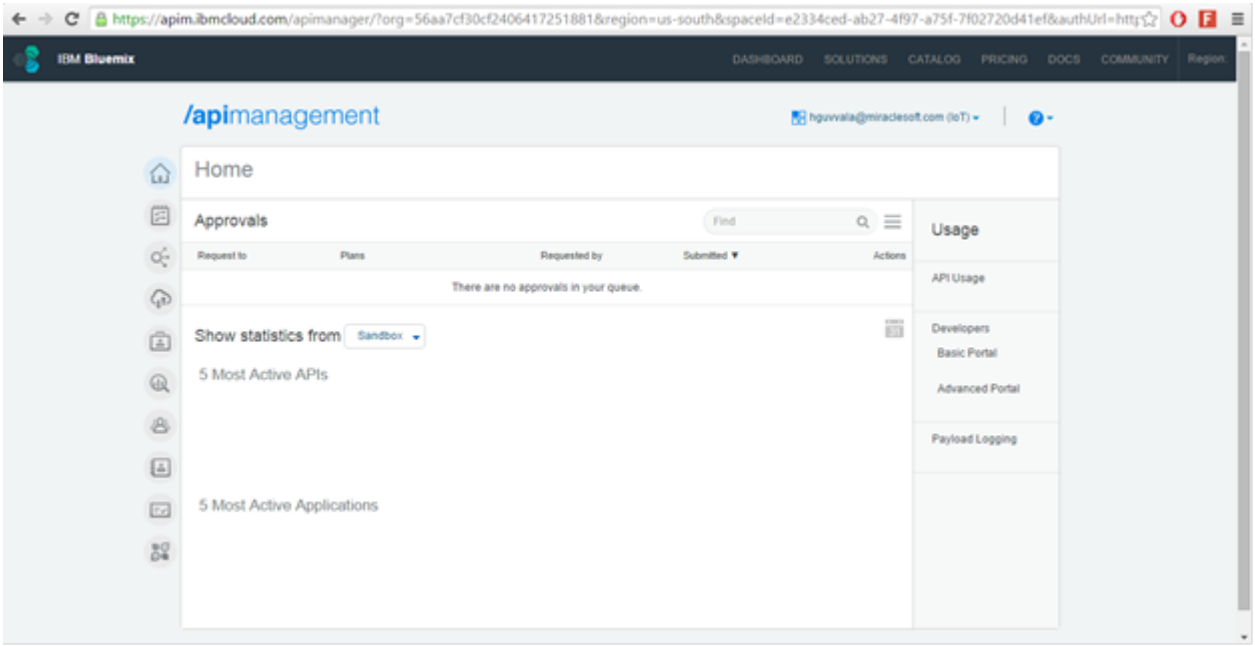


Click on **“Go To API Manager”**.

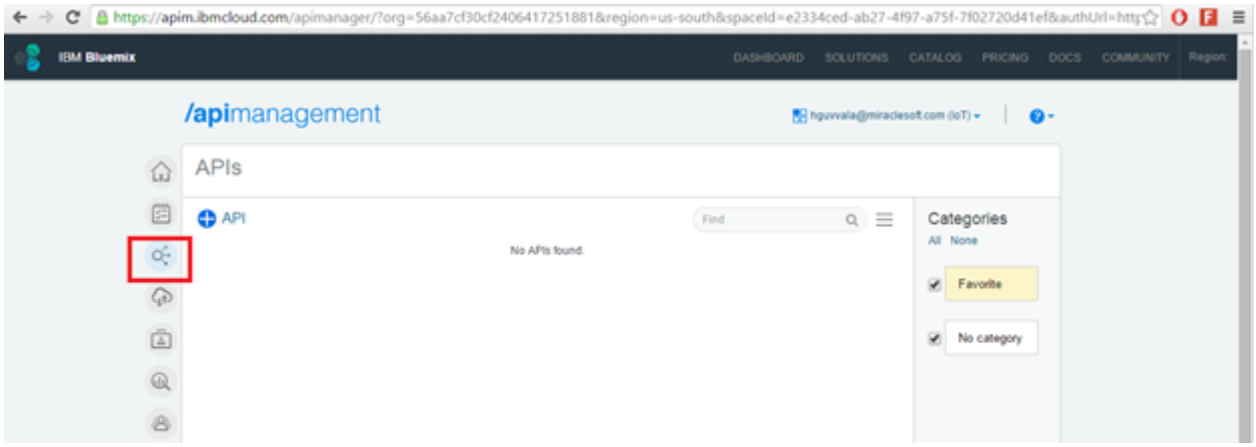


#3 | Create API

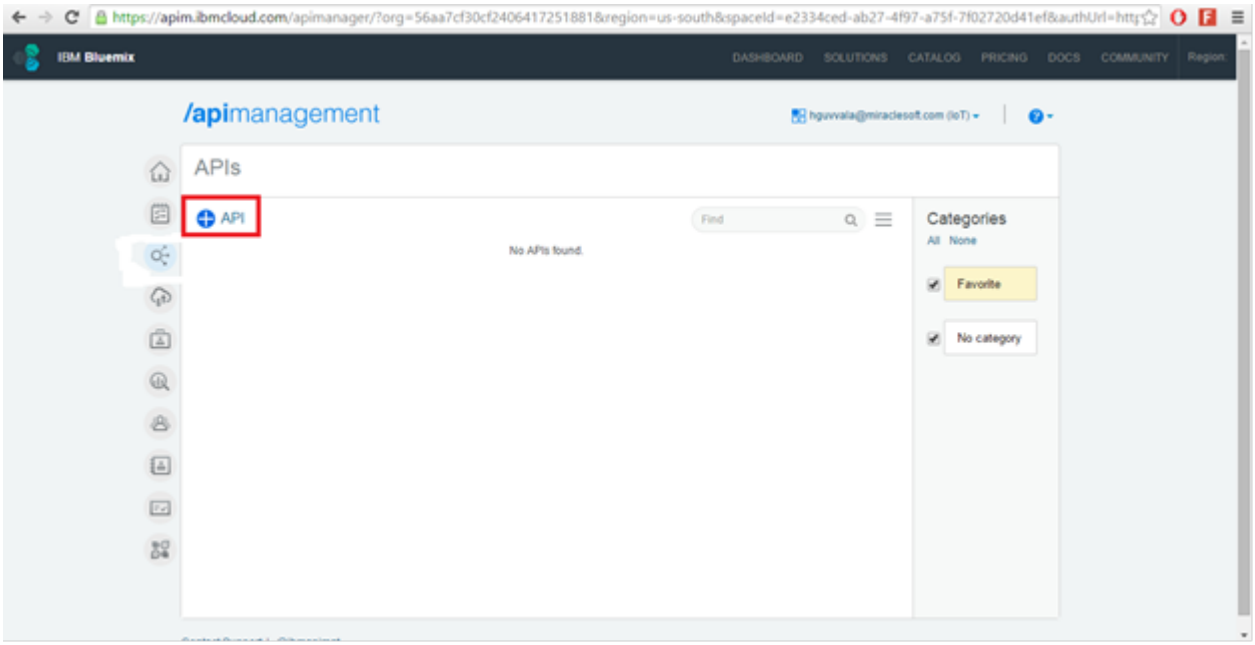
It will be redirecting into **API Manager** Console.



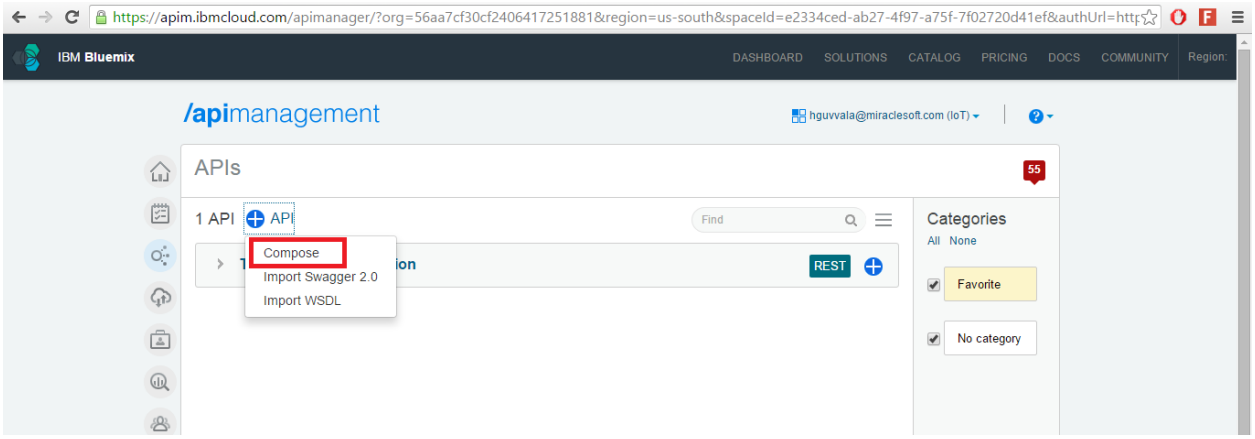
Create API by clicking on API's icon in Navigation Pane, this will take you to Draft API's view.



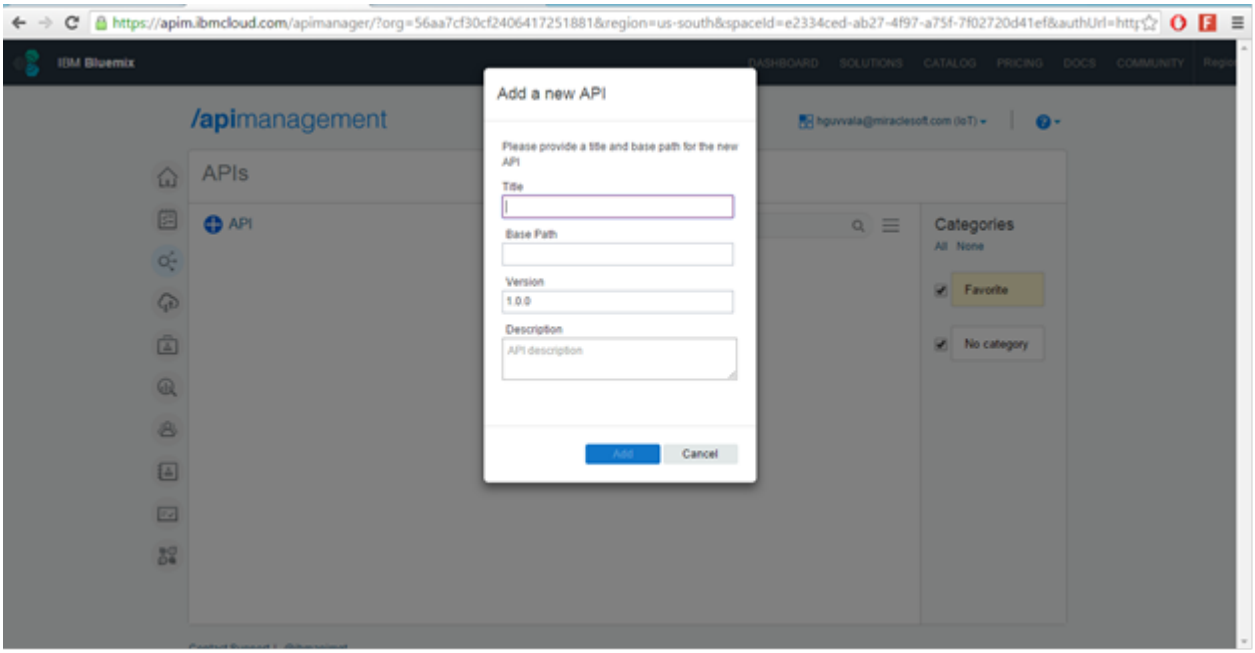
Click on **"+API"** Button to create New REST API.



Select **Compose** from the list to create a REST API.



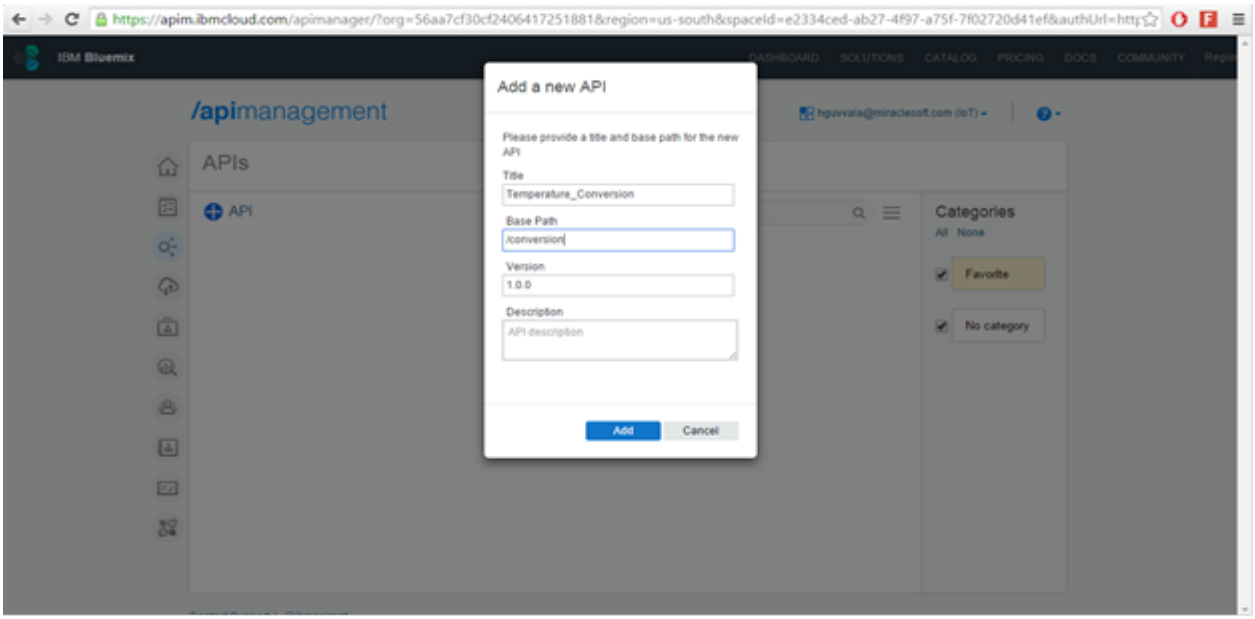
Fill the fields as shown below. Click the **Add button**. This will add the API to the Draft APIs list.



Fill the fields and click add button. This will add the API to the drafts list.

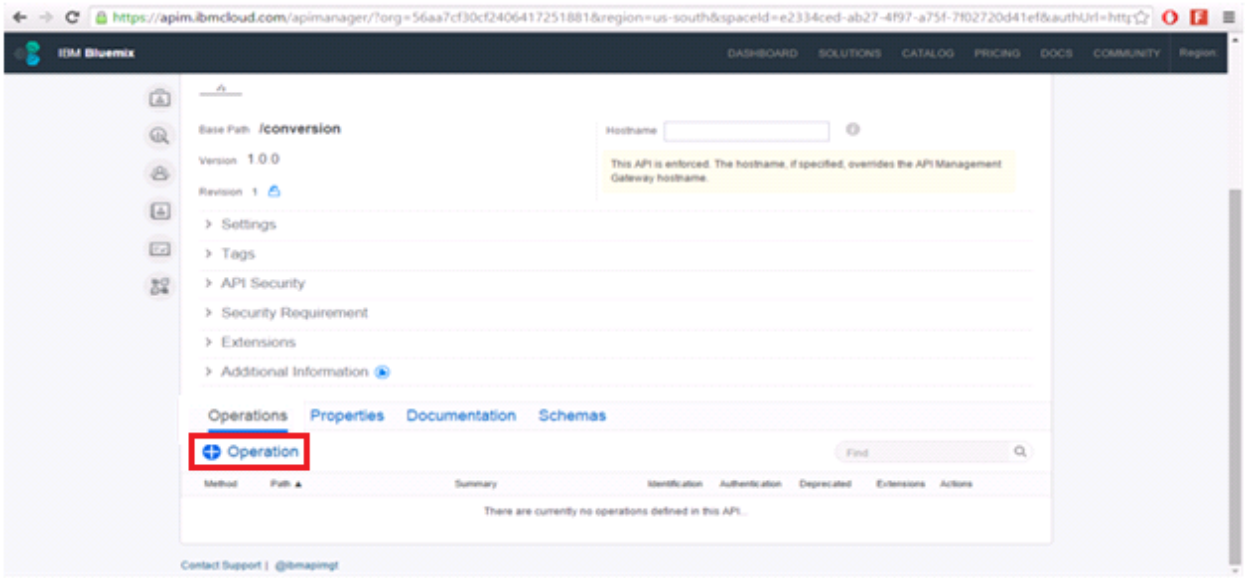
Field Name	Value
API Name(Title)	Temperature_Conversion
Base Path	/conversion
API Description	Converting the temperature from Celsius to Fahrenheit

Note : The Path specified here will become a part of the URL which will eventually be called.

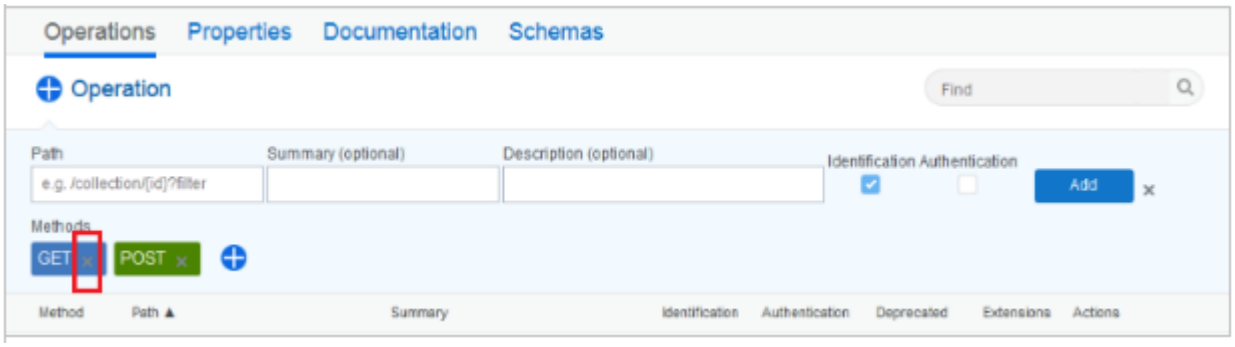


After creating the **Temperature_Conversion** API, it will immediately appear in the list of draft APIs.

Click on the **"Operations"** pane for defining the API.

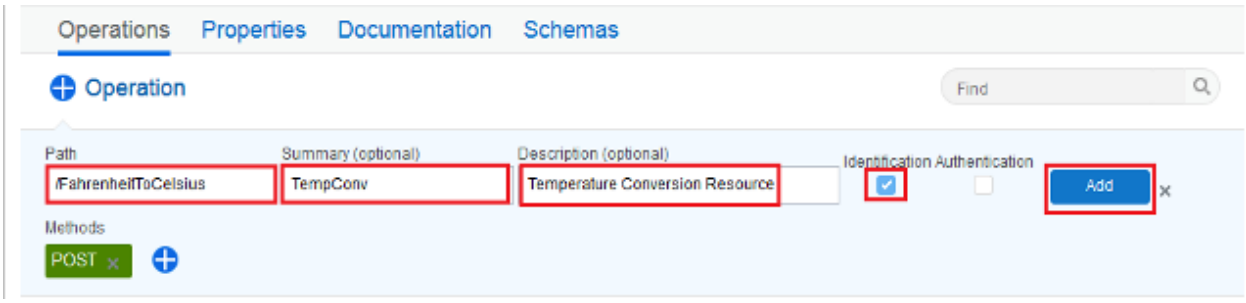


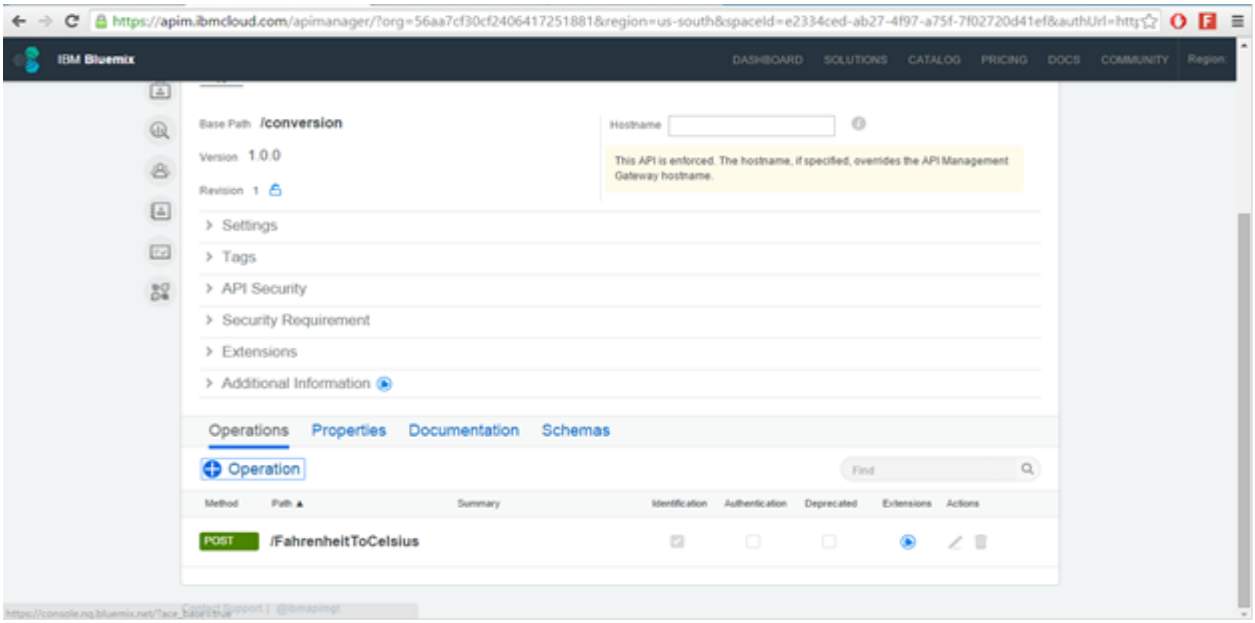
By default, the HTTP methods **GET** and **POST** are displayed. Remove the GET method by clicking the **'X'** inside of the GET button.



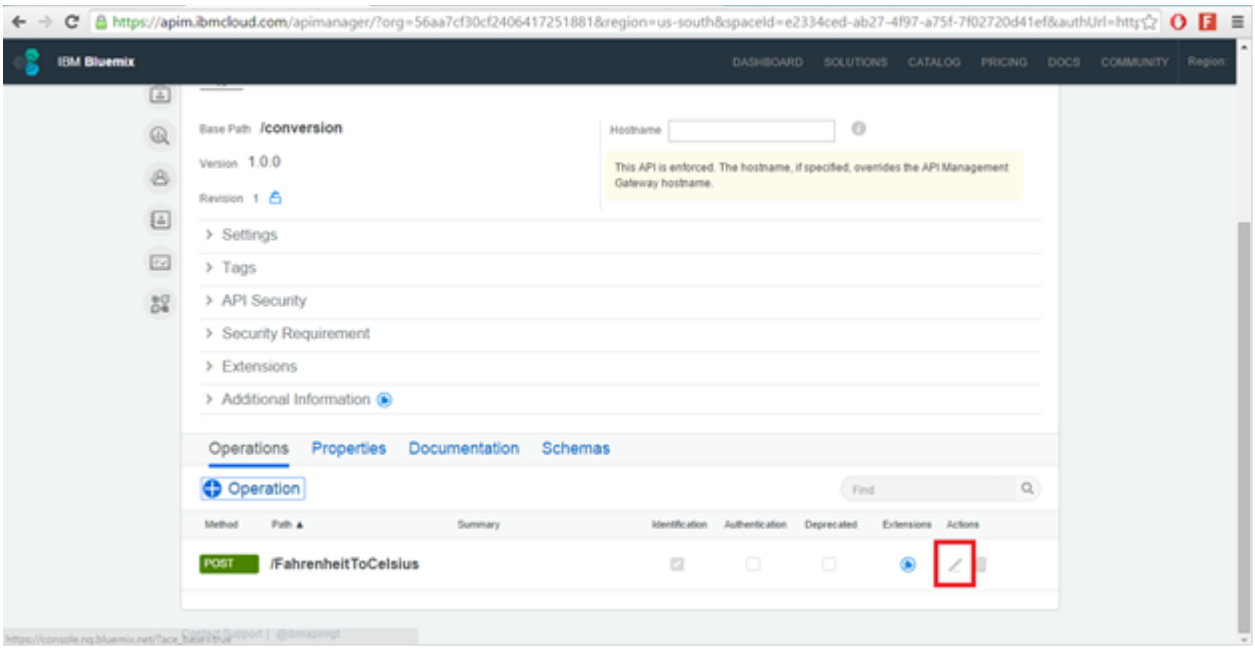
Populate the fields of the Operation for that API.

