



# 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 02<sup>nd</sup>, 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

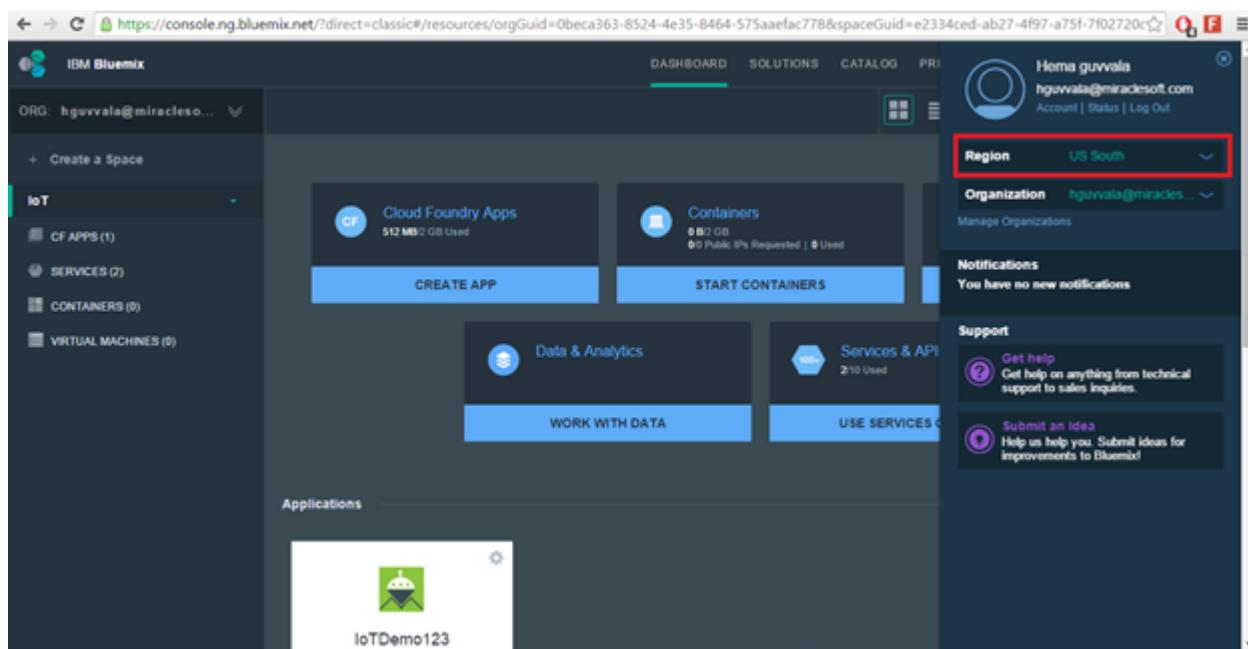
The first step will be to make sure that you have access to the IBM Bluemix Console with either the free trial option (or) the paid subscription option.

Login to Bluemix at <http://bluemix.net>

(or)

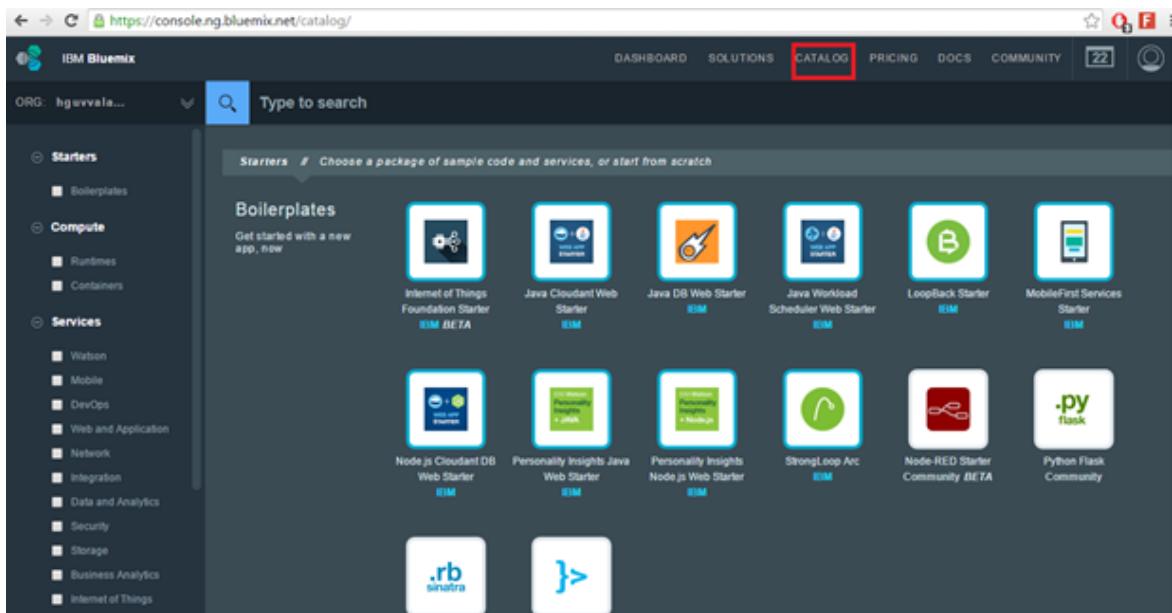
Register today at <https://console.ng.bluemix.net/registration/>