Field Name	Value
Path	/FahrenheitToCelsius
Summary	TempConv
Description	Temperature Conversion Resource



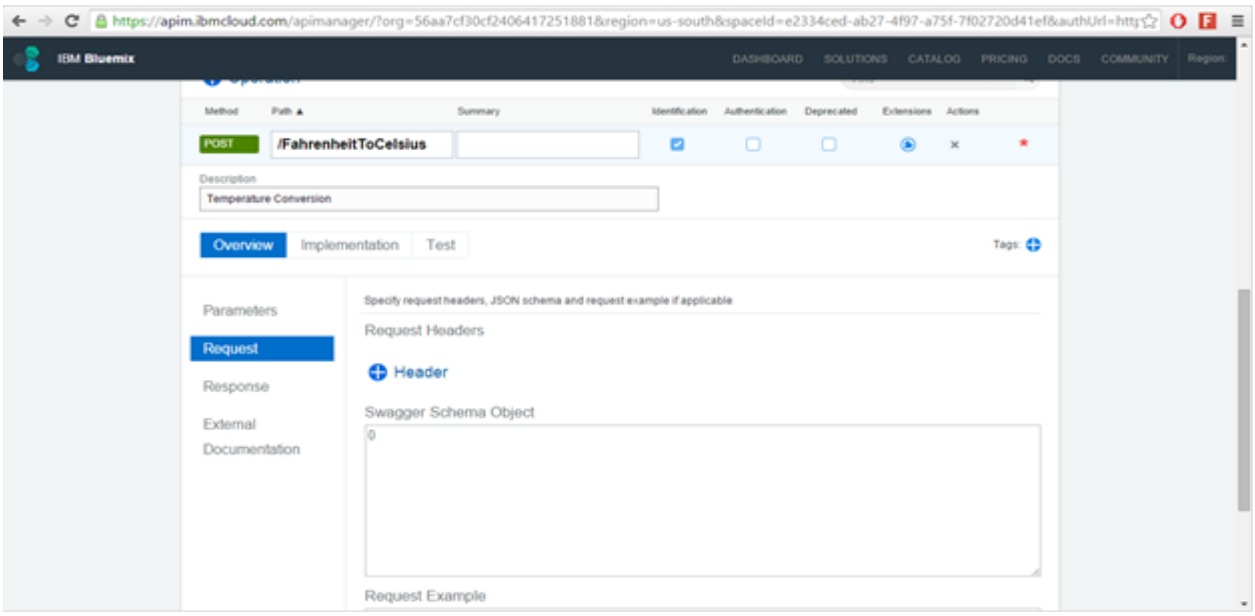


Now that the FahrenheitToCelsius resource has been defined, click on the **Edit** icon to edit the resource details.

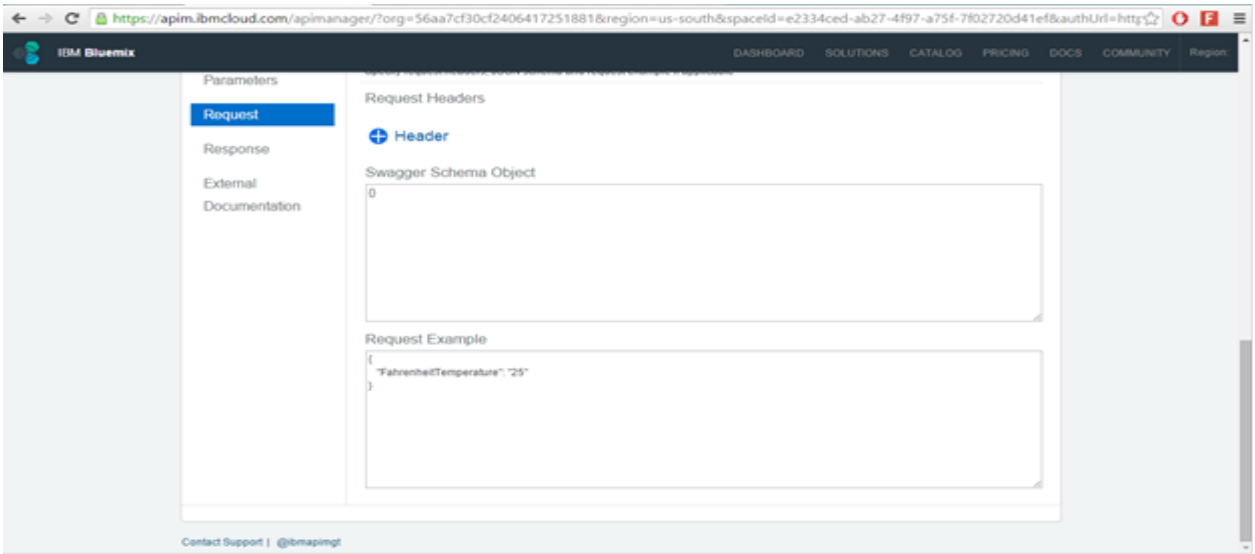


In this resource there is no need of adding any request or response headers . So the next step is to add a Request Body to define the structure of the request.

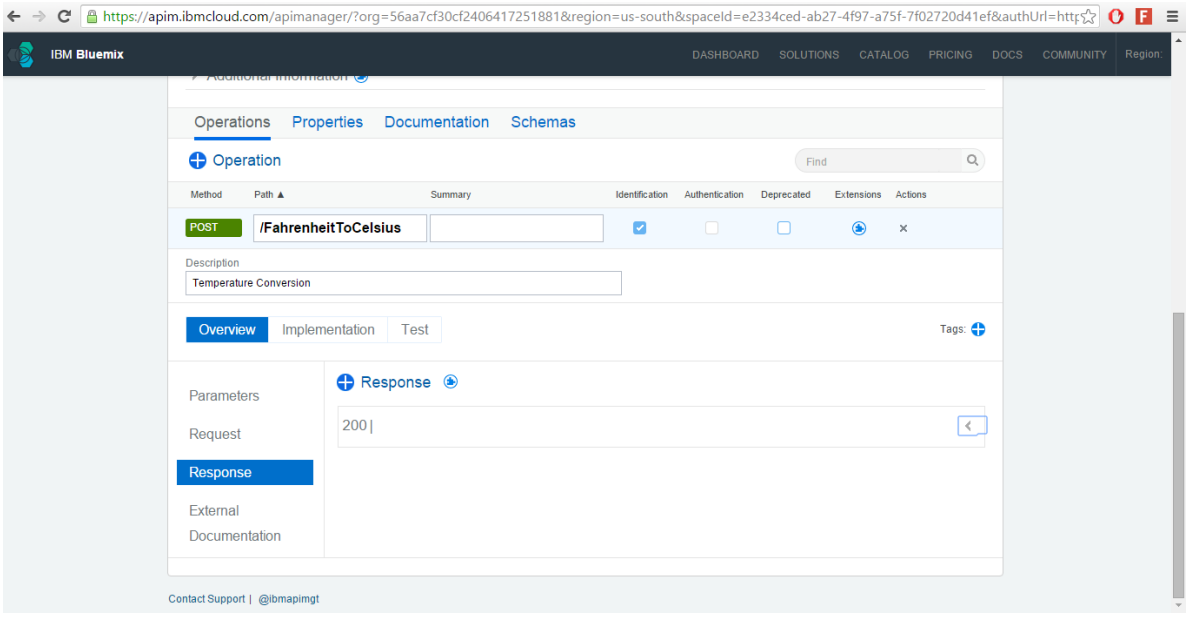
The Temperature Converter request type is SOAP, but we want to use JSON as our data format. So here you need to define your own JSON Request type.



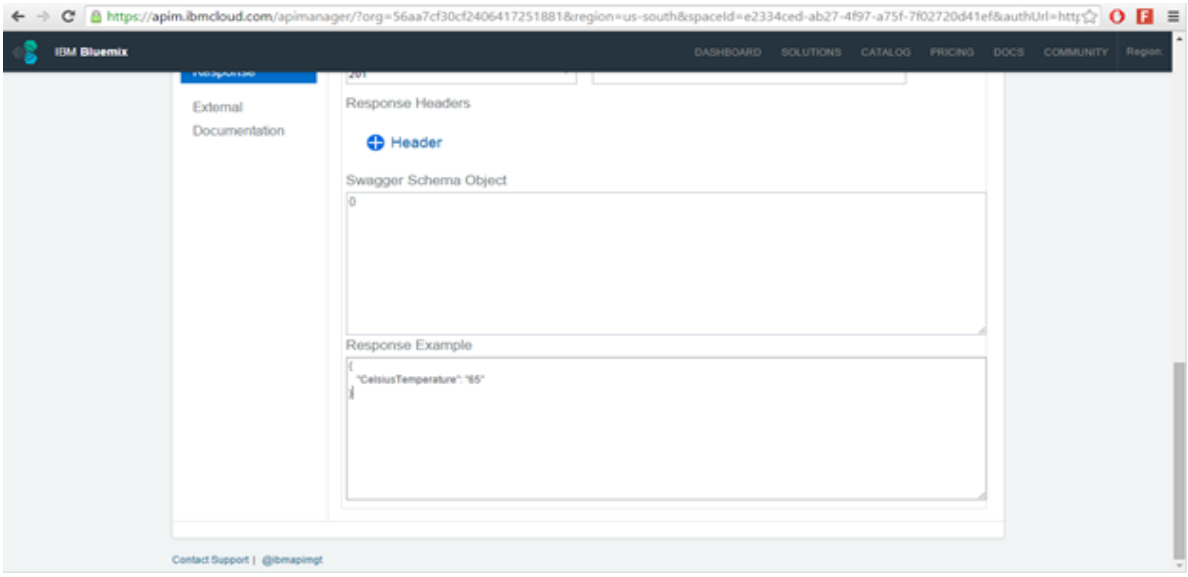
Add the Request header as **{ “Fahrenheit” : “25” }**



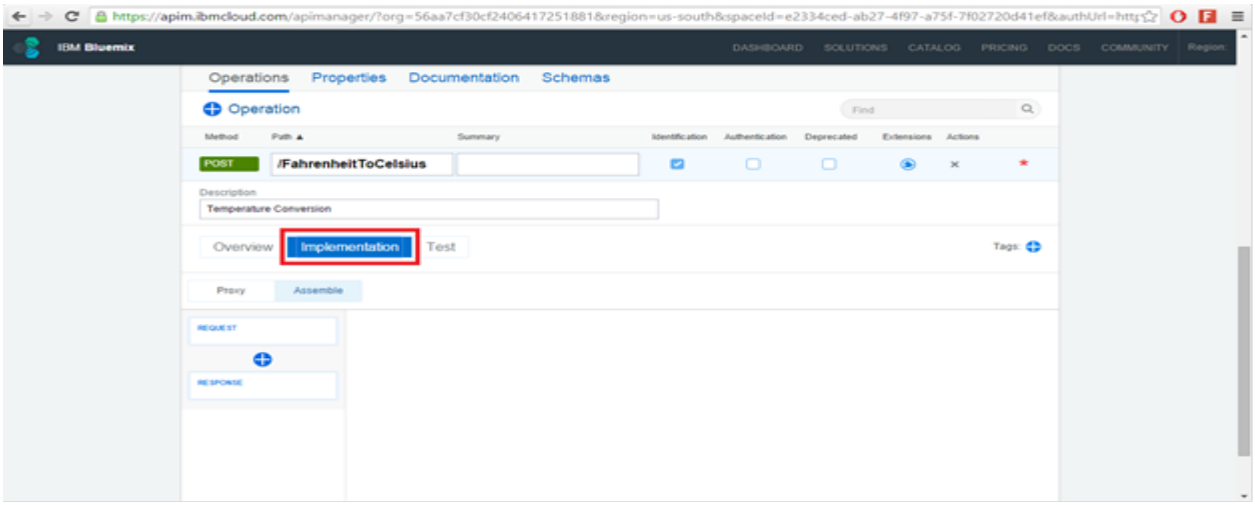
Also Add Response Headers, the request type is SOAP, but we would like to use JSON as the data format. Here 201 will be the default Response code that we need to change to **200** to add the success response.



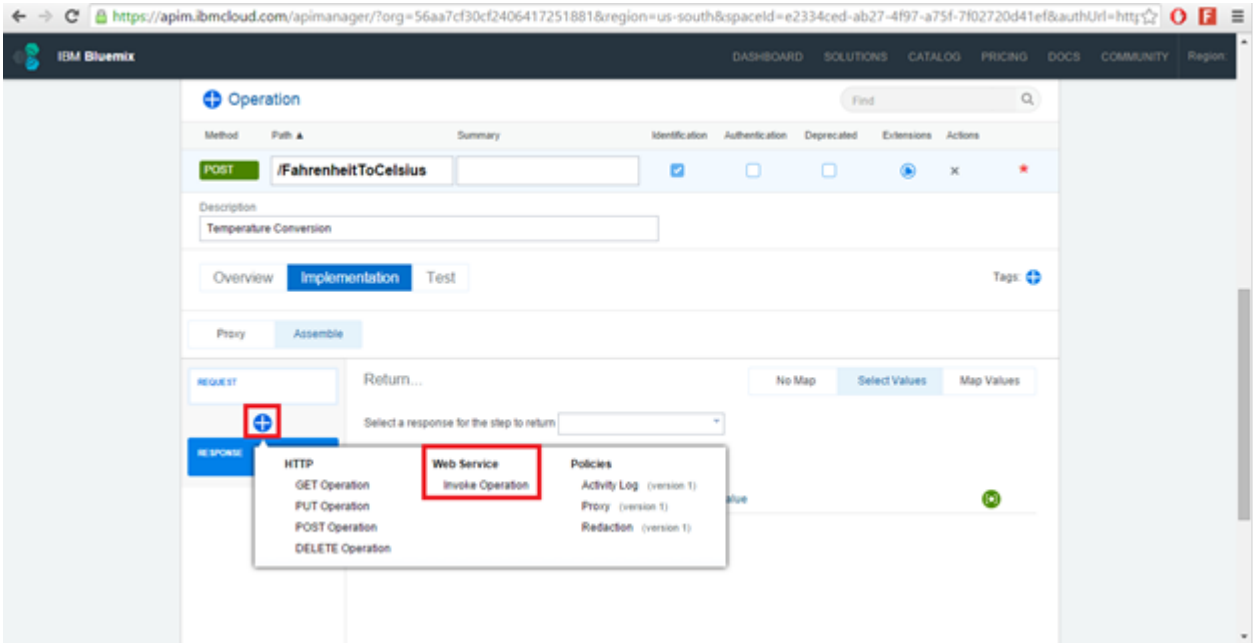
Here you need to define your own JSON Response type. Open the Response **"200 Status"** and add the Response Header format as **{ "Celsius" : "65" }**



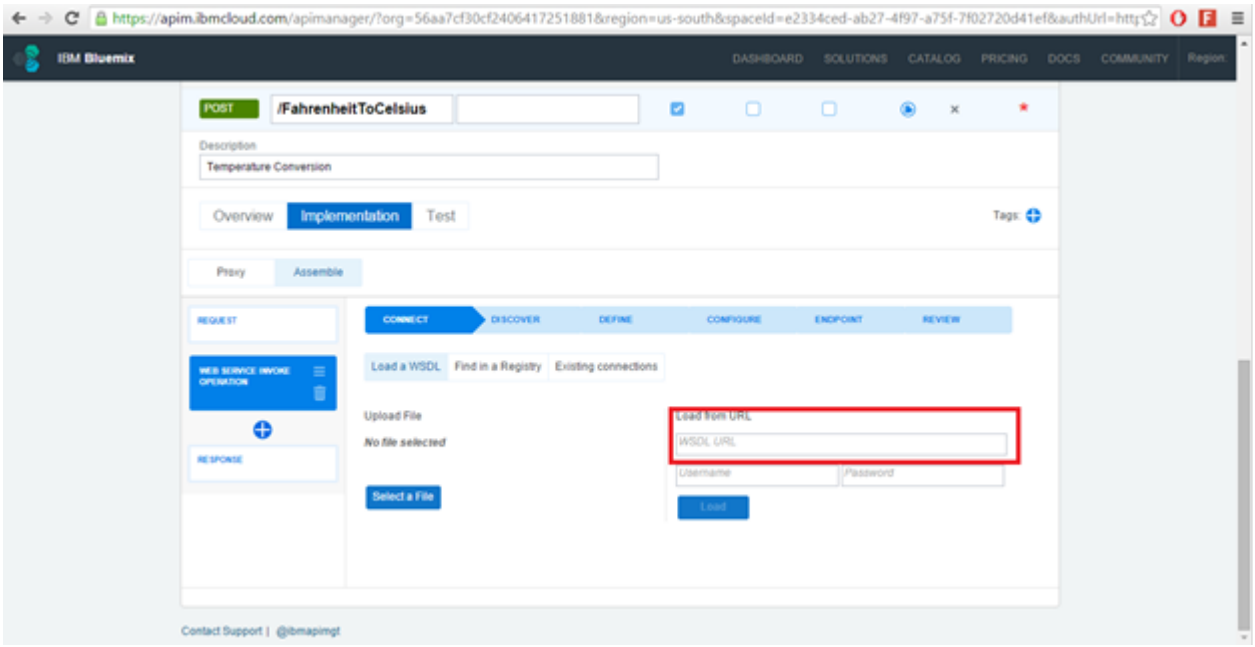
Switch back to API Manager Console, and click on **Implementation** tab for invoking the web Service.



Click the **"+"** icon to add a task to the resource implementation, and select **Invoke Operation** from the **web Service** section. This will be used to invoke the Temperature Conversion web Service.

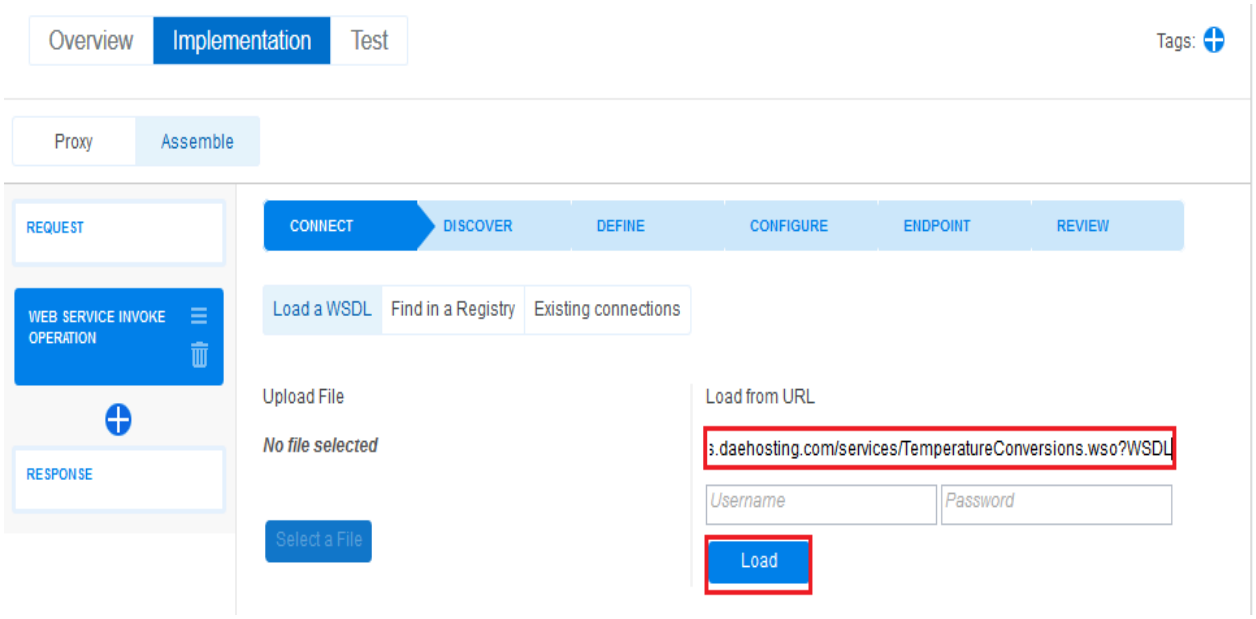


You are now on the **CONNECT** tab for the assembly task. Populate the fields as shown in the table below. The WSDL URL can be copied and pasted from the other browser tab where you viewed the WSDL. When it is completed, click the **Load** button.

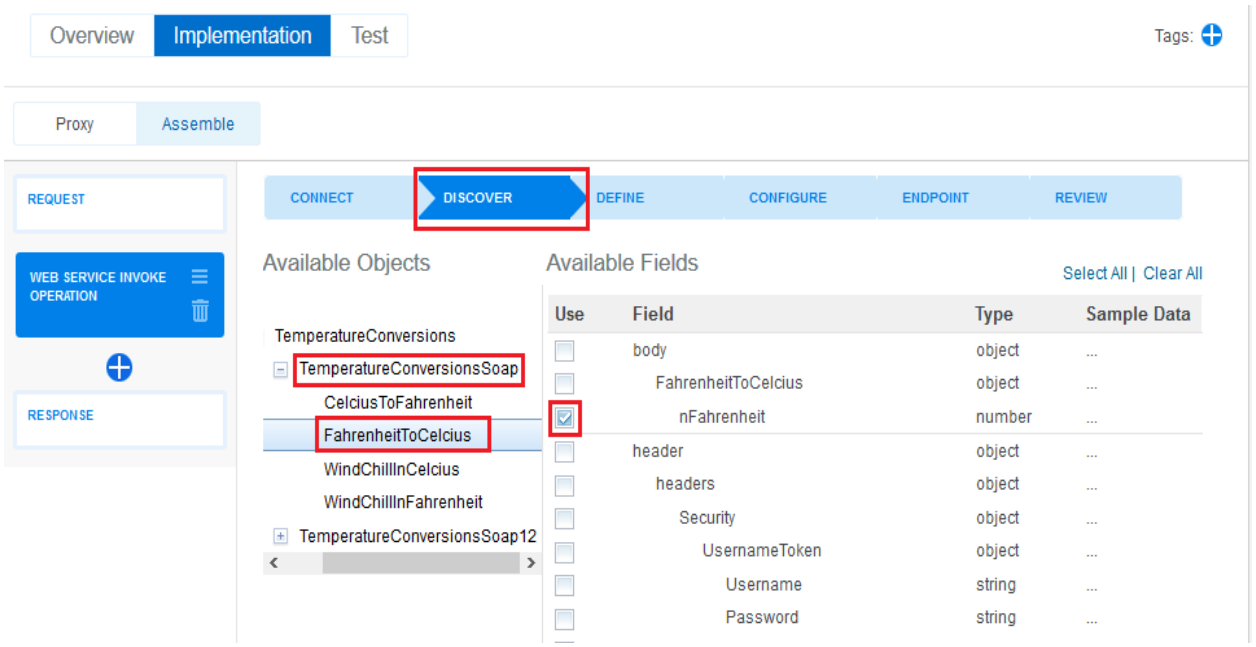


Load the WSDL URL for Fahrenheit to Celsius Temperature Conversion.