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.



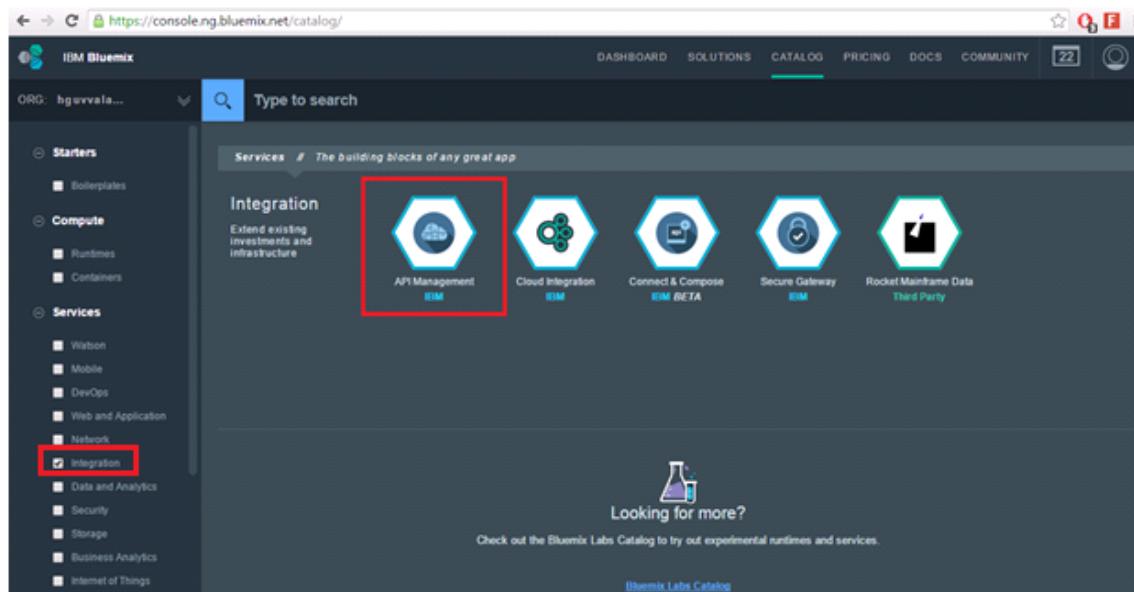
## #2 | Bind API Management Service

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



The screenshot shows the IBM Bluemix Catalog page. The top navigation bar includes links for DASHBOARD, SOLUTIONS, CATALOG (which is highlighted with a red border), PRICING, DOCS, and COMMUNITY. A search bar is located above the catalog grid. On the left, a sidebar menu is open under the 'Services' category, specifically the 'Integration' section, which is also highlighted with a red border. The main content area displays a grid of service icons and names. In the 'Boilerplates' section, the 'API Management' icon is visible. Other services shown include Internet of Things Foundation Starter, Java Cloudant Web Starter, Java DB Web Starter, Java Workload Scheduler Web Starter, LoopBack Starter, MobileFirst Services Starter, Node.js Cloudant DB Web Starter, Personality Insights Java Web Starter, Personality Insights Node.js Web Starter, StrongLoop Arc, Node-RED Starter Community BETA, and Python Flask Community.

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



This screenshot shows the same IBM Bluemix Catalog page as the previous one, but with a different selection in the sidebar. The 'Integration' option under the 'Services' category is now selected and highlighted with a red border. The main catalog grid has been updated to show services related to integration. The 'API Management' service is now highlighted with a red box. Other services listed in the 'Integration' section include Cloud Integration, Connect & Compose, Secure Gateway, and Rocket Mainframe Data. The sidebar on the left remains the same, showing the expanded 'Integration' section.