Field Name	Value
WSDL URL	http://webservices.daehosting.com/services/TemperatureConversions.wso?WSDL
Username	Leave blank
Password	Leave blank

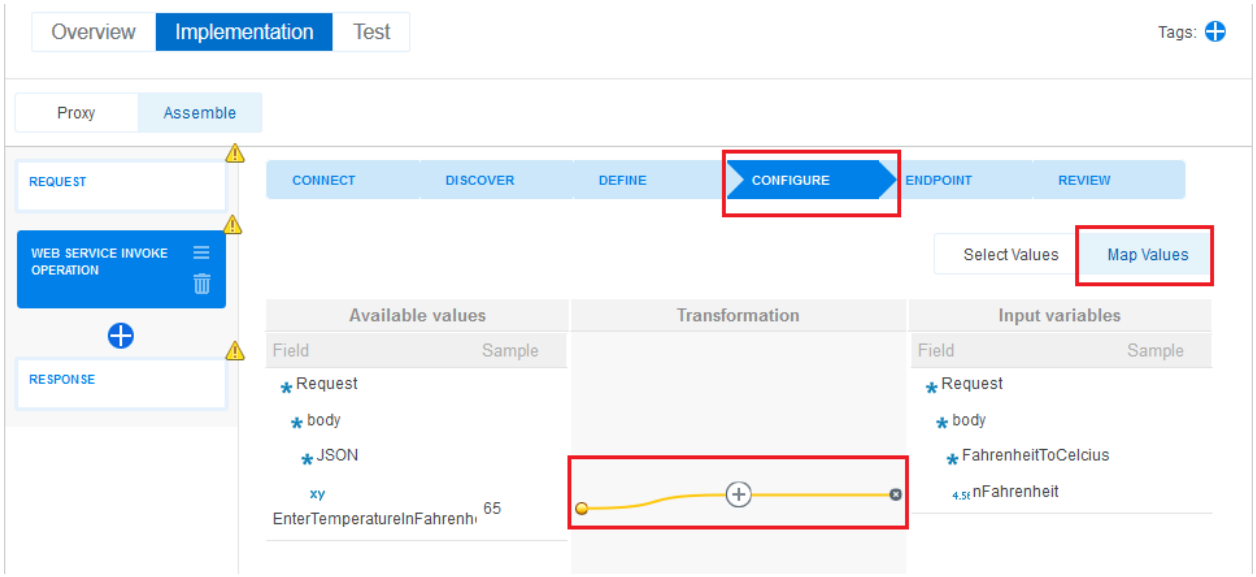


In the **DISCOVER** tab of the implementation module, the WSDL which has been provided has been parsed and presented in a form which allows you to select the fields you wish to expose in the CONFIGURE stage. In this case you wish to map only one field i.e. **Fahrenheit** into the request message for the Conversion Temperature operation of the service.

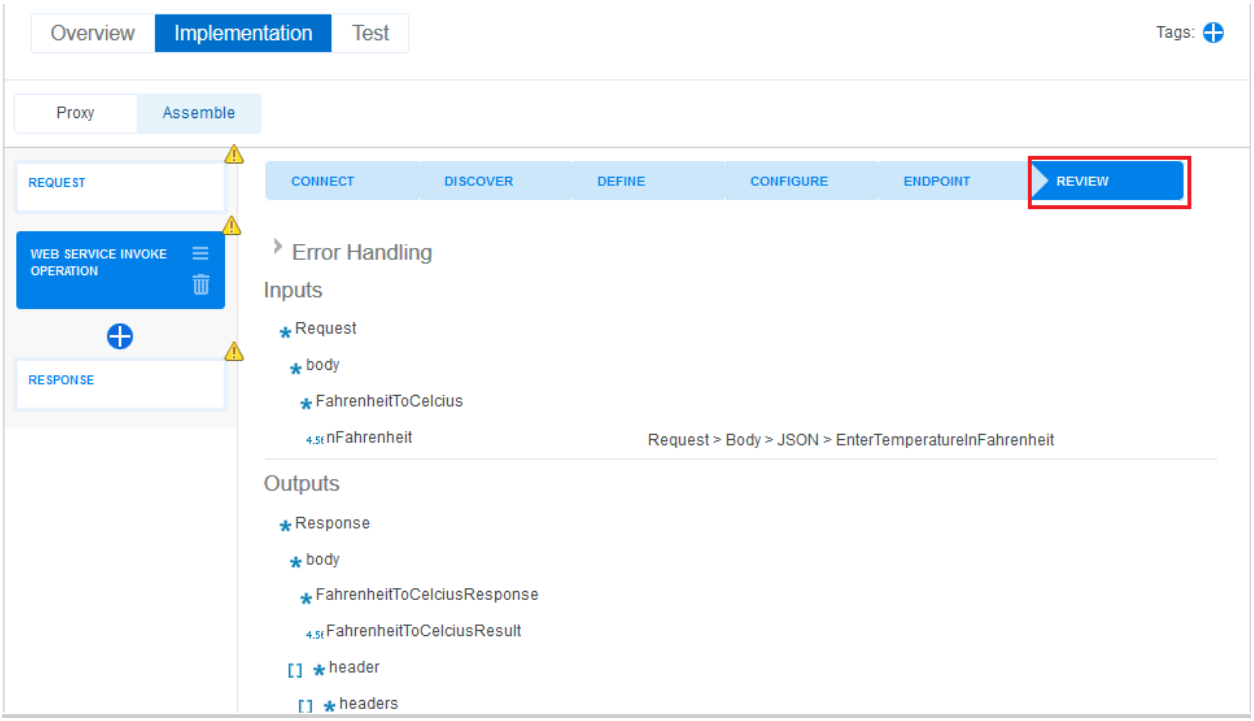


Select **CONFIGURE** to map the input parameter of your API to the input parameter of the Web Service Invoke Operation.

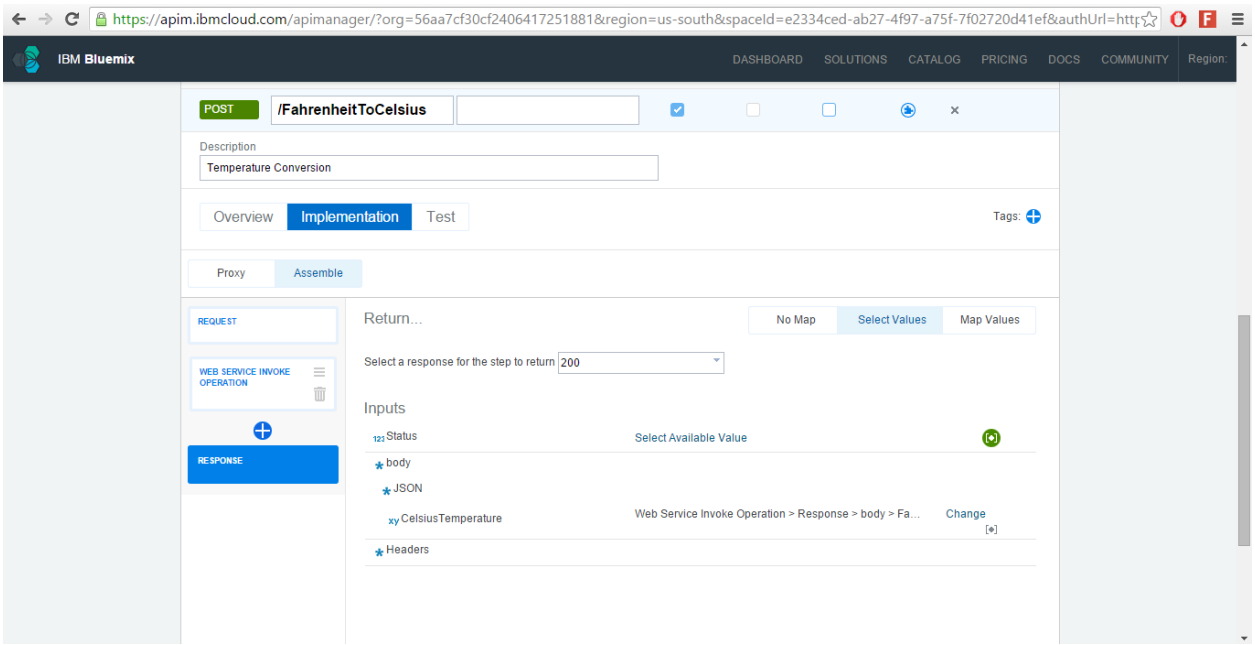
After selecting the available values select the **Map Values** tab you will be able to map the SOAP values to the JSON values in a drag-and-drop fashion.



Select **Review**. In the review section you can review the configuration as well as set specific actions if an error occurs when the Web Service Invoke Operation is called. For this document you will not take any actions if an error is returned.



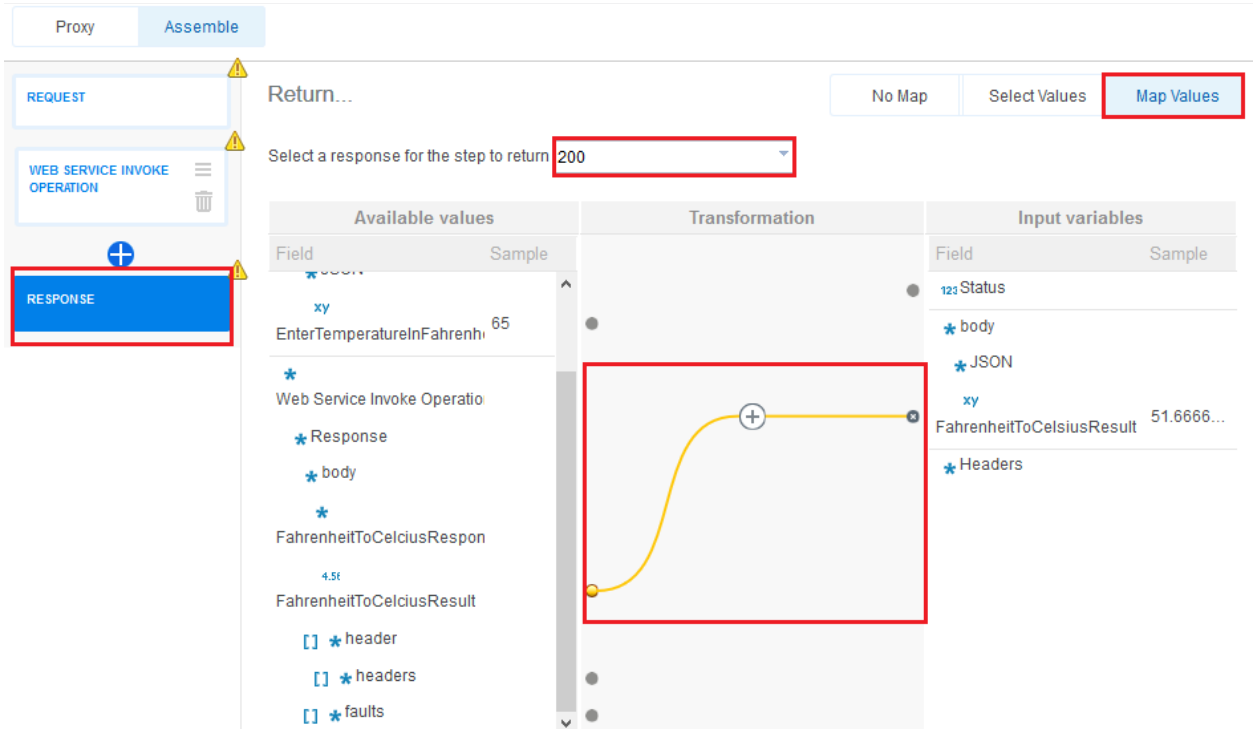
Select **Response tab** in the Implementation Module for Mapping the values to the Response Header.



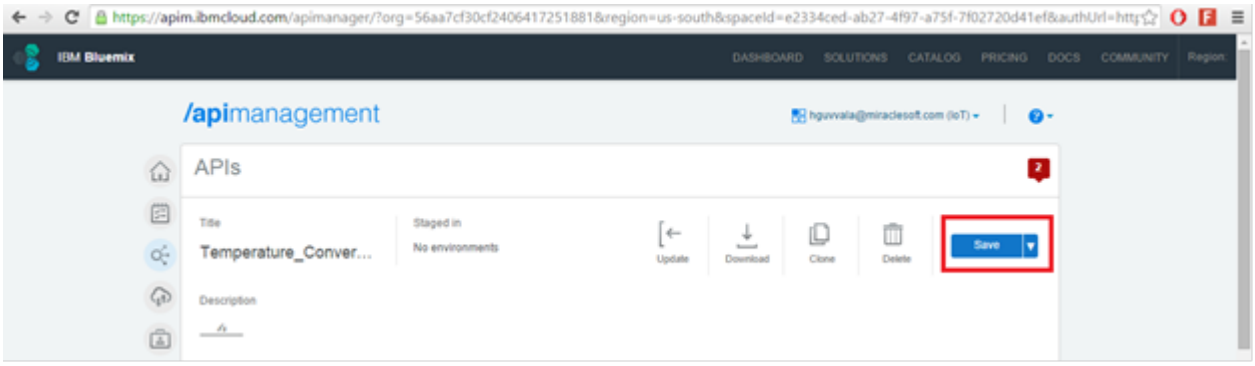
Select **Map Values** to see the alternative way to create mappings using the graphical mapping tool.

Click on the connector for the **FahrenheitToCelsiusResult** output field from the Web Service. Invoke Operation and drag and drop it on the connector for the **FahrenheitToCelsiusResult** field in the API response to map the two fields. Note that the mapping line is green which indicates that the two fields are compatible types and the mapping will succeed.

Map the values for Response as below.



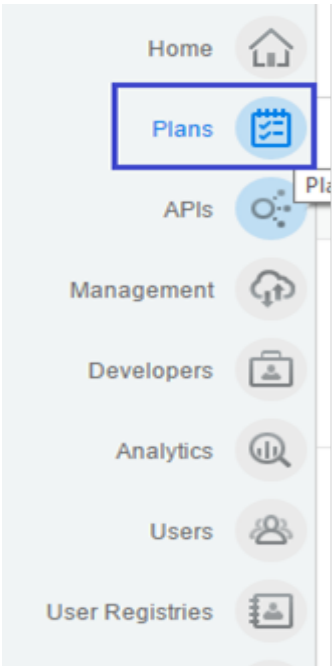
Click **Save** button at the top of the API editor.



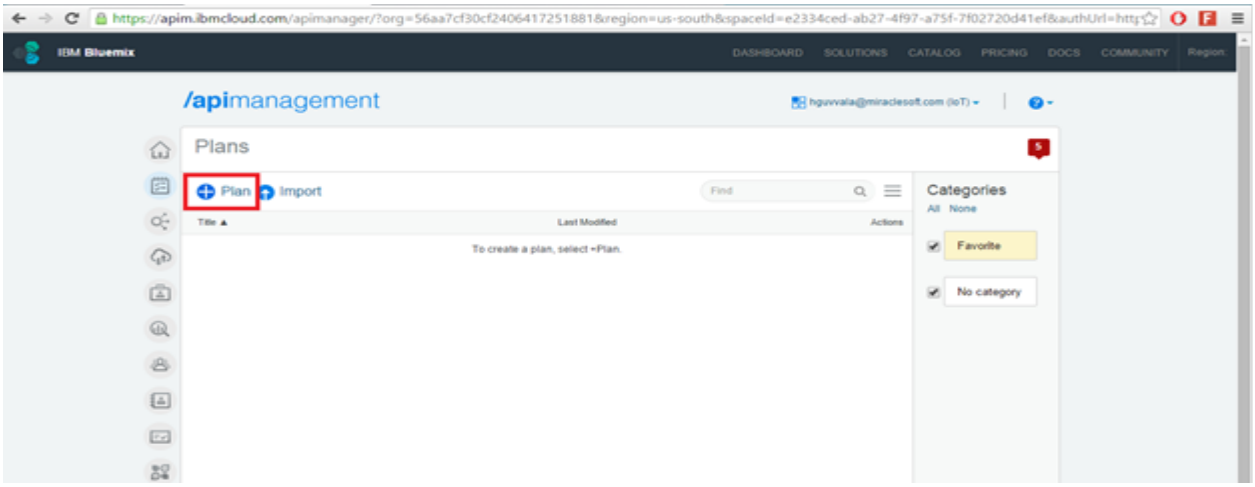
You have now defined your **CurrencyConverter_Mapping** REST API. Before you can test the resource from within the API Manager UI you will need to add it to a Plan.

Now, you need to add this Resource to the existing Sandbox Plan.

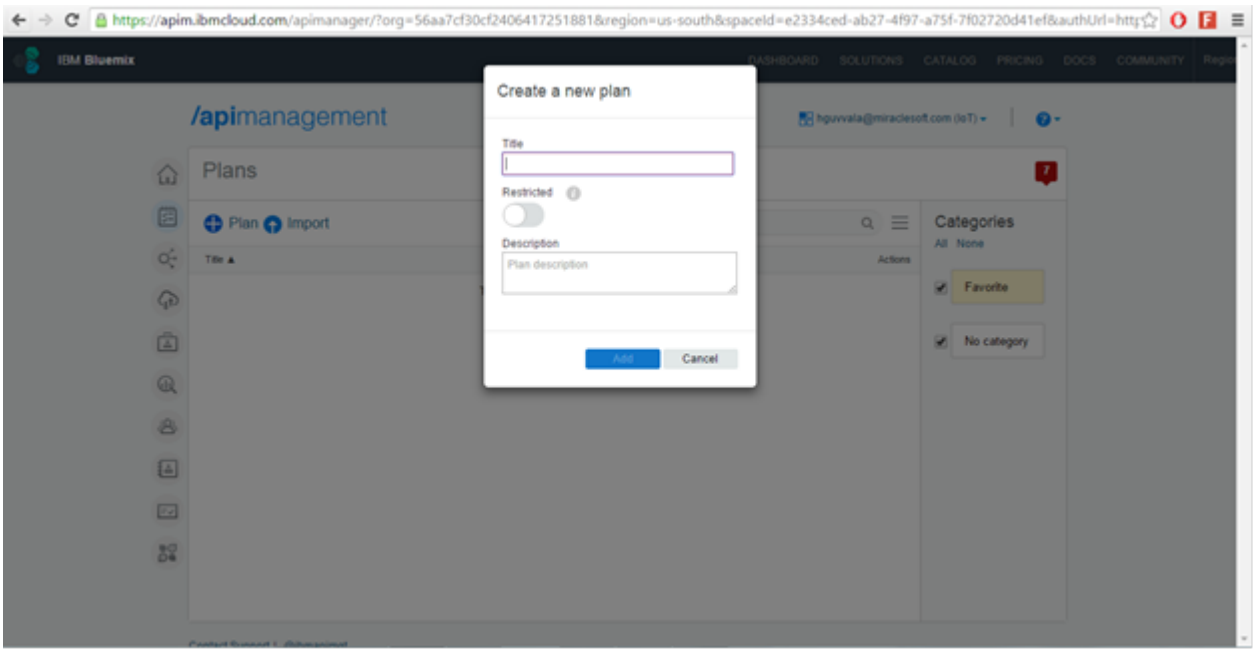
Select **Plans** from the Navigation Pane.

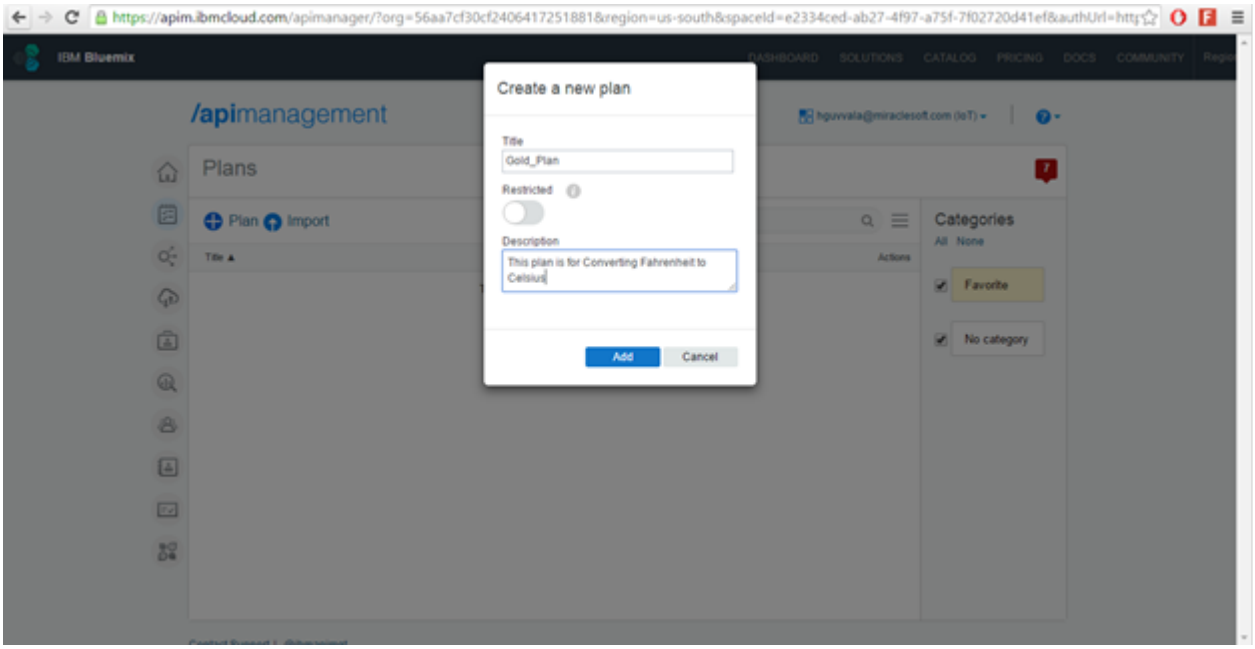


Click on **+Plan** for creating Plan.

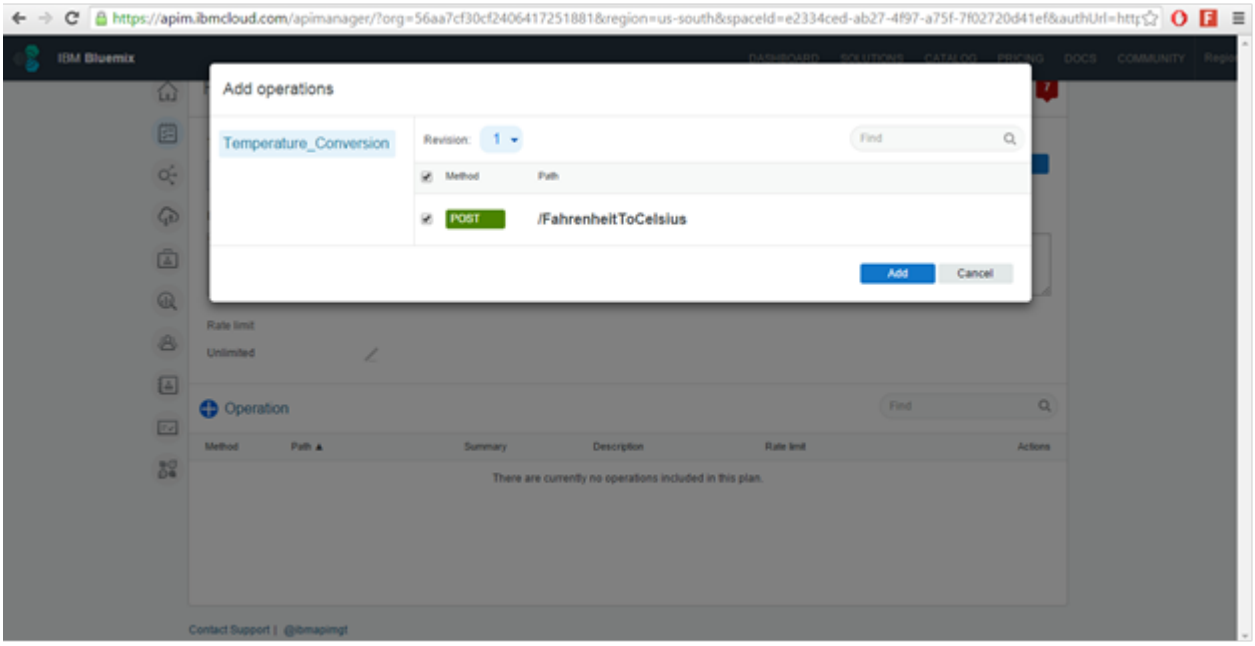


Specify the Plan name and Description for creating the plan.

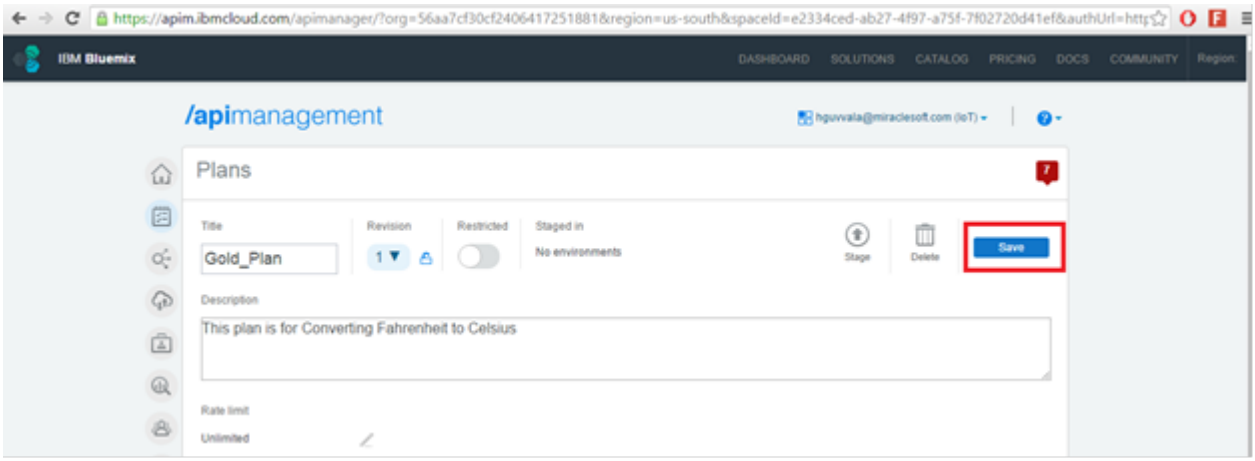




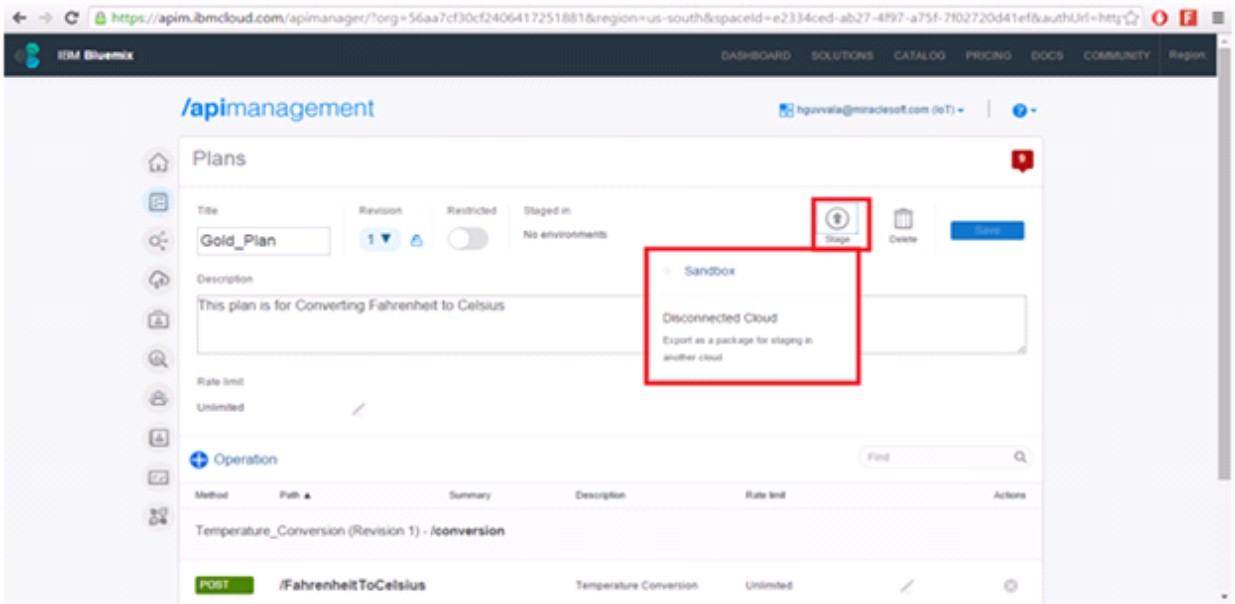
In order to expose the resource you have created you need to add it to the Public plan. Click **+ Operation**.



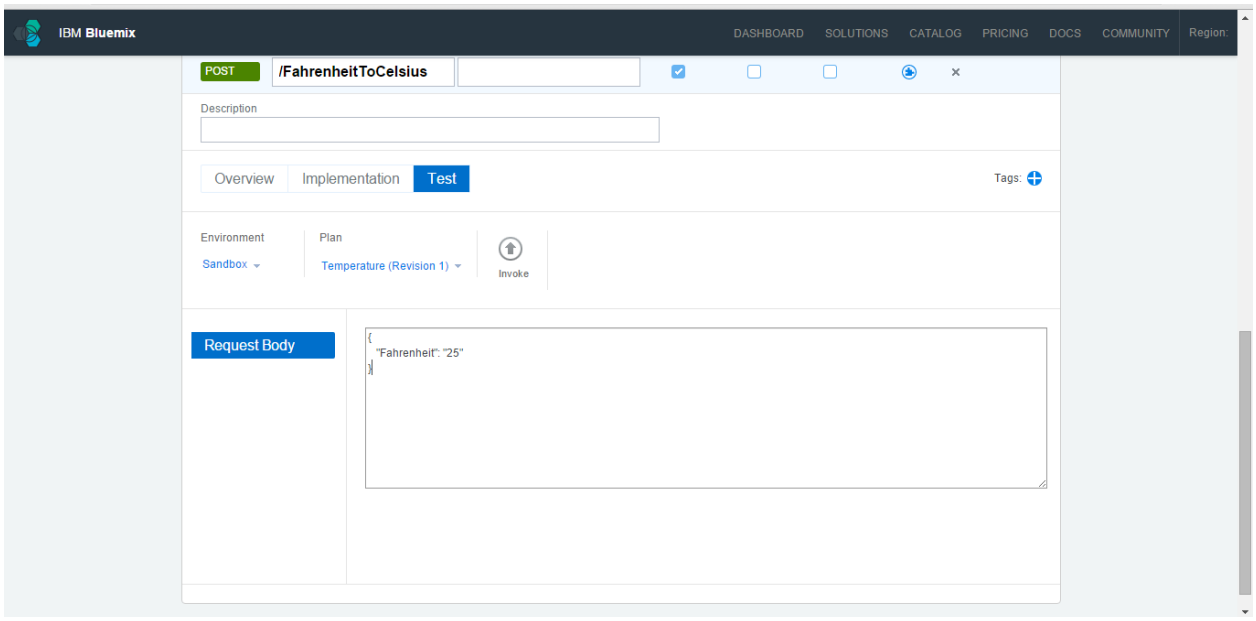
Click on Save to save the Plan.



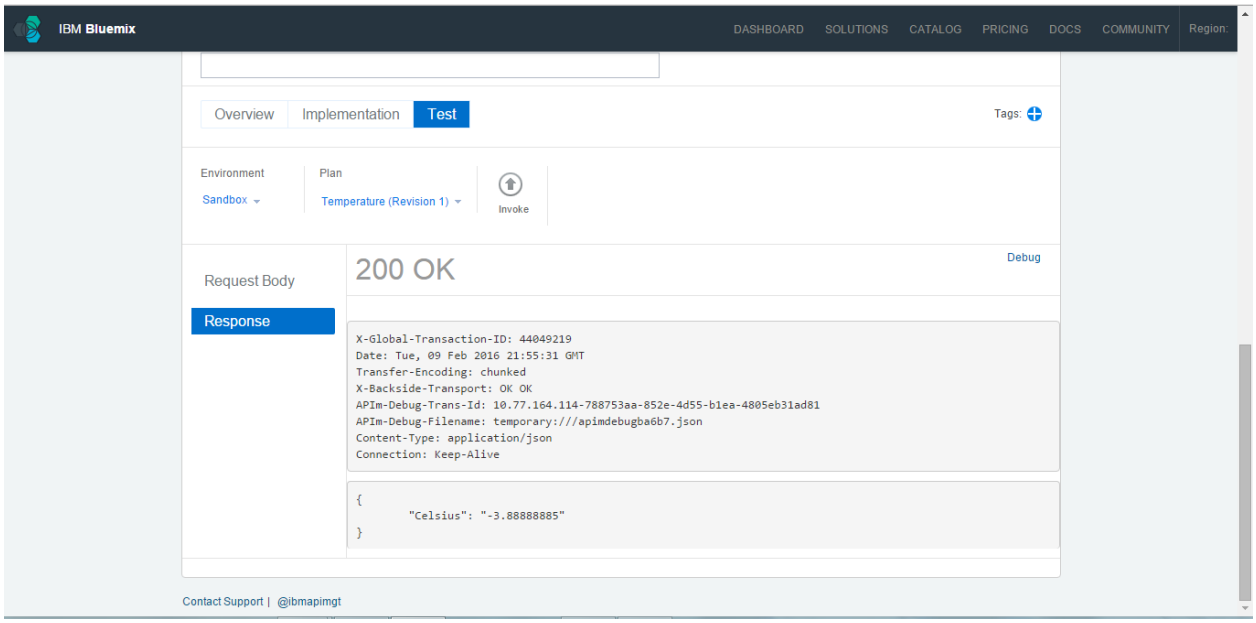
Now, you need to deploy this application to Sandbox Environment.



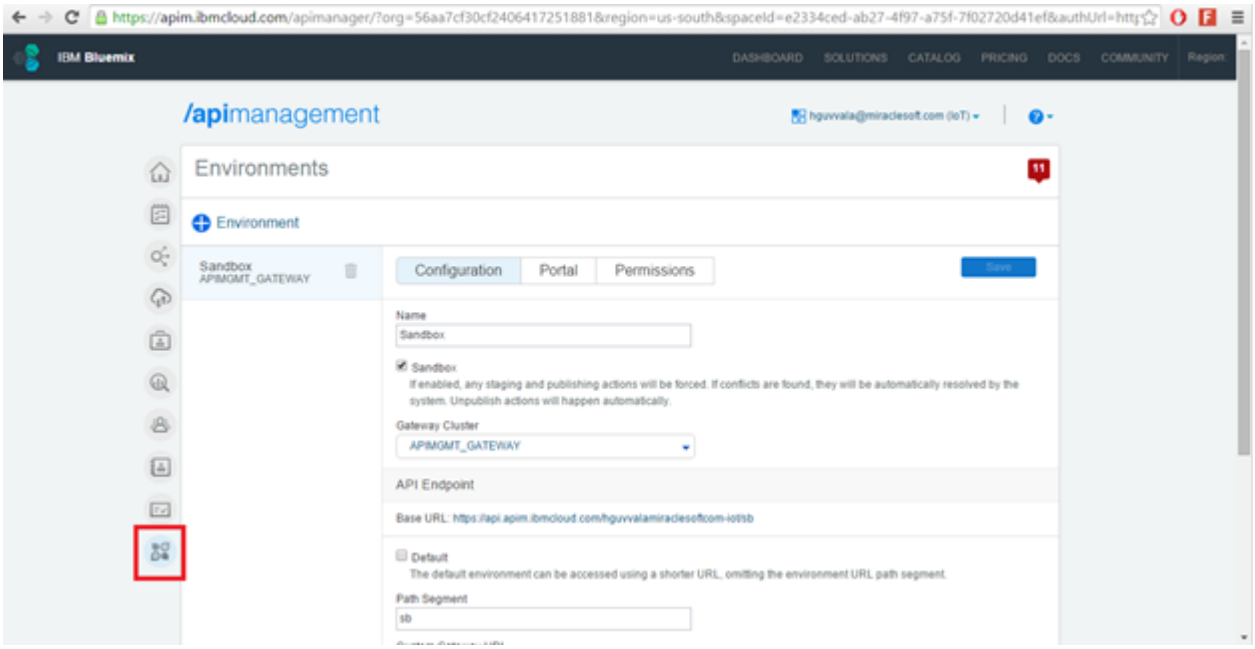
To test your REST resource to ensure that it is defined and implemented correctly, **Select APIs from the navigation pane**. Click the **Edit icon** for the POST quote resource. **Now, Test your API.**



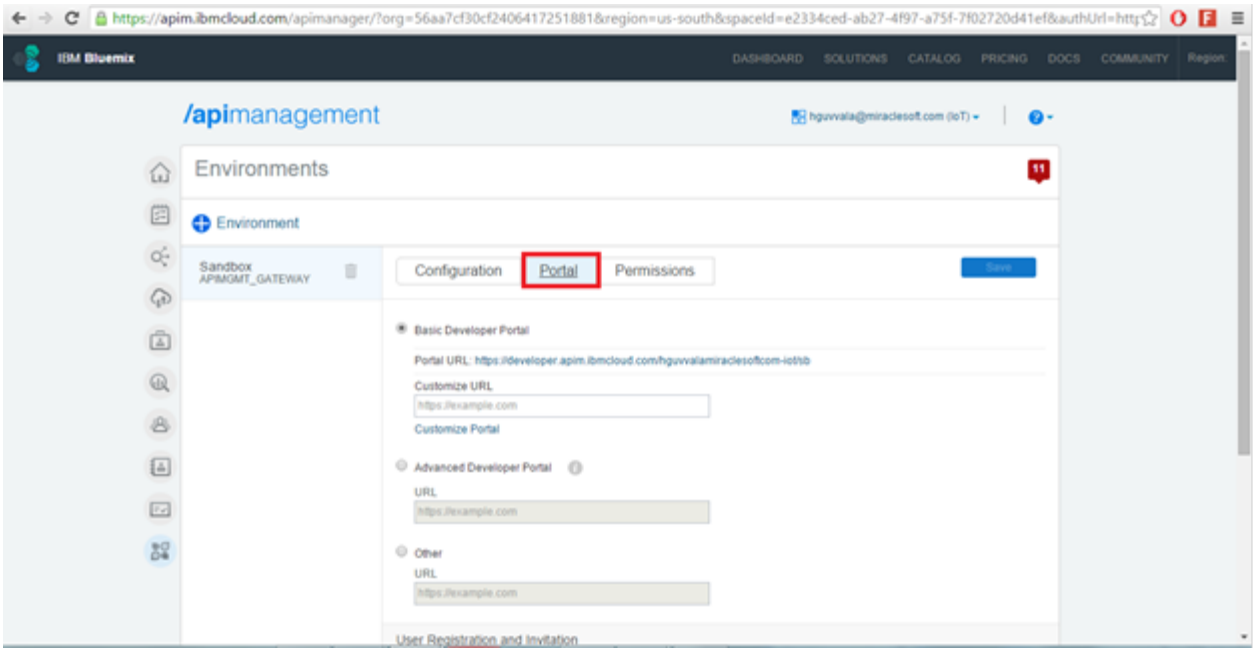
Click on Invoke to invoke your API.



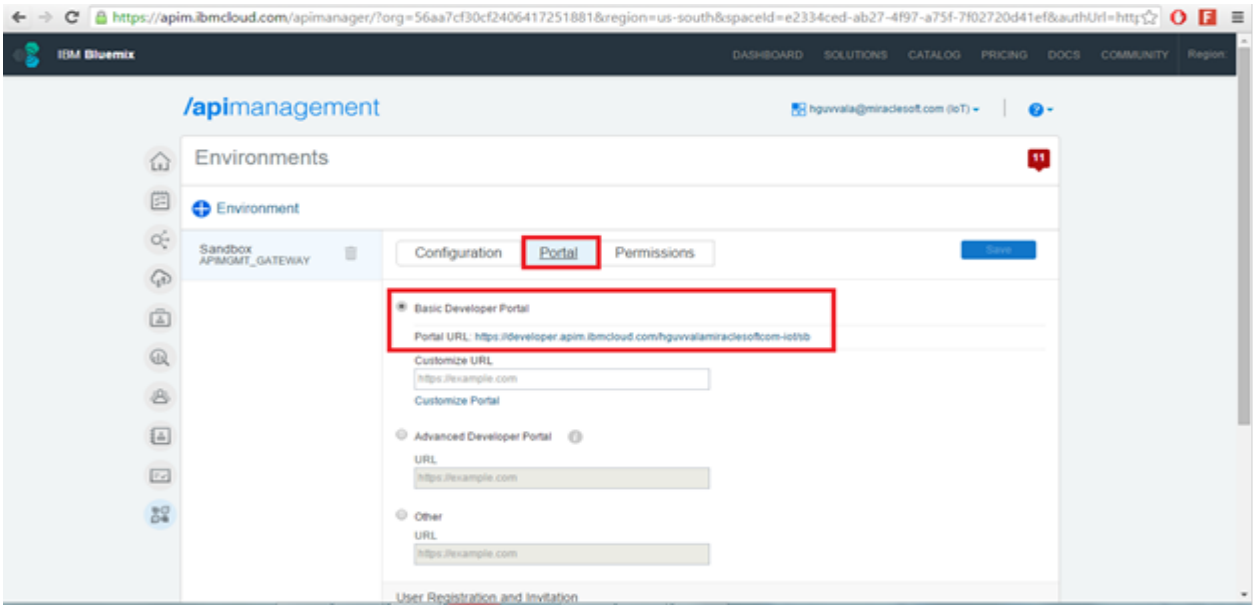
Once you get the response, then redirect to API Manager Console. In order to publish your API you need to redirect to Developer Console.



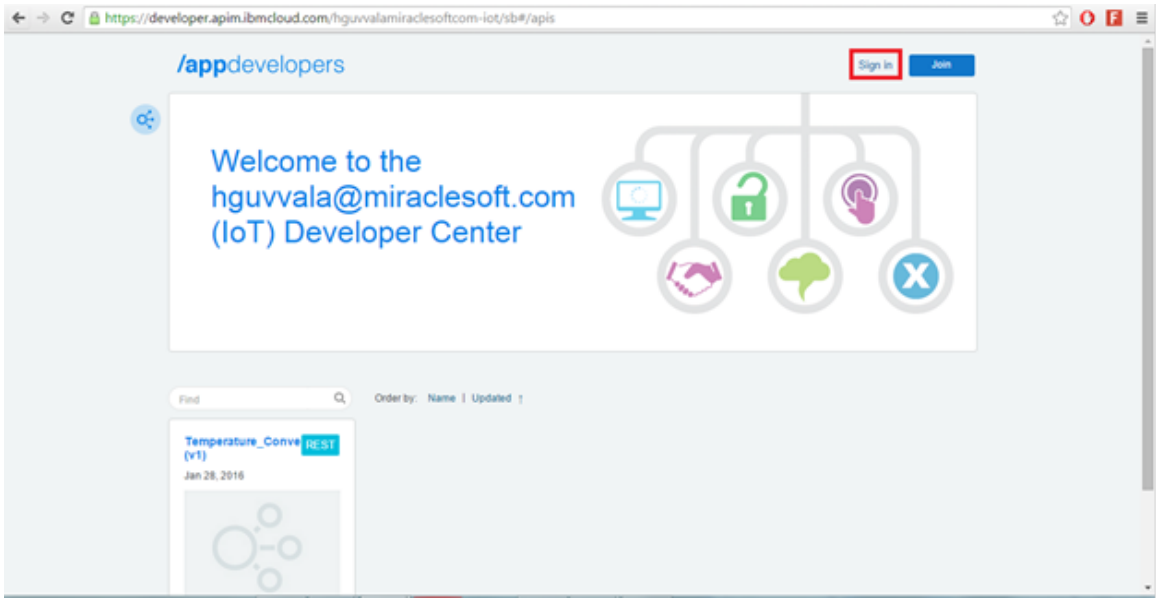
Click on **Portal** Tab.



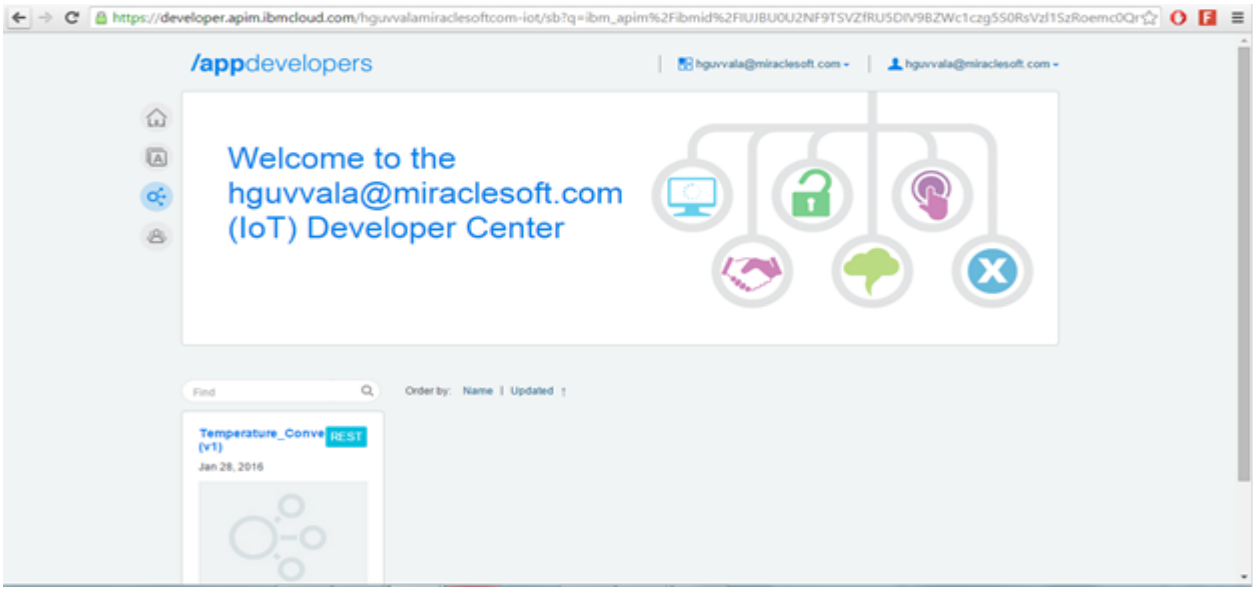
Click on the Basic Developer Portal checkbox and open the URL which was specified.