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

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a card for the 'API Management' service, which includes details like Publish Date (11/17/2015), Author (IBM), Type (Service), and Location (US South). Below this card is a 'VIEW DOCS' button. The main area displays two sections: 'Discover Existing APIs' and 'Design New APIs'. The 'Discover Existing APIs' section shows a graph of API usage over time and a list of available values. The 'Design New APIs' section shows a diagram of a transformation process. To the right, there's a form for adding a new service. The 'Space' dropdown is highlighted with a red box and set to 'iot'. The 'Service name' field contains 'API Management-yg'. The 'Selected Plan' dropdown is set to 'Standard v2'. At the bottom right of the form is a large green 'CREATE' button.

Click on “Go To API Manager”.

The screenshot shows the 'API Management-yg' dashboard. On the left sidebar, there are links for 'Manage' and 'Service Access Authorization'. The main content area has a heading 'Get started with API Management' with the sub-instruction 'You can design, publish, and manage APIs through the API Manager console.' Below this is a green bar with the text 'All set! Use the links below to get started..'. There are four circular icons with corresponding text: 'Import APIs, or compose a new one.', 'Create a new plan, add resources and rate limits, and deploy.', 'Invite other Bluemix organizations to use your APIs.', and 'Publish your plan.'. At the bottom of the page, it says 'Application developers will be able to discover and consume your APIs from the Bluemix catalog.' Below this are two buttons: a large green 'GO TO API MANAGER' button and a smaller 'LEARN MORE' button.

## #3 | Create API

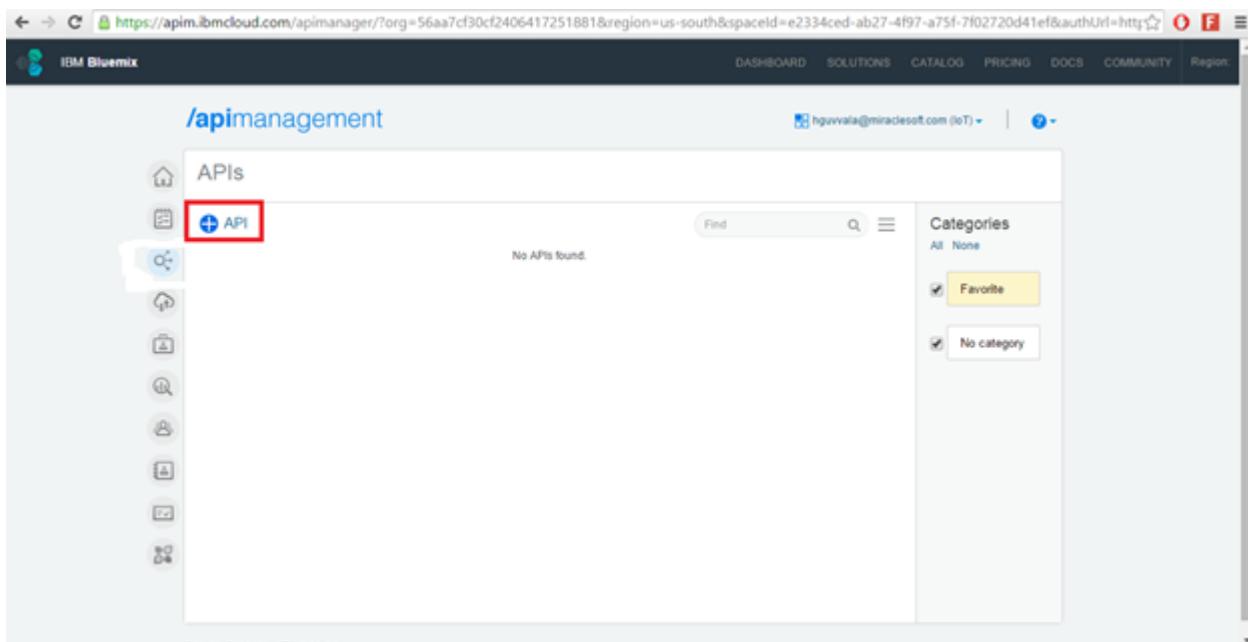
It will be redirecting into **API Manager Console**.