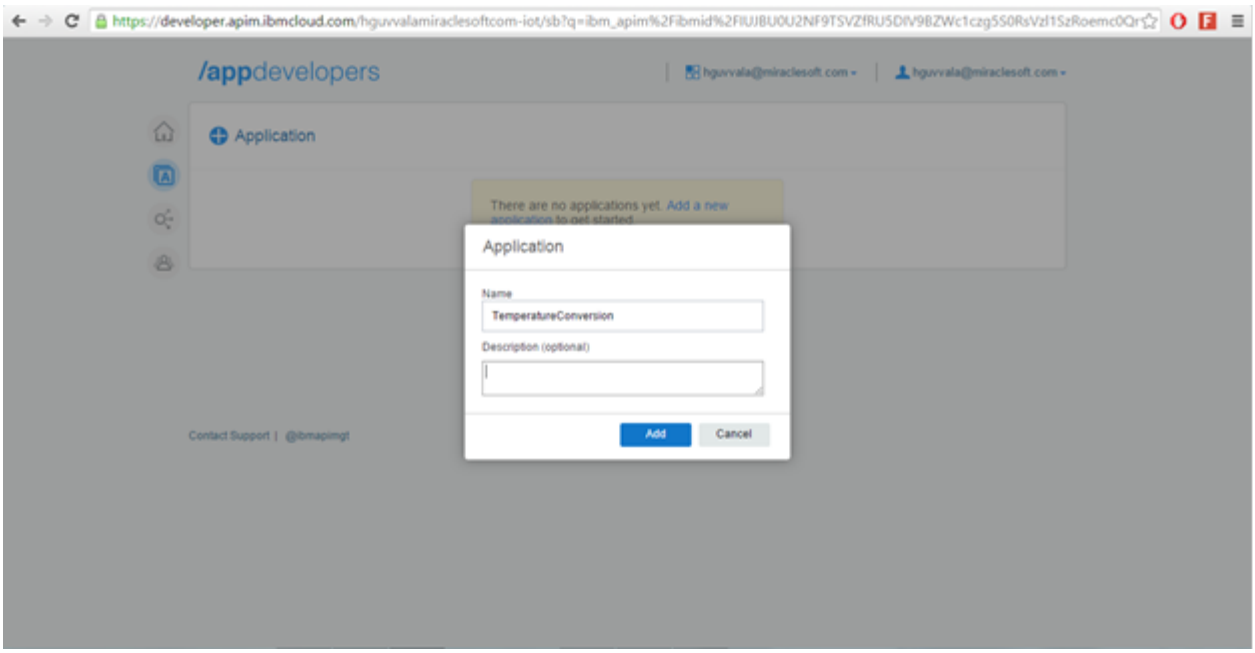
After clicking the URL you will be redirecting to API Developer Portal. Click on **SignIn**, it will automatically synchronize your credentials into Developer Portal and lets you login into that.



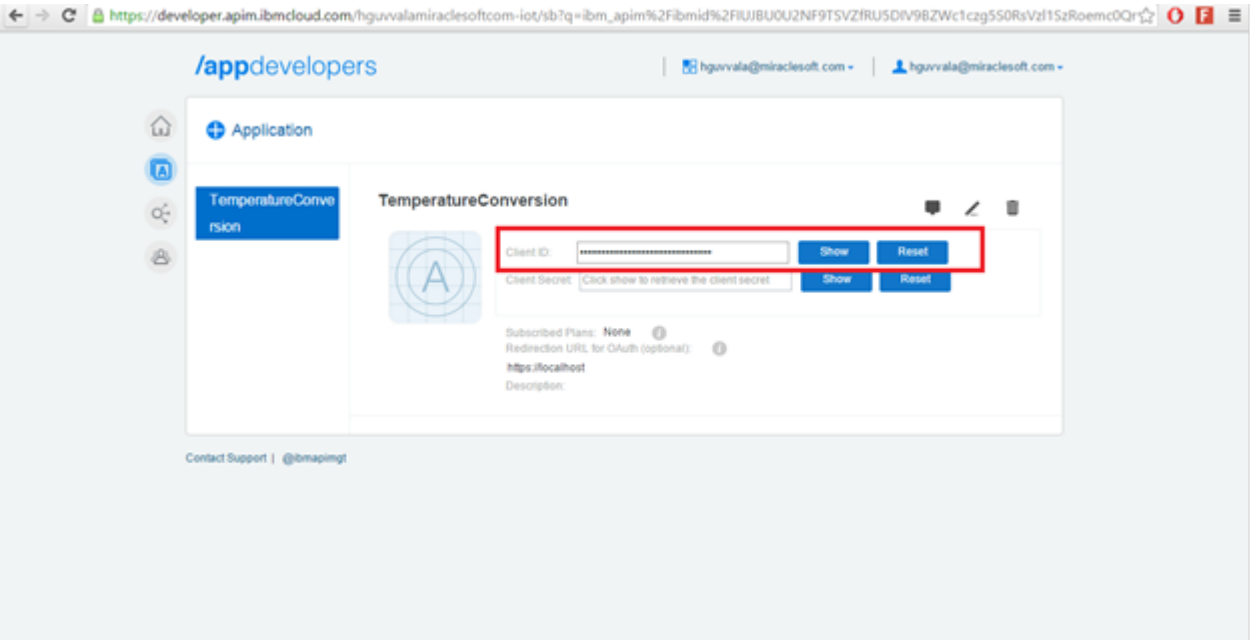
Click on **SignIn**.



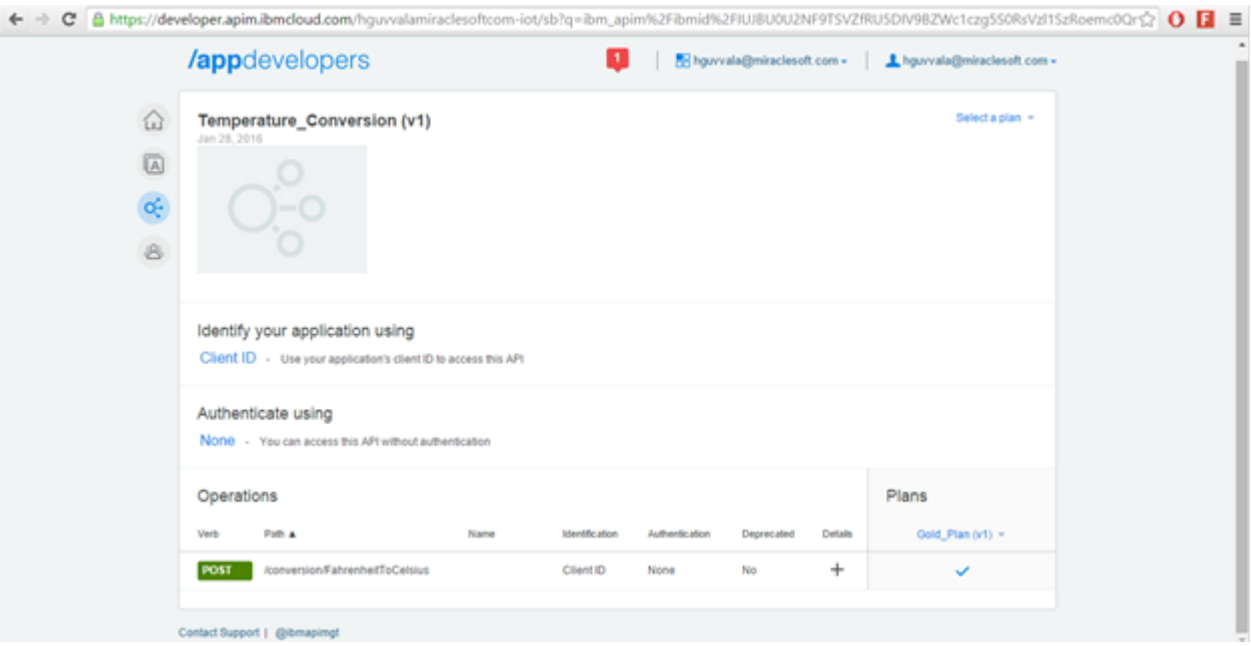
Click on **+Application** and specify Application Name and Description for the API and click Add Button.



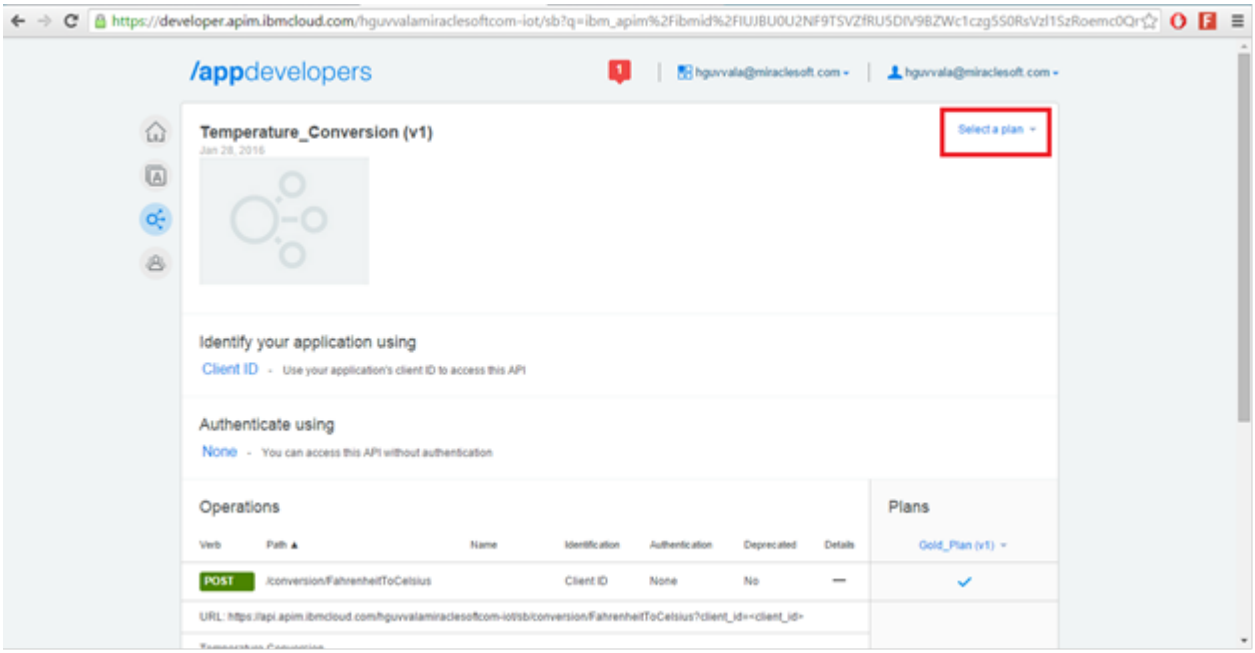
Click on the required API and copy the client ID as shown below.



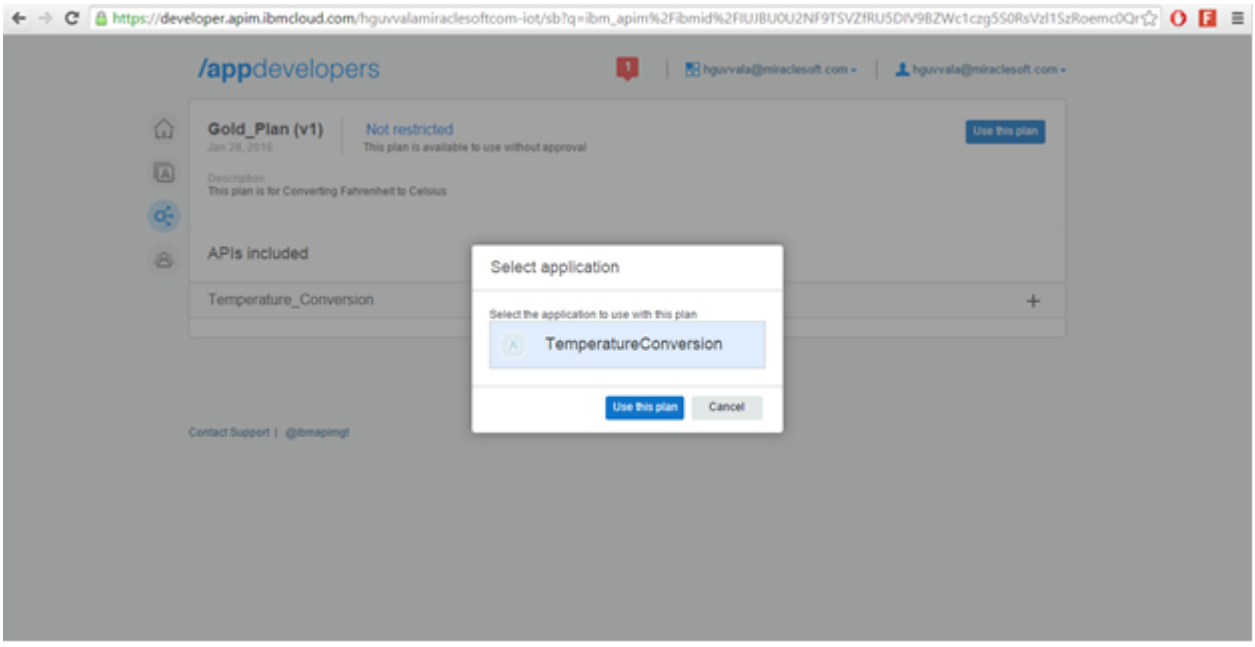
Go back to Developer Portal Console, Click on API's in Navigation Pane and open Temperature_Conversion API.



For invoking the API in Developer Portal, you need to add plan by clicking on **Select a plan** option.

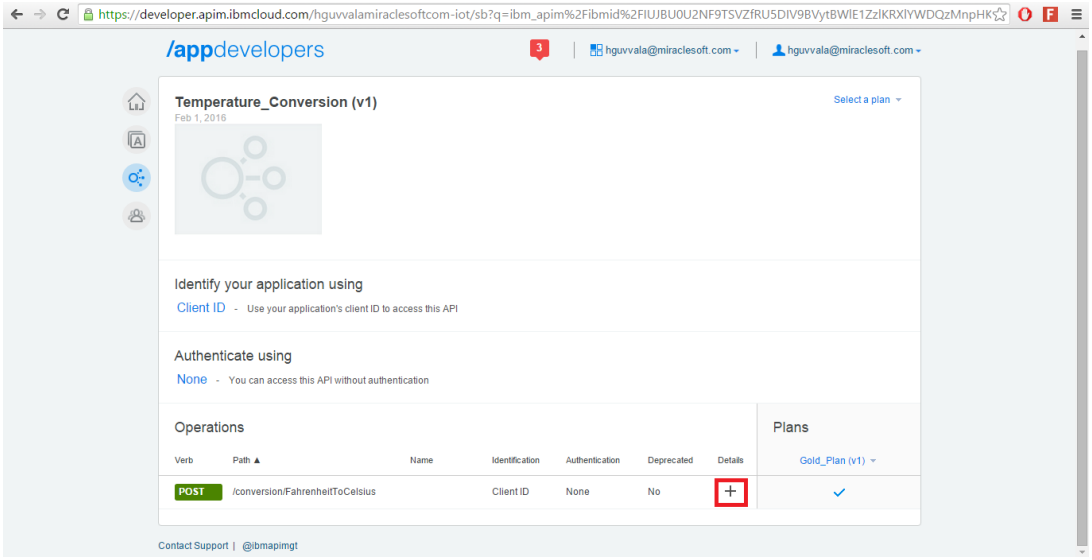


After selecting the plan, click on the application to which you are going to add, to the plan and then click “Use this plan”.

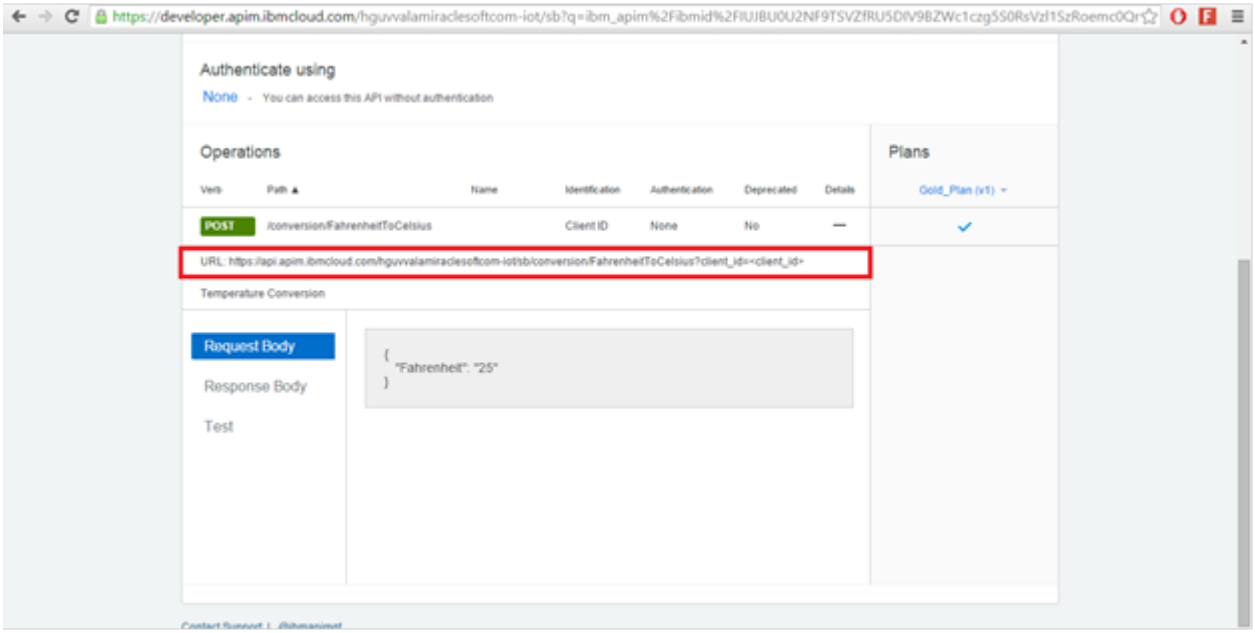


After adding the Application to the API, Click on API's and open Temperature_Conversion API. Test the API to get the response.

Click on plus(+) symbol to show the entire URL as shown in the figure.

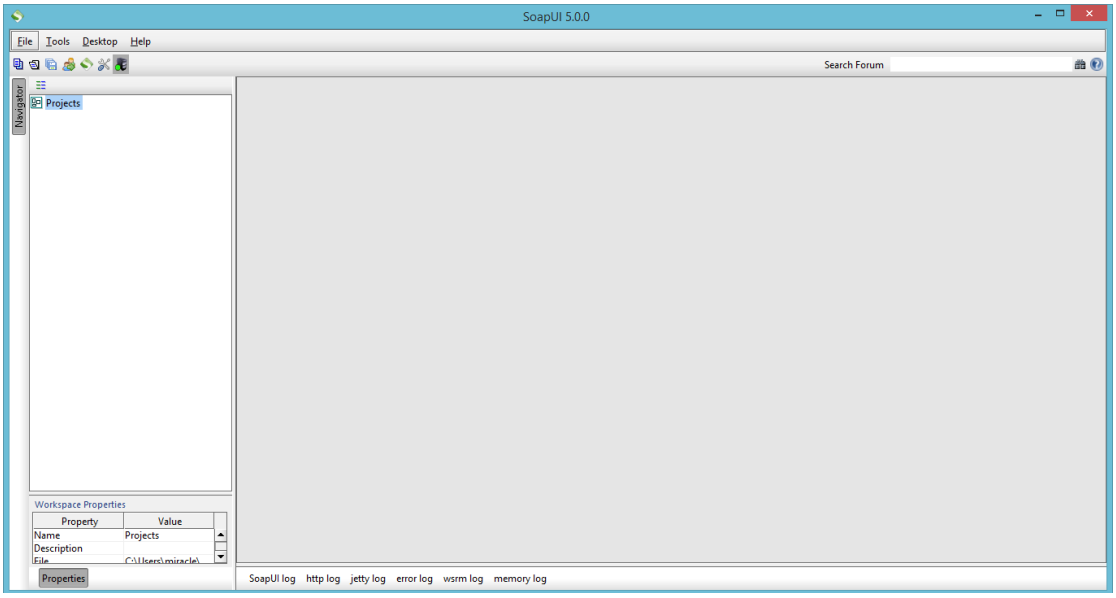


Here is the URL, which has to be consumed by Node.js Application. Copy the link.

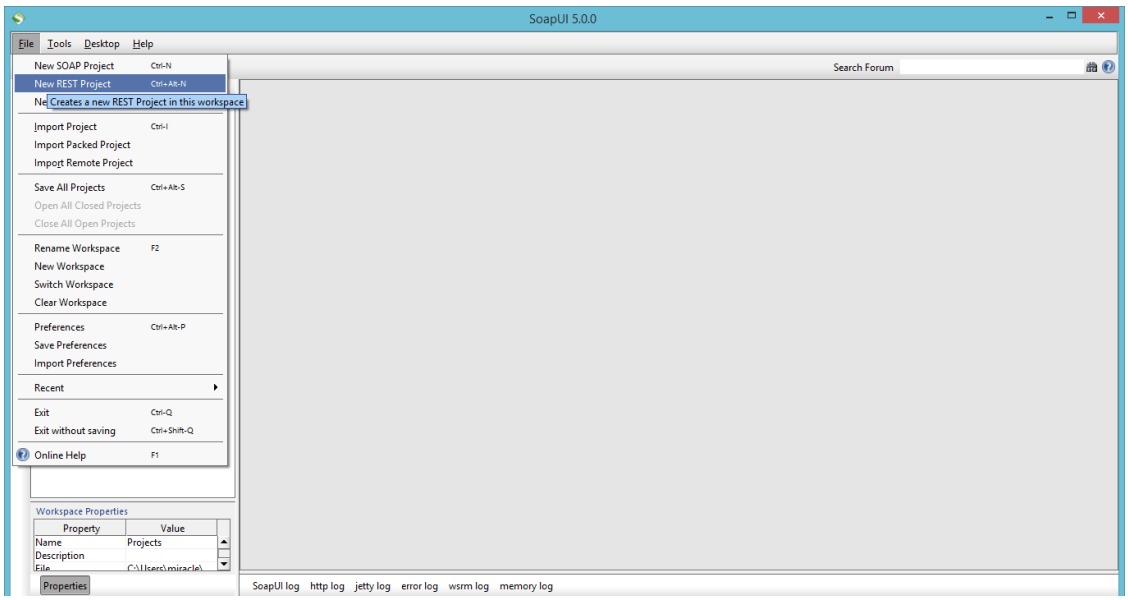


Replace the <client_Id> of the URL with the client id generated in the previous step. You can test that URL in SOAPUI. Double click on SOAPUI icon to open it.

The SOAPUI Dashboard will appear like this,



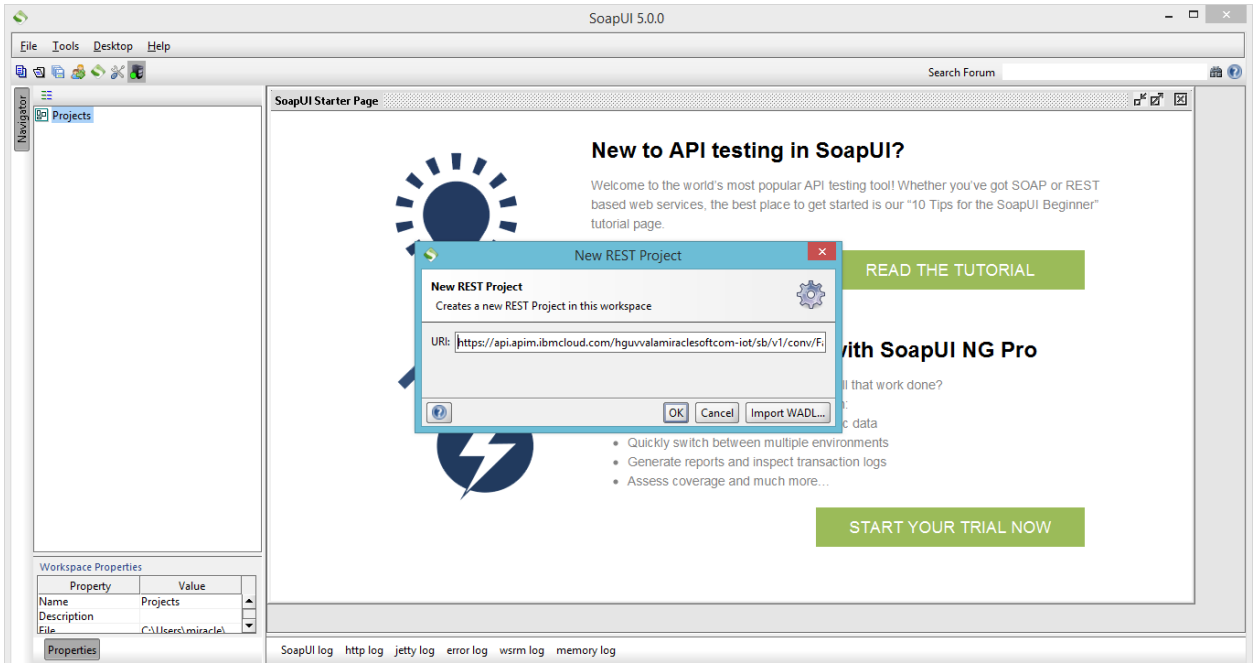
Create a new REST project from the File menu by choosing the "New REST Project" option in the File menu.



Specify the following API Management URL in the Service Endpoint Field

https://api.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb/v1/conv/FahrenheitToCelsius?client_id=<client-id>

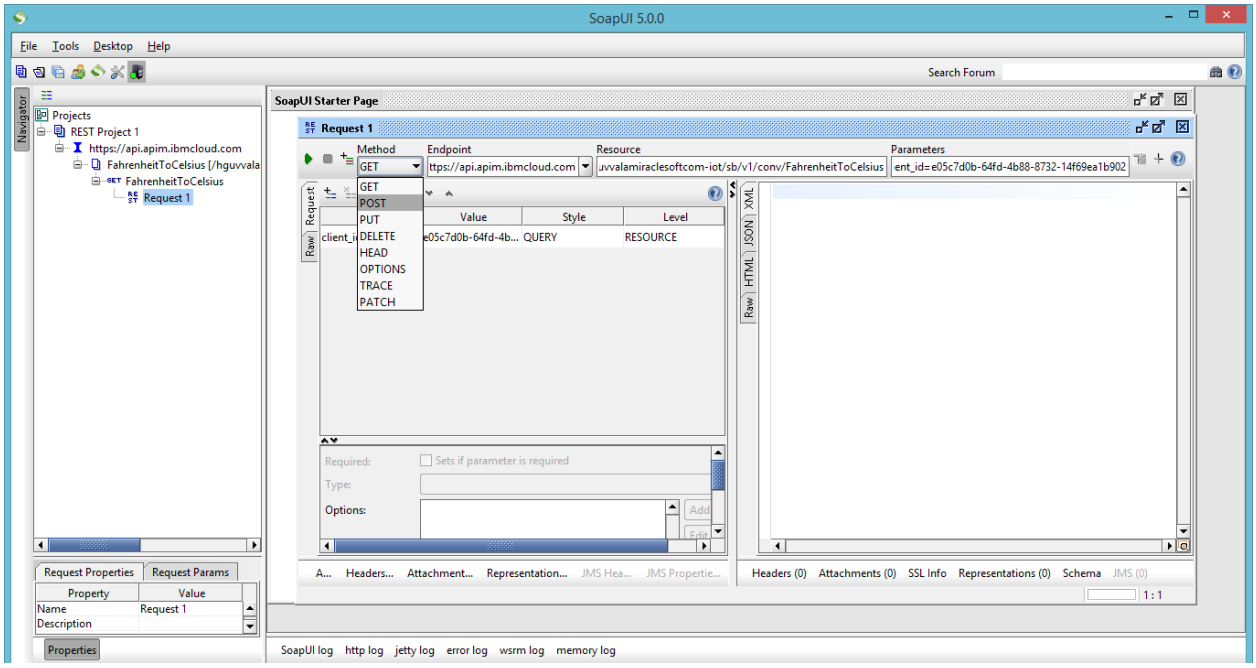
Specify the Client ID which was generated in IBM Developer Portal and the process okay.



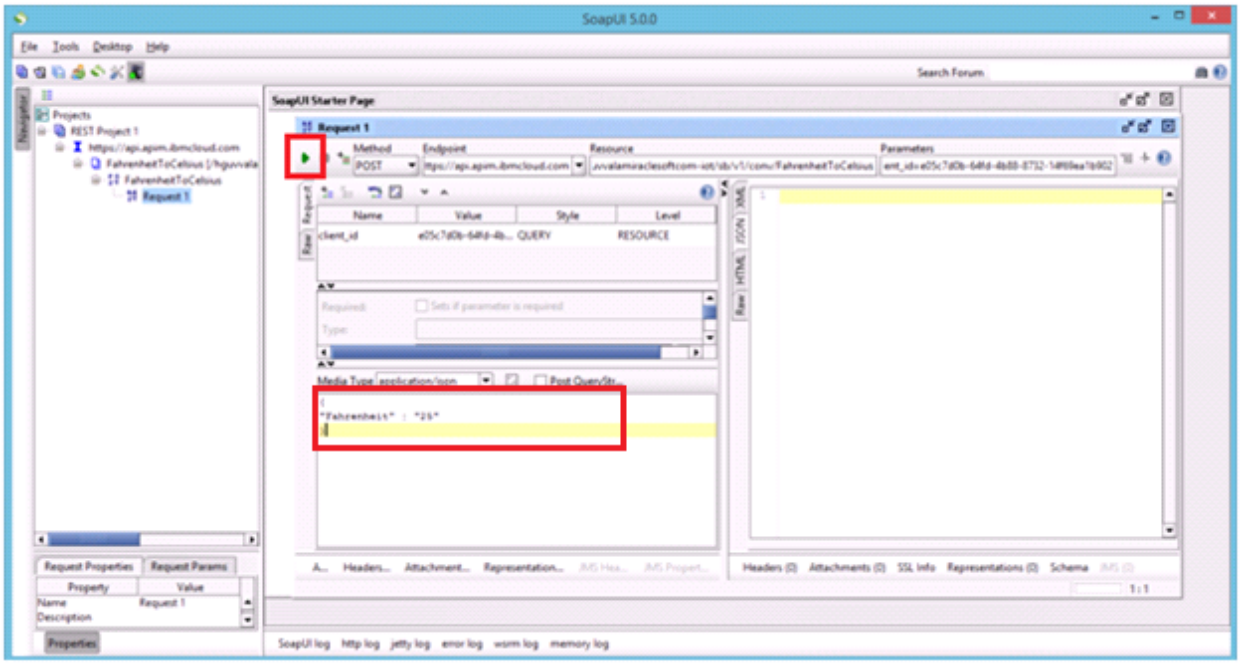
SOAPUI creates the project completely with a Service, Resource, Method and the actual Request and opens the Request editor.

By default, it will give GET method. Click on drop down list to change it to POST method.

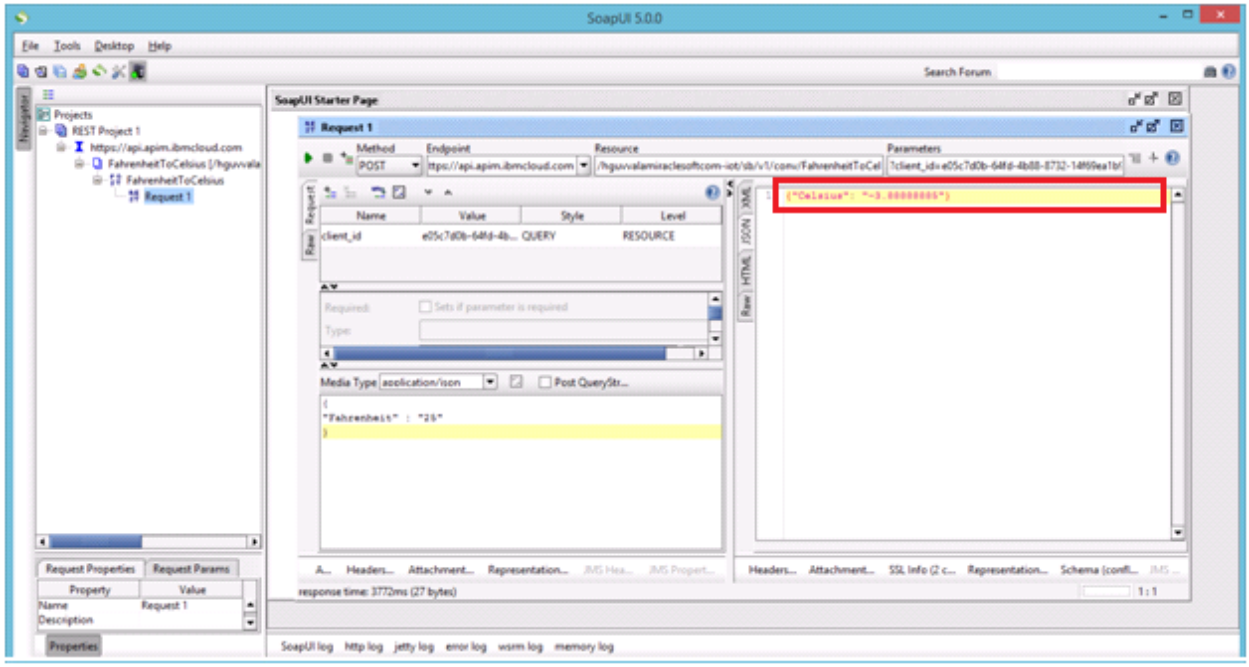
Click on JSON tab in the Right Navigation Pane for getting the result in JSON format.



Enter the Request Data in JSON format as **{ “Fahrenheit” : “25” }**

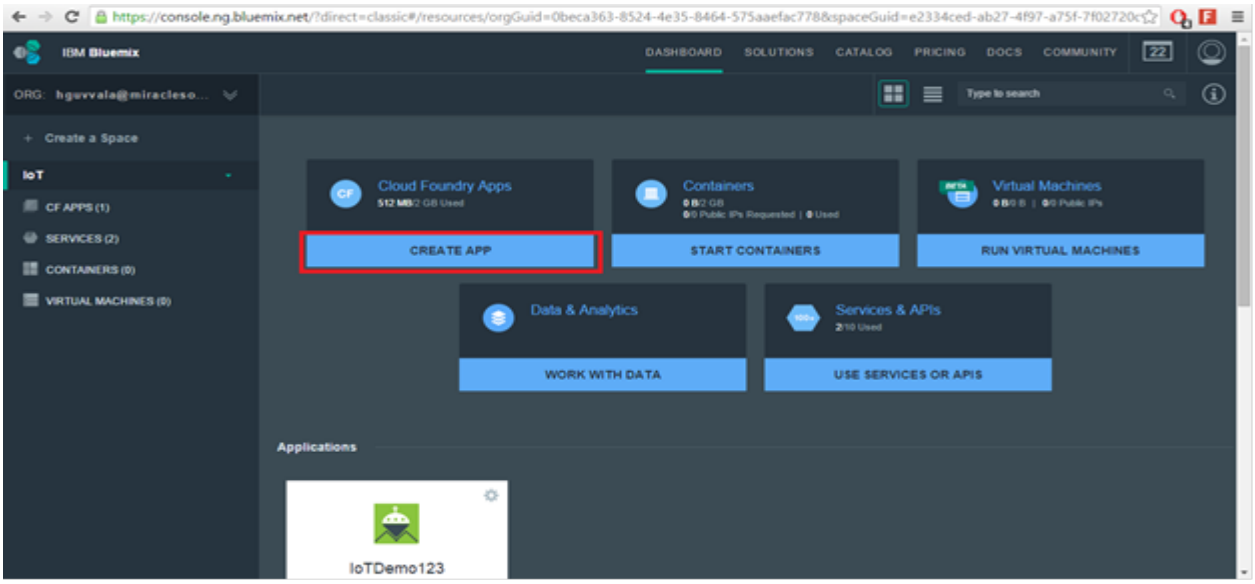


Press the green arrow at the top left in the Request editor and you can see the JSON output returned by the service.

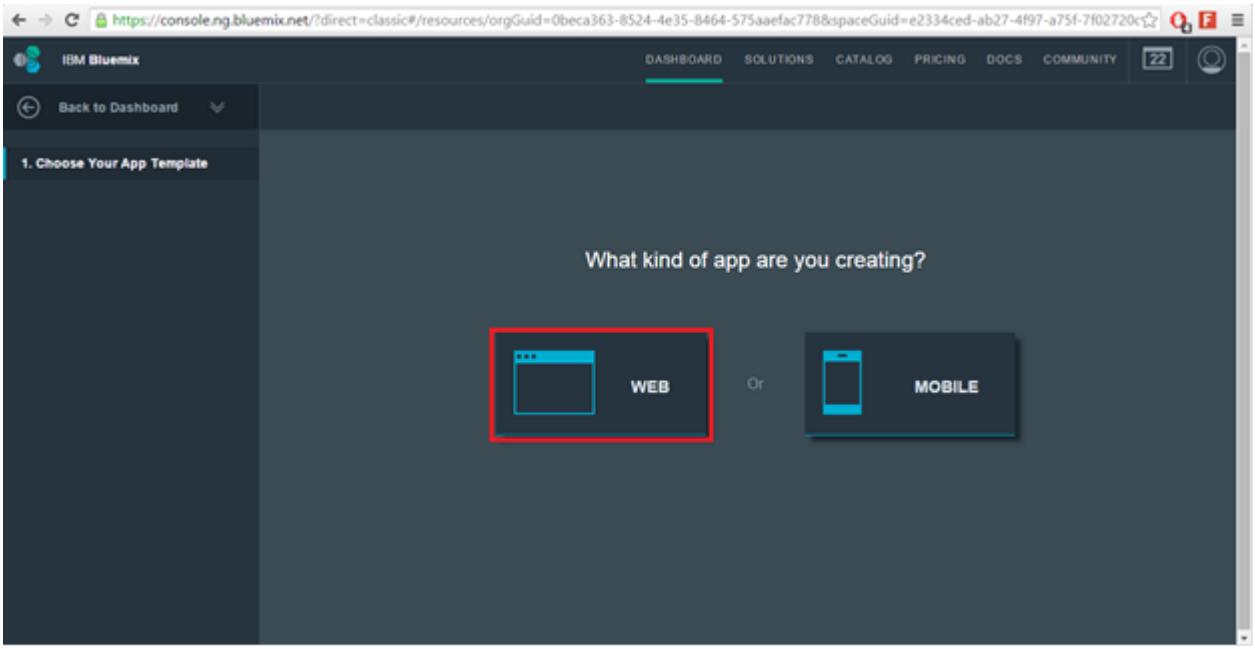


#5| Create Application in Bluemix

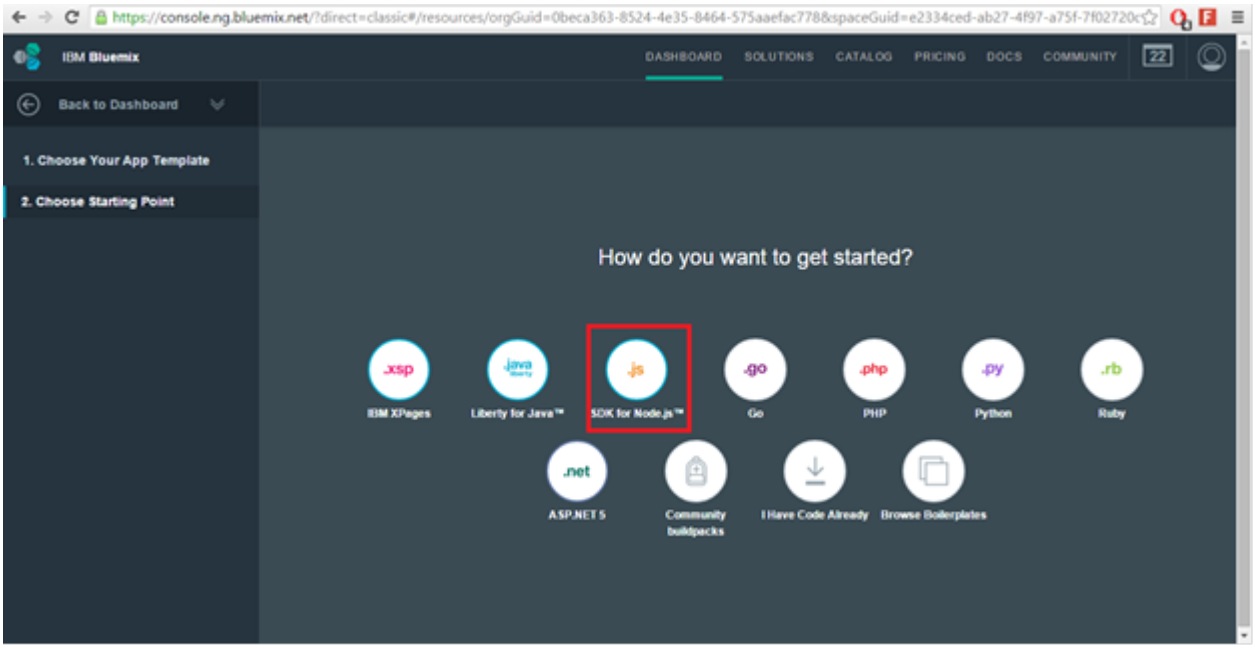
Click on the Cloud Foundry Apps module on the Dashboard to start creating the application.



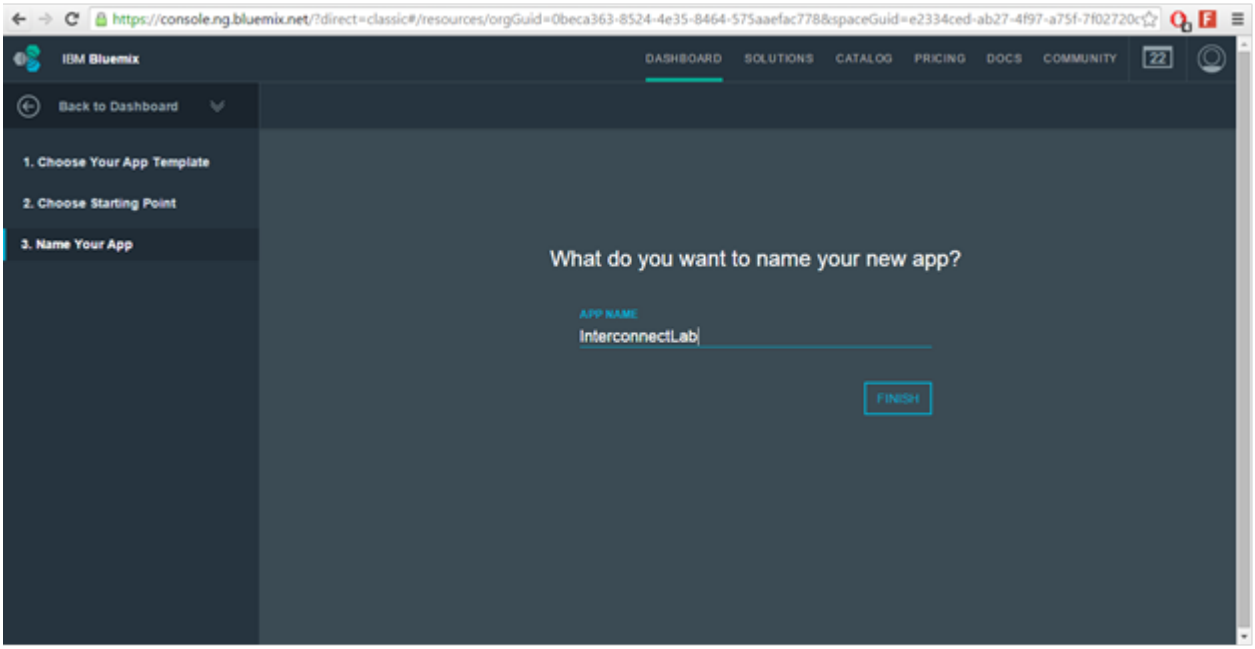
When prompted for the kind of application you want to create, please select Web Application.



Select "**SDK for Node.js**" in the page.

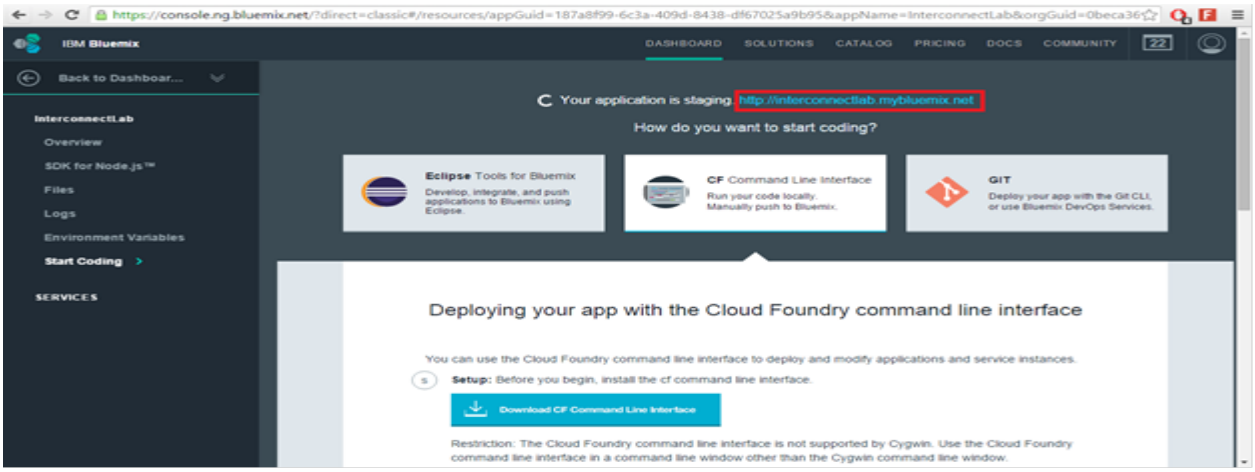


Click on continue and enter an application name and click finish. Application names must be unique as they will be on a public domain.

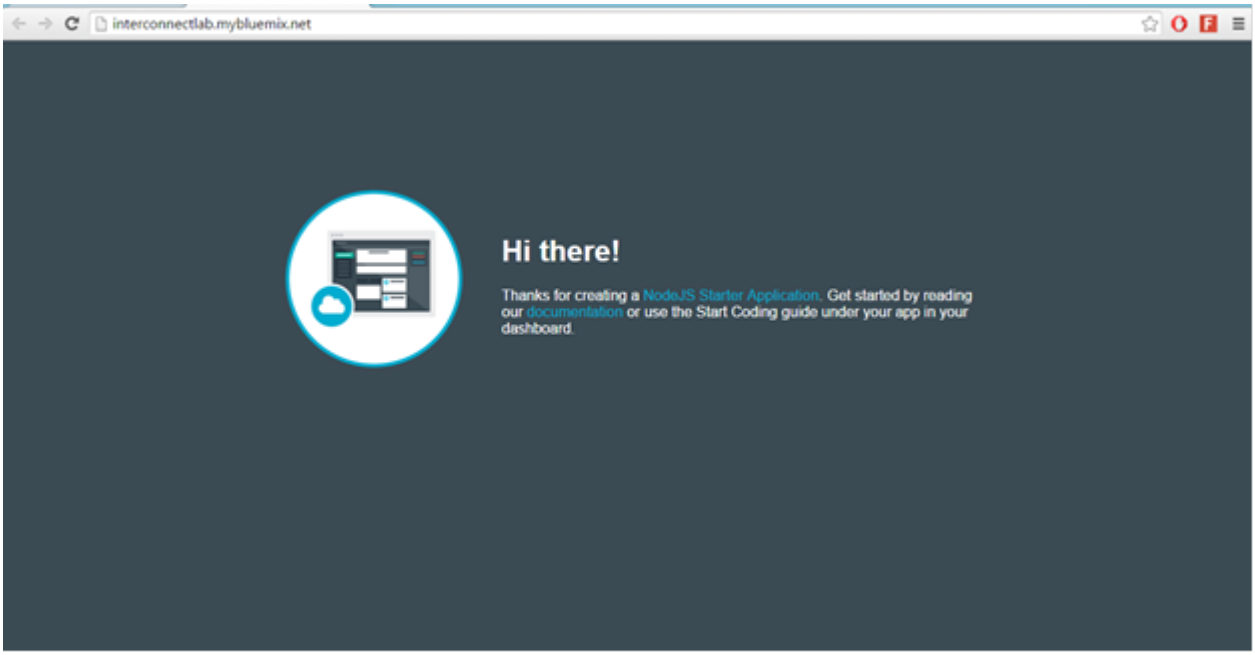


Note : Once created application will take about 2 minutes for staging and starting the application.