The screenshot shows the IBM Bluemix API Management console at the URL <https://apim.ibmcloud.com/apimanager/>. The top navigation bar includes links for DASHBOARD, SOLUTIONS, CATALOG, PRICING, DOCS, and COMMUNITY, along with a Region selector. The main content area is titled '/apimanagement' and features a 'Home' section with various icons and statistics. A sidebar on the left contains navigation links for Approvals, Requests, Plans, and Applications. On the right, there are sections for Usage (API Usage, Developers, Basic Portal, Advanced Portal), and Payload Logging.

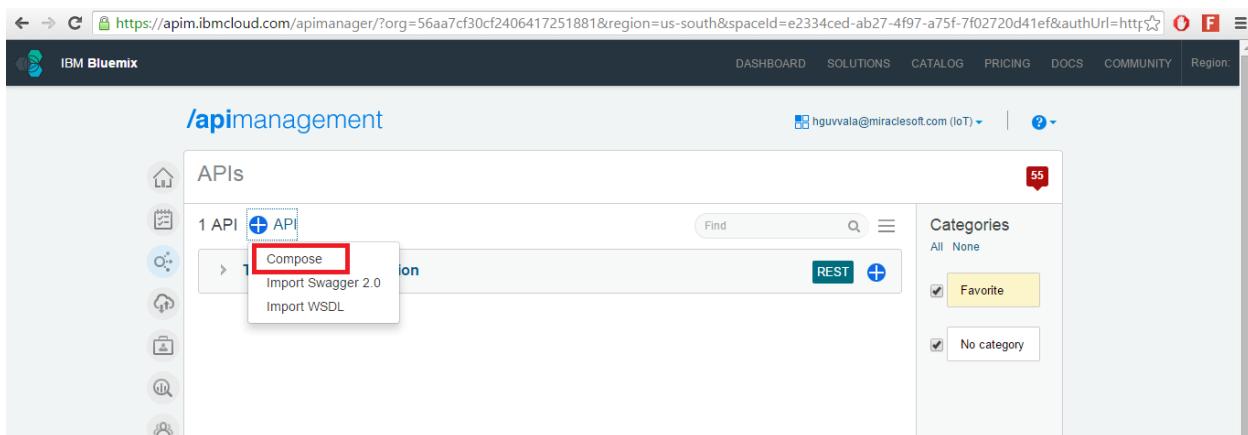
Create API by clicking on API's icon in Navigation Pane, this will take you to Draft API's view.

The screenshot shows the IBM Bluemix API Management console at the URL <https://apim.ibmcloud.com/apimanager/>. The top navigation bar includes links for DASHBOARD, SOLUTIONS, CATALOG, PRICING, DOCS, and COMMUNITY, along with a Region selector. The main content area is titled '/apimanagement' and features a 'APIs' section with a search icon highlighted with a red box. The search bar shows the placeholder 'Find' and the text 'No APIs found.' To the right, there is a 'Categories' section with options for 'All' and 'None', and buttons for 'Favorite' and 'No category'.