After creating the application you can use the application URL and view the sample application in the Browser. Click on the URL given on the page.

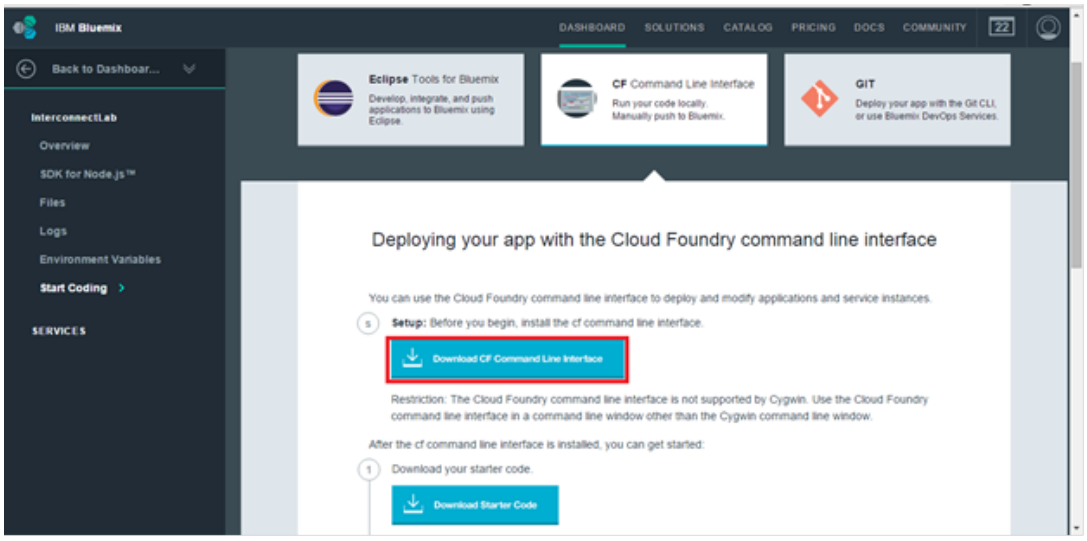


After clicking the URL you can view the page as below.

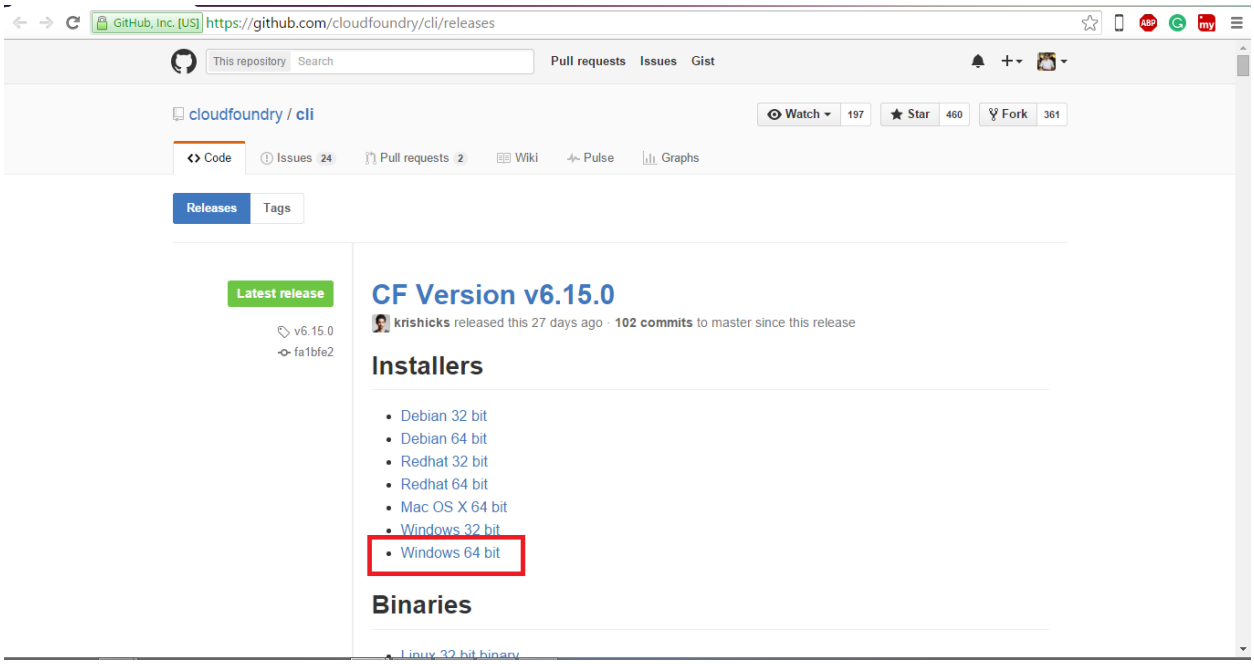


#6 | Download and Edit Starter Code

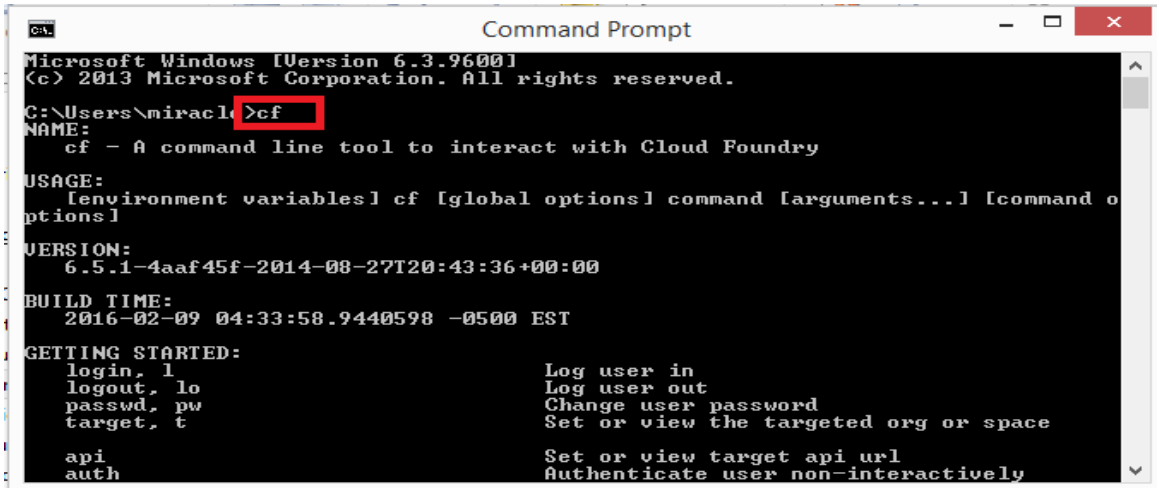
At this point, you will be asked how you want to start coding your application. For this lab, you will be using the Cloud Foundry (CLI) option. Download and Install the **CF CLI** Tool (any version) as the next step.



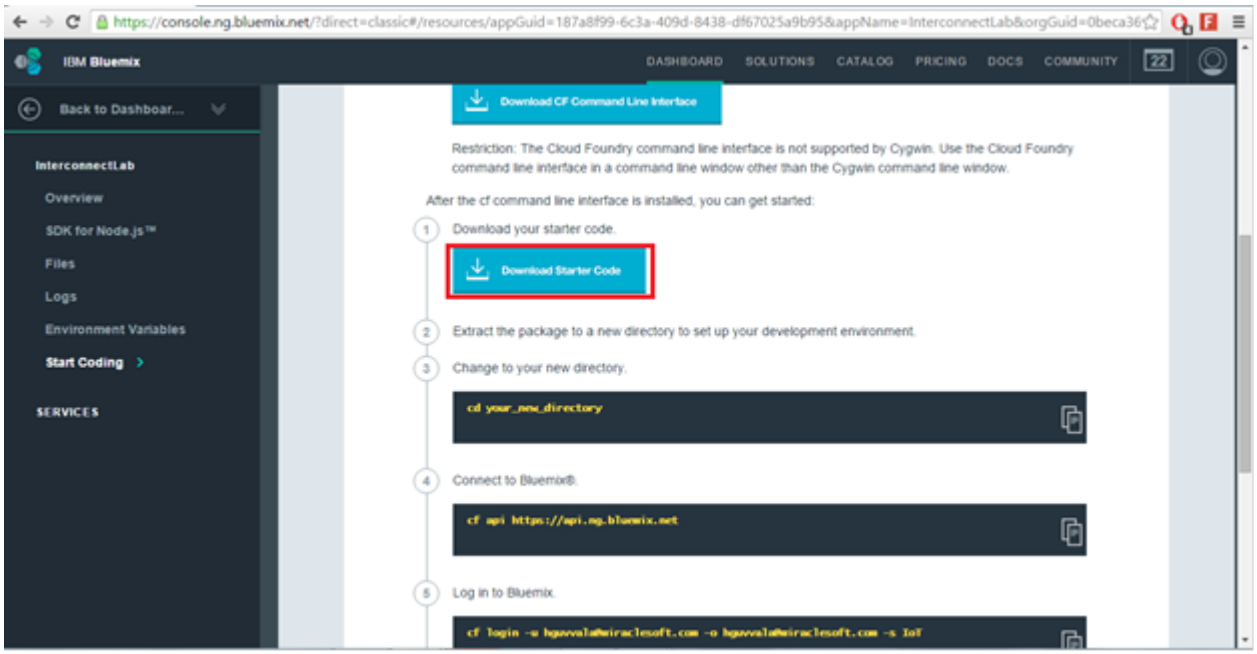
For this, Click on “Download CF Command Line Interface” then it will open a new webpage, where you can download your CF installer file. Select the Installer file which suits your OS.



Then you will get a zip file. After extracting the zip file, you can find a **.exe** file inside. **Install it.** To check whether **CF** is installed properly or not, open command prompt and execute CF command. Then it will show you a set of **CF** commands, which indicates that CF is successfully installed on your machine.



Once you have CF CLI running on your machine go ahead and download the Starter Application code for your application. After the application is downloaded, save to your working directory and extract the zip download (unzip).



#7 | Let's Start Coding

Once you have the started files, go to your application folder in your local file system. Delete the index.html file which is present in **Public** folder. Our goal will be to modify some pages in the downloaded starter code folder,

Open "***package.json"*** file using an editor (Notepad++). Replace the code in ***"package.json"*** file file with the following content,

---

```
{
  "name": "NodejsStarterApp",
  "version": "0.0.1",
  "description": "A sample nodejs app for Bluemix",
  "scripts": {
```

```
    "start": "node app.js"
  },
  "dependencies": {
        "cfenv": "^1.0.3",
    "ejs": "^2.4.1",
    "express": "4.12.x",
    "https": "^1.0.0",
    "path": "^0.12.7",
    "request-json": "^0.5.5"
  },
  "repository": {},
  "engines": {
    "node": "0.12.x"
  }
}
```

Note : This above code is used to add dependencies which are used in this application.

Open ***app.js*** file using an editor (Notepad++). Replace the code in ***app.js*** file with the following content,

---

```
/*eslint-env node*/

// This application uses express as its web server
// for more info, see: http://expressjs.com
var express = require('express');
var path=require('path');
// cfenv provides access to your Cloud Foundry environment
// for more info, see: https://www.npmjs.com/package/cfenv
var cfenv = require('cfenv');

// create a new express server
```

```
var app = express();

// serve the files out of ./public as our main files
app.use(express.static(__dirname + '/public'));
var routes = require('./routes/index');
var users = require('./routes/users');
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use('/', routes);
app.use('/users', users);
// get the app environment from Cloud Foundry
var appEnv = cfenv.getAppEnv();

// start server on the specified port and binding host
app.listen(appEnv.port, '0.0.0.0', function() {

        // print a message when the server starts listening
  console.log("server starting on " + appEnv.url);
});
```

Go to Public folder in your application and open Stylesheets folder. Open ***"style"*** file (CSS) using an editor (Notepad++). Replace the code in ***"style"*** file (CSS) with the following content,

---

```
body {
  padding: 50px;
  font: 14px "Lucida Grande", Helvetica, Arial, sans-serif;
}

a {
  color: #00B7FF;
```

}

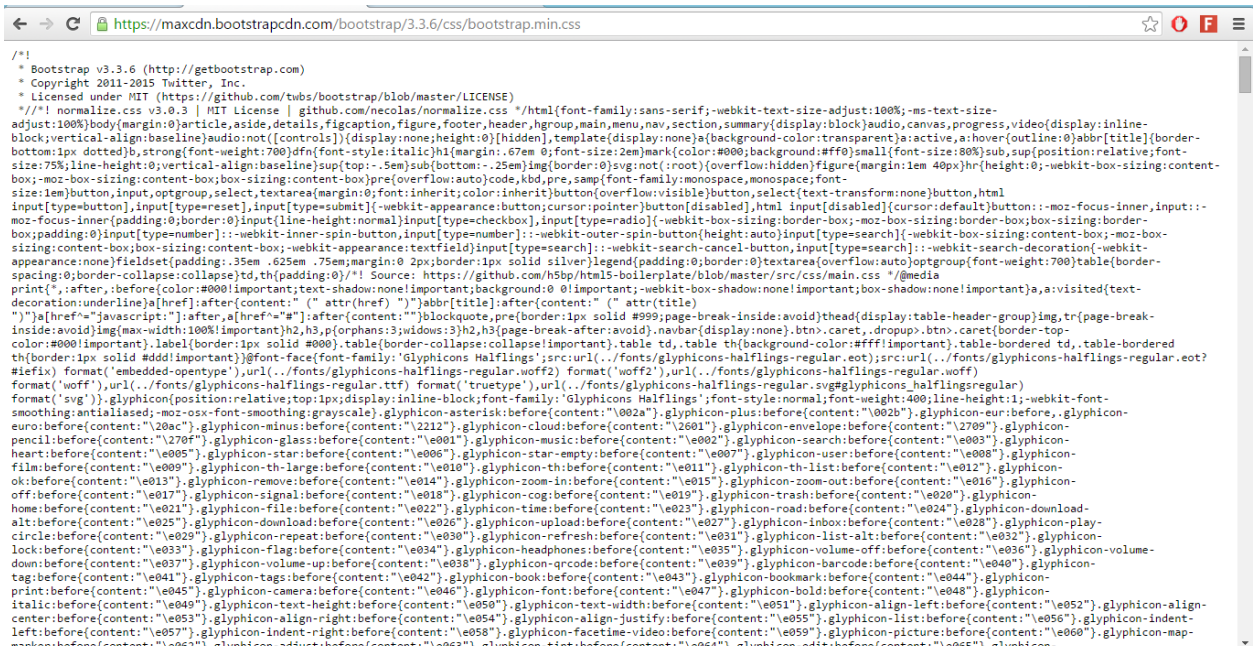
Create a "***styles.css***" file in public→stylesheets. Place the following content in it.

.modal-footer { border-top: 0px; }

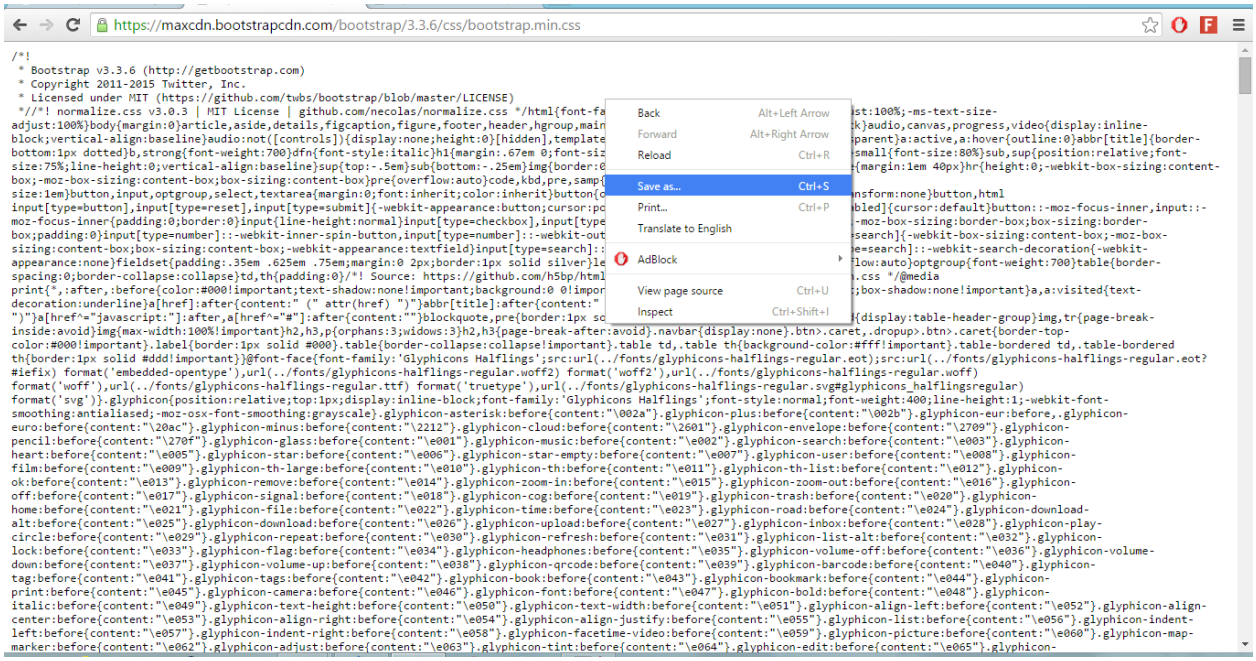
Go to this URL,

https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css

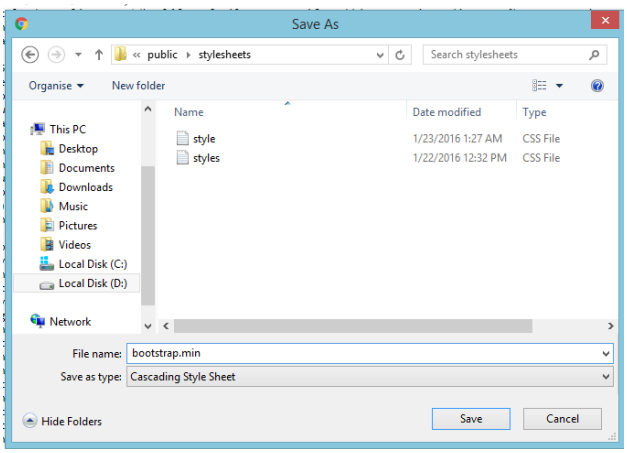
A page with bootstrap code appears.



Right click anywhere and select "Save As".



This will prompt you to select storage location. Go to your application → Public → stylesheets. When stylesheets folder is opened, click on “Save”.

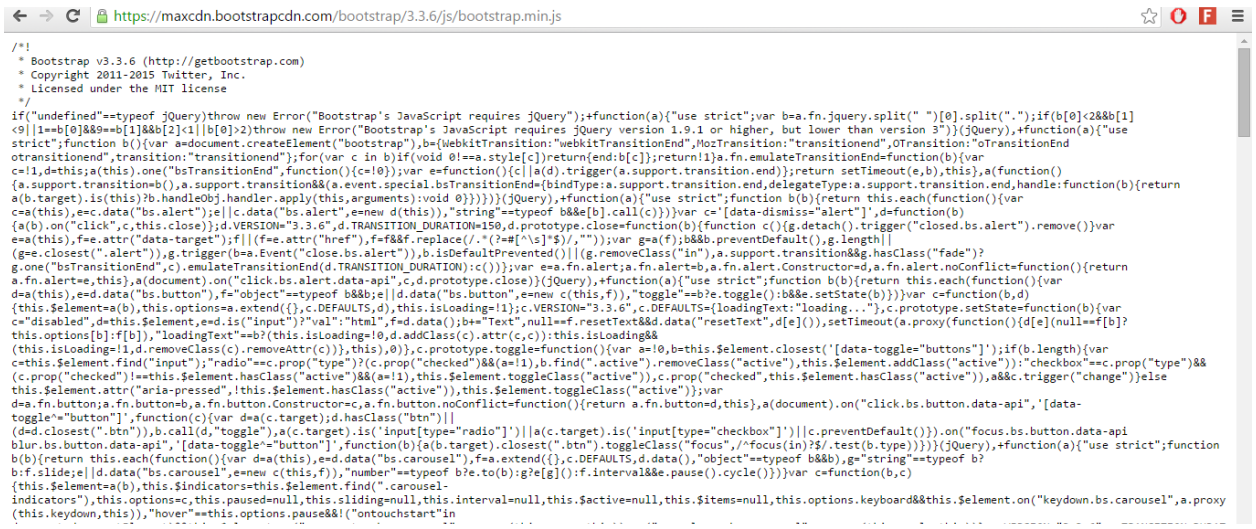


Go to your application → Public folder and create a “javascripts” folder.

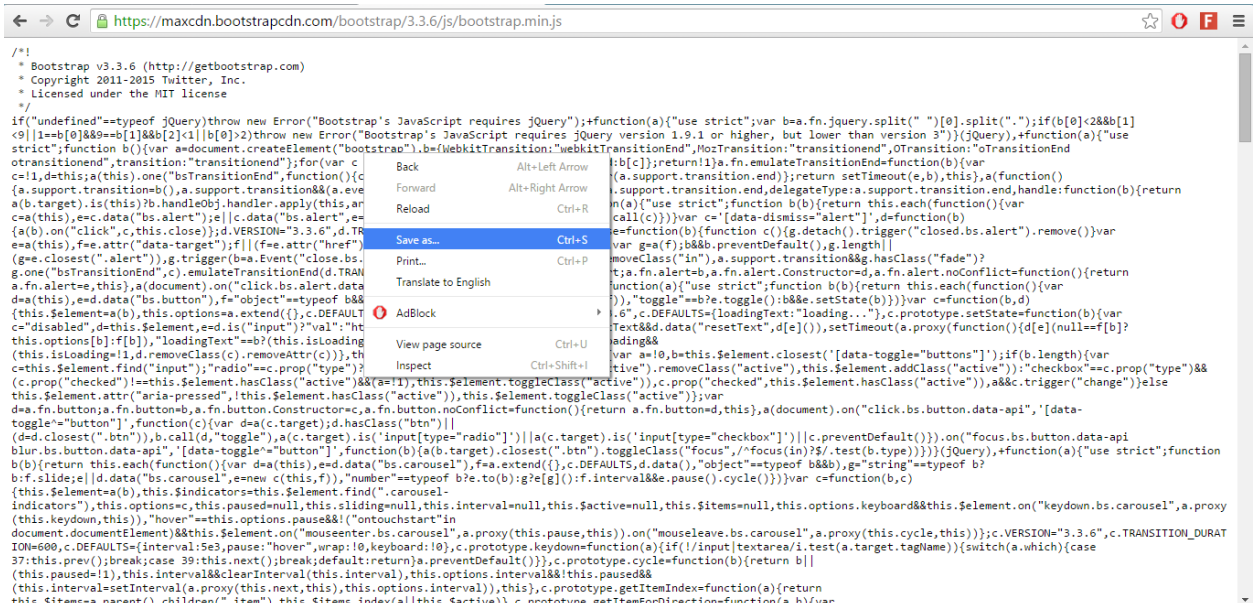
Go to this URL,

https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js

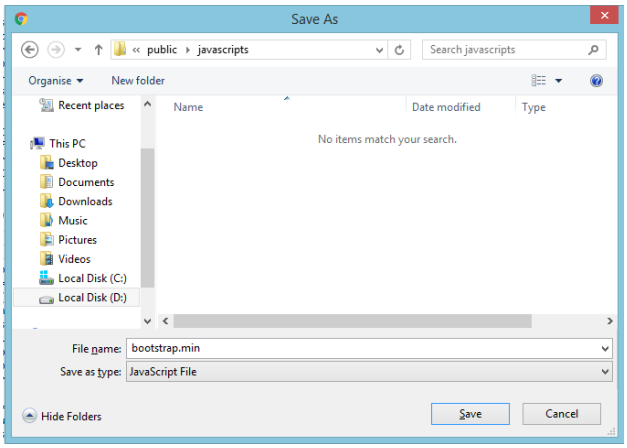
A page with bootstrap code appears.



Right click anywhere and select “Save As”.



This will prompt you to select storage location. Go to your application → Public → javascripts. When javascripts folder is opened, click on “Save”.



Now, create two folders with names as **“routes”** and **“views”** in the application which you downloaded from Bluemix.

Note: **Views** is used for designing Frontend**, Routes** is used for Controlling.

In **routes** folder, create two files - ”***index.js”*** and “***users.js***”

Open ***“index.js”*** file using an editor (Notepad++). Copy the following code, and paste it in “***index.js***” file

Also replace **<your-ibm-api-management-URL>** from this code with ther URL that was generated in IBM API Management Developer Portal.

```
var express = require('express');
var https=require('https');
var request = require('request-json');
var client = request.createClient('http://localhost:8888/');
var router = express.Router();
```

```
/* GET users listing. */
router.get('/', function(req, response, next) {
        var data = {
 'Fahrenheit':'25'
};
        client.post('<your-ibm-api-management-URL> ', data, function(err, res,
body) {
                response.render('index',{title:body.Celsius});

});

});
module.exports = router;
```

Note : Copy the Client ID and paste in the place of <your_client_id> in the above code.

Open "***users.js***" file using an editor (Notepad++). Copy the following code, and paste it in "***users.js***" file

Also repacle **<your-ibm-api-management-URL>** from this code with ther URL that was generated in IBM API Management Developer Portal.

```
var express = require('express');
var https=require('https');
var request = require('request-json');
var client = request.createClient('http://localhost:8888/');
var router = express.Router();

/* GET users listing. */
router.get('/', function(req, response, next) {
        var data = {
```

```
 'Fahrenheit':'25'
};
        client.post('<your-ibm-api-management-URL>', data, function(err, res,
body) {
                response.render('index',{title:body.Celsius});

});

});
module.exports = router;
```

Note : In the path, you should give the URL which was generated in IBM API Developer Portal. Also Copy the Client ID and paste in the place of <your_client_id> in the above code.

Open ***Views*** folder in application. Create a file – "index.ejs"

Open "***index.ejs***" file using an editor (Notepad++). Copy the following code, and paste it in "***index.ejs***" file

---

```
<html lang="en">
        <head>
                <meta http-equiv="content-type" content="text/html; charset=UTF-
8">
                <meta charset="utf-8">
                <title>Fahrenheit to Celsius Converter</title>
                <meta name="generator" content="Bootply" />
                <meta name="viewport" content="width=device-width, initial-
scale=1, maximum-scale=1">
                <link href="stylesheets/bootstrap.min.css" rel="stylesheet">
                <!--[if lt IE 9]>
                        <script
src="//html5shim.googlecode.com/svn/trunk/html5.js"></script>
```



```
                <![endif]-->
                <link href="stylesheets/styles.css" rel="stylesheet">
        </head>
        <body>
<!--login modal-->
 <form action="/users" method=get>
<div id="loginModal" class="modal show" tabindex="-1" role="dialog" aria-hidden="true">
  <div class="modal-dialog">
  <div class="modal-content">
    <div class="modal-header" style="background:black">

      <h2 class="text-center" style="color:white"><b>Miracle Software Systems</b></h2>
                <h4 class="text-center" style="color:#3498DB"><b>Fahrenheit to Celsius Converter</b></h4>
    </div>
    <div class="modal-body" style="background:#3498DB">
      <form class="form col-md-12 center-block">
        <div class="form-group" ><br><br>
          <center style="color:black"><h3><b>Celsius Result</b> : <%= title %></h3>
        </div>

      </form>
    </div>
    <div class="modal-footer" style="background:#3498DB">
      <div class="col-md-12">

                </div>
    </div>
  </div>
  </div>
</div>
</form>
```

```
        <!-- script references -->
                <script
src="//ajax.googleapis.com/ajax/libs/jquery/2.0.2/jquery.min.js"></script>
                <script src="js/bootstrap.min.js"></script>
        </body>
</html>
```

Note : This file will be used for displaying the output.

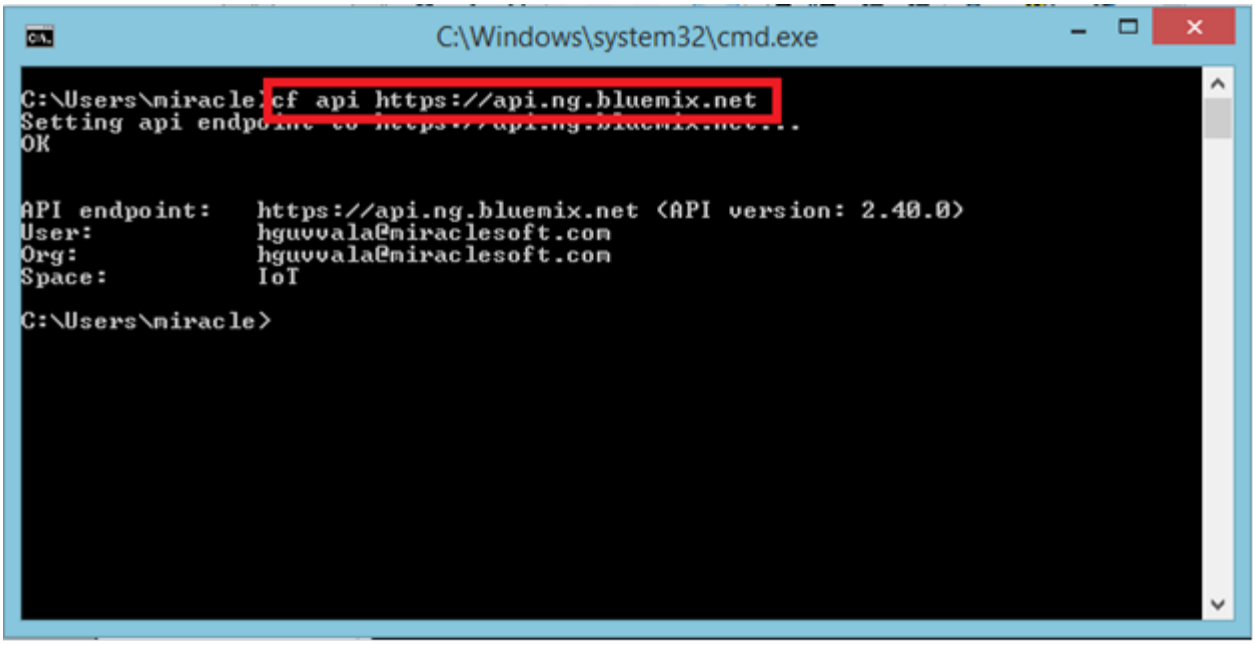
Now, you need to check whether the application is running fine or not.

#8 | Deploying to the Cloud with CF CLI

The next step will be to take your application and deploy it back to Bluemix so that you can share it with your friends.

Open to Command Prompt and go to the location where you have extracted your application. Then connect to Bluemix using one of the following commands (Depends on which region they have selected in your profile).

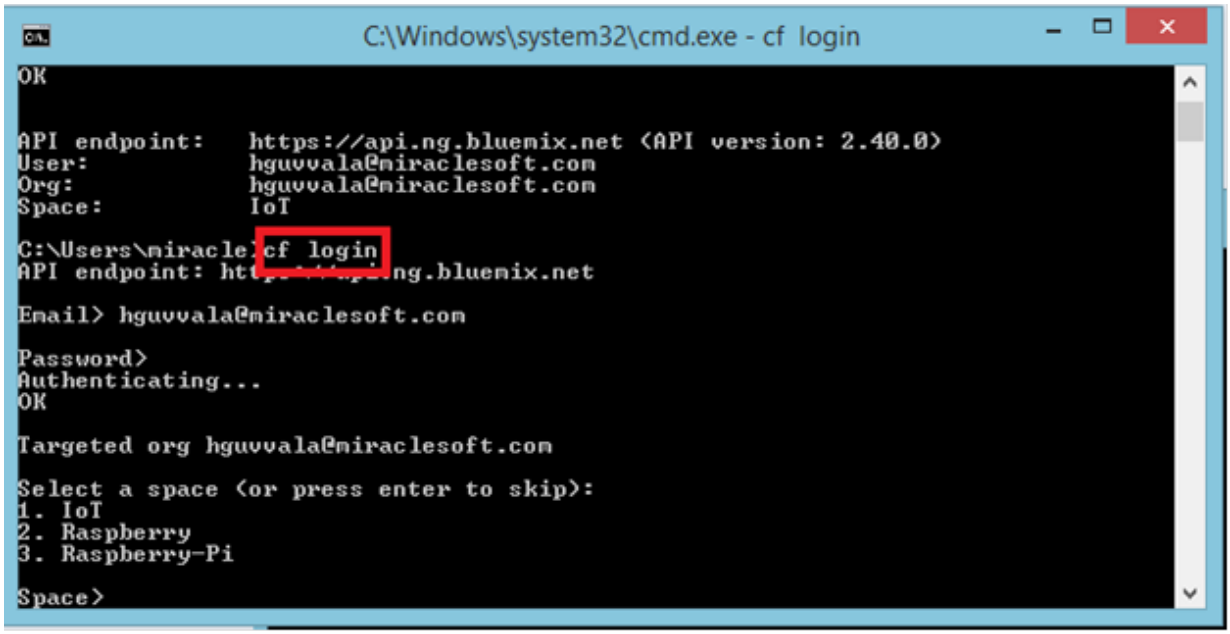
For Sydney : cf api https://api.au-syd.bluemix.net
For US South : cf api https://api.ng.bluemix.net
For US North : cf api https://api.eu.gb.bluemix.net



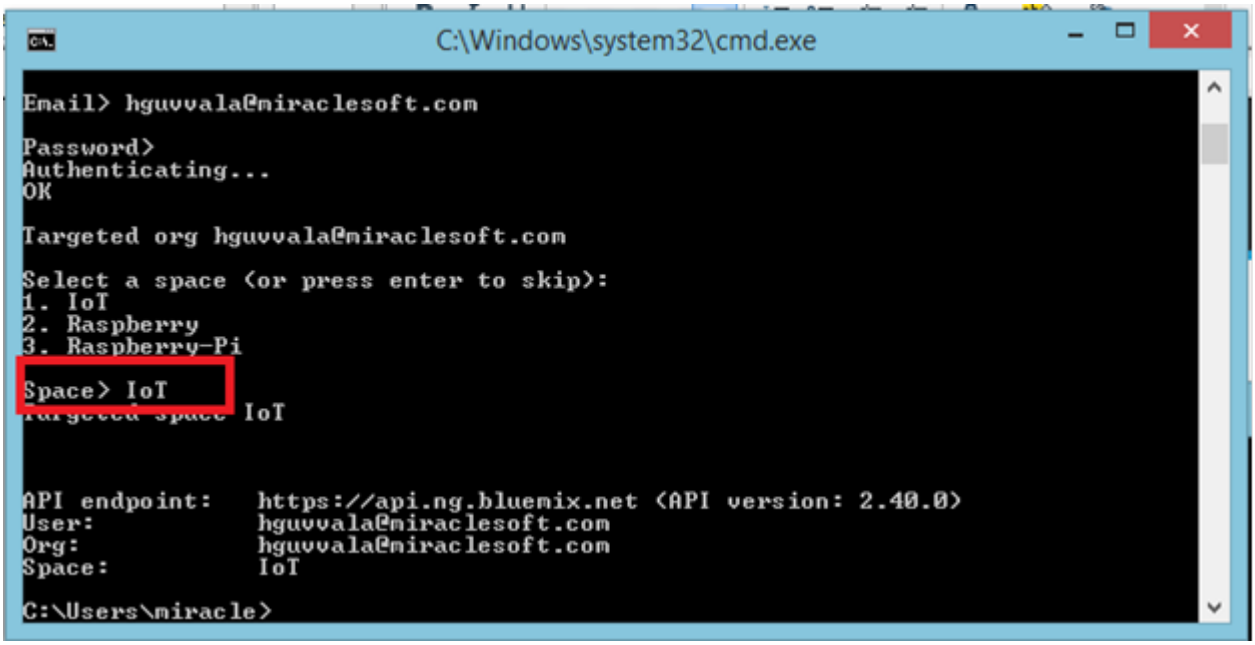
Login to Bluemix using the command,

cf login

When prompted enter username and password.

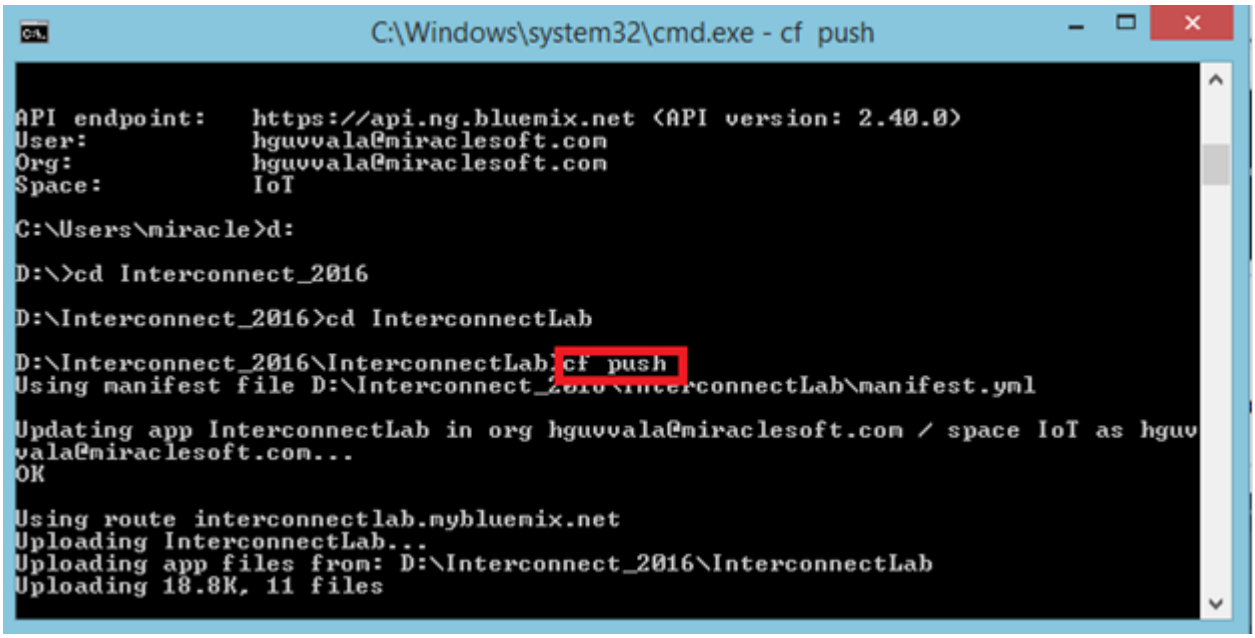


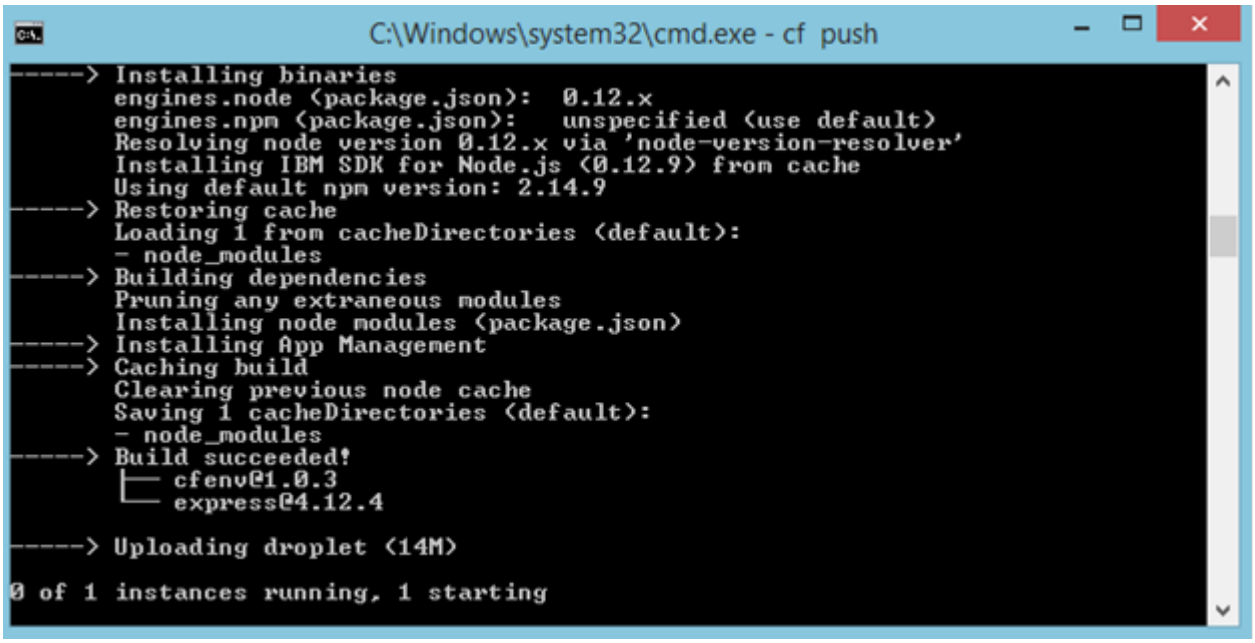
Specify the space where you deployed your application.



Make sure that you are in application's directory.

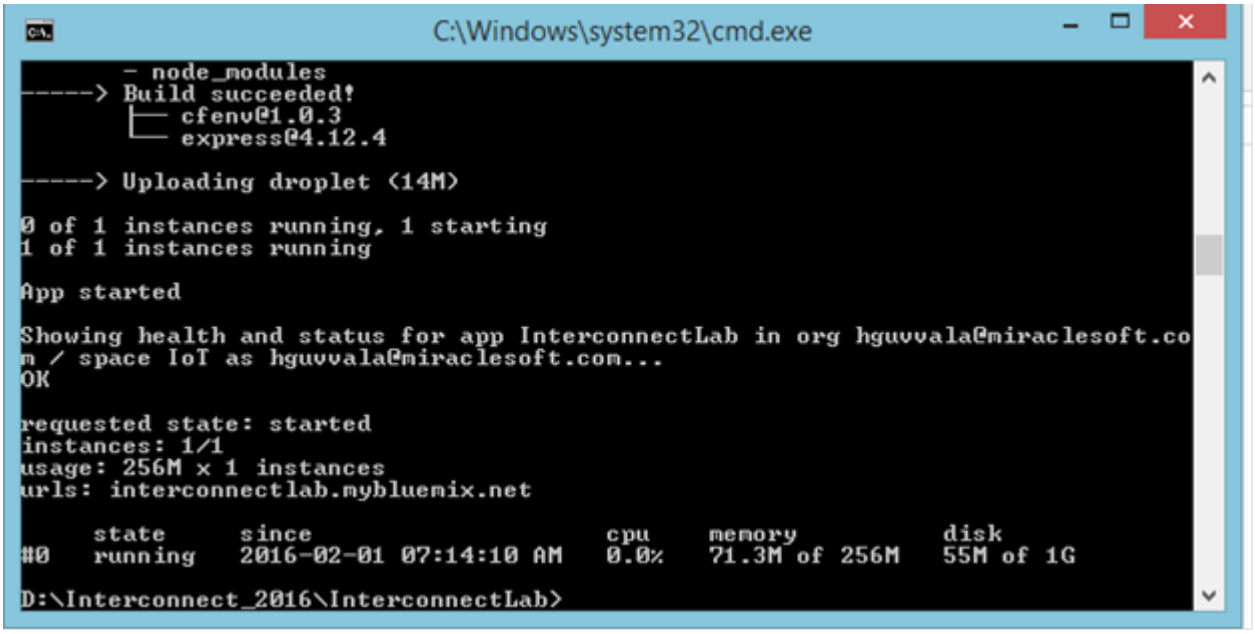
Use **cf push** to push your application to your Bluemix Organization.



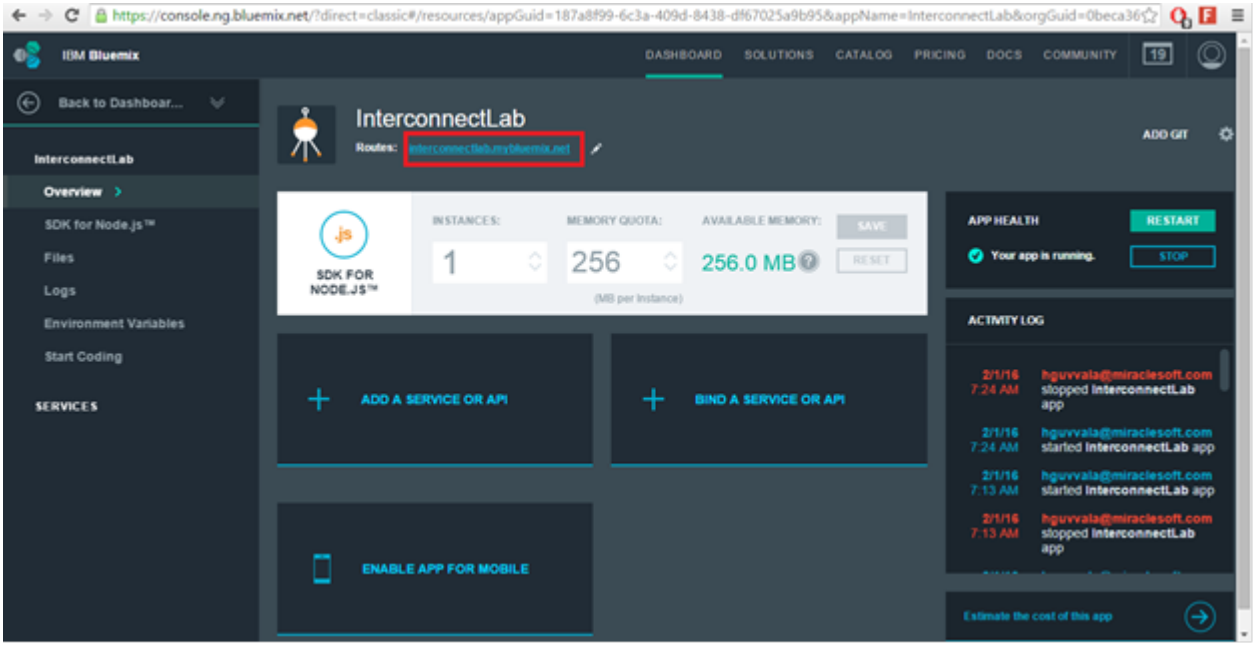


Note : This will take around 3 minutes for completion

Once your application is pushed, your Command prompt should look as below,



Now, you can go back to your Bluemix Dashboard in Browser and access your applications URL through **Dashboard--> Application Overview--> Application URL**.



The application, which you created and deployed in Bluemix should be as below.