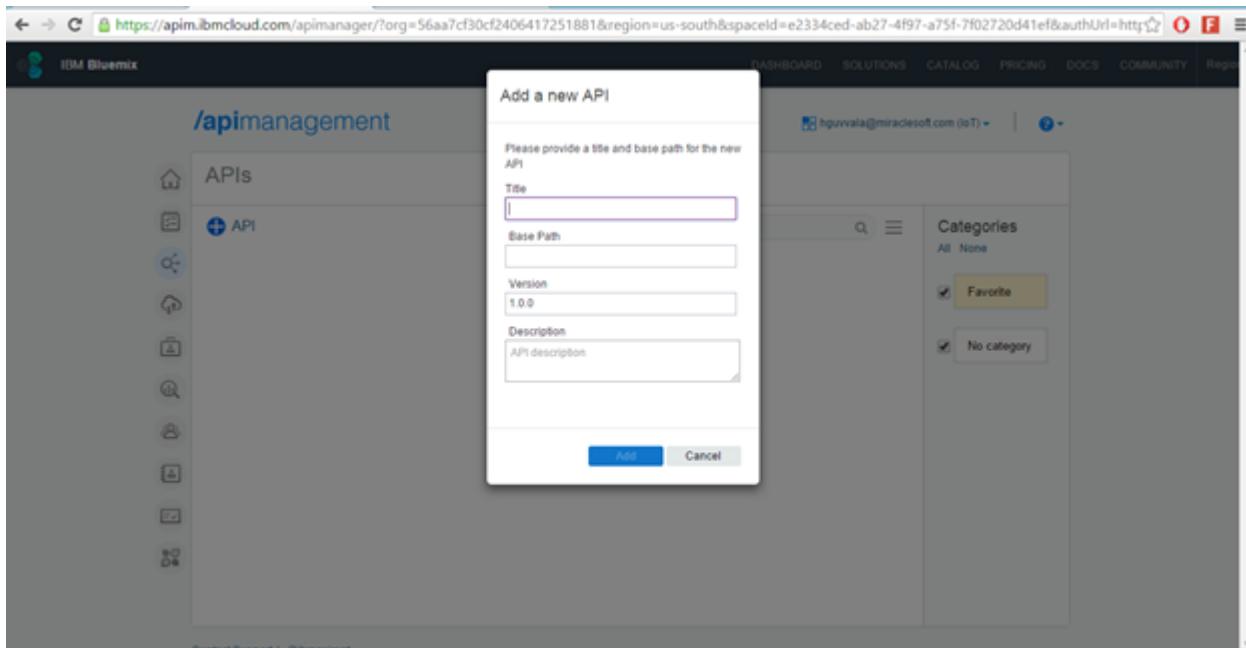
Click on “+API” Button to create New REST API.



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



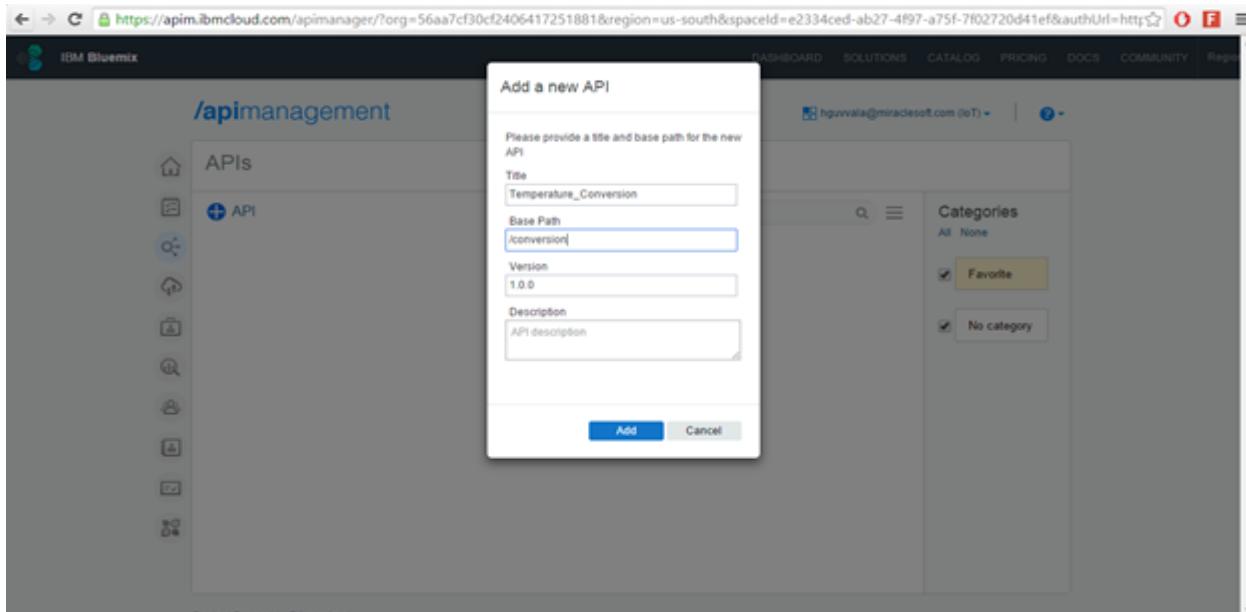
Fill the fields as shown below. Click the **Add button**. This will add the API to the Draft APIs list.



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

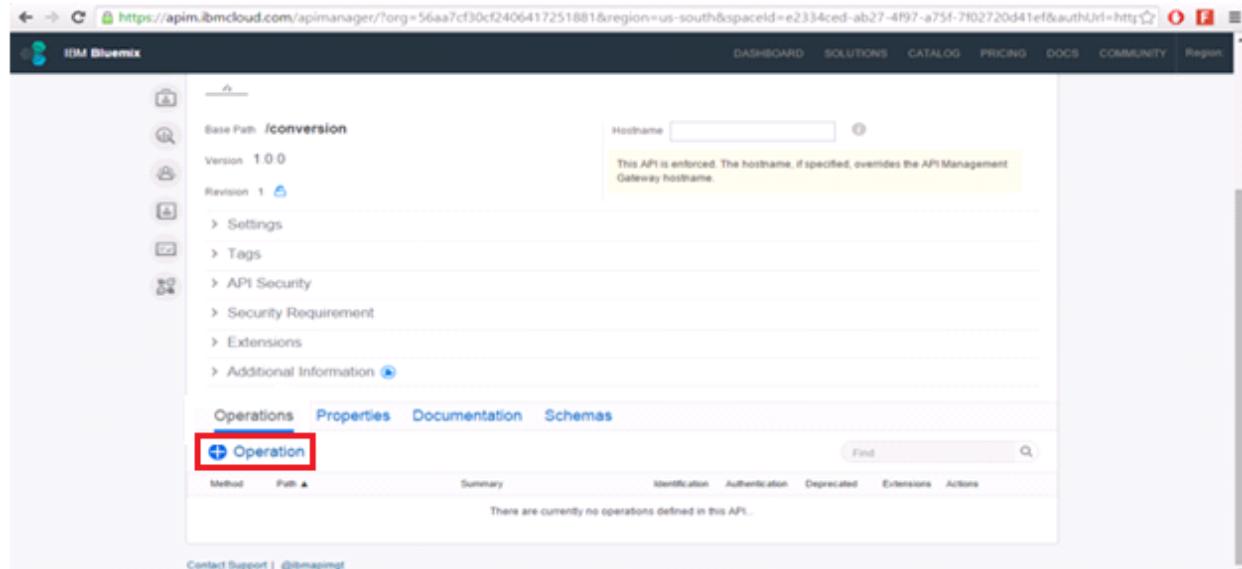
Field Name	Value
API Name(Title)	Temperature_Conversion
Base Path	/conversion
API Description	Converting the temperature from Celsius to Fahrenheit

**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.

The screenshot shows the 'Operations' tab selected in the top navigation bar. Under the 'Operation' heading, there's a 'Path' input field containing 'e.g. /collection/{id}?filter'. Below it, a 'Methods' section lists 'GET X' and 'POST X'. A red box highlights the 'GET X' button. At the bottom, there are tabs for 'Method', 'Path ▲', 'Summary', 'Identification', 'Authentication', 'Deprecated', 'Extensions', and 'Actions'.

Populate the fields of the Operation for that API.

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

The screenshot shows the 'Operations' tab selected in the top navigation bar. Under the 'Operation' heading, there's a 'Path' input field containing '/FahrenheitToCelsius', a 'Summary (optional)' input field containing 'TempConv', and a 'Description (optional)' input field containing 'Temperature Conversion Resource'. To the right of these fields are 'Identification' and 'Authentication' checkboxes, with 'Identification' checked and 'Authentication' unchecked. A red box highlights the 'Add' button at the bottom right. Below the input fields are tabs for 'Method', 'Path ▲', 'Summary', 'Identification', 'Authentication', 'Deprecated', 'Extensions', and 'Actions'.

The screenshot shows the IBM Bluemix API Manager interface. On the left, there's a sidebar with icons for Home, Overview, Catalog, and API Management. The main area displays a resource named '/conversion'. The resource details are as follows:

- Base Path:** /conversion
- Version:** 1.0.0
- Revision:** 1
- Operations:** POST /FahrenheitToCelsius
- Properties:** Identification, Authentication, Deprecated, Extensions, Actions.

A yellow callout box on the right states: "This API is enforced. The hostname, if specified, overrides the API Management Gateway hostname." A red box highlights the 'Edit' icon (pencil) next to the 'Actions' column for the POST operation.

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

This screenshot is identical to the one above, showing the same resource details for '/conversion'. However, a red box now highlights the 'Edit' icon (pencil) located in the 'Actions' column of the table for the POST operation, indicating where the user should click 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.

The screenshot shows the IBM Bluemix API Manager interface. A POST operation named /FahrenheitToCelsius is selected. The Request tab is active. In the Request Headers section, there is one entry: a header named 'Header' with the value 'Content-Type: application/json'. The Swagger Schema Object section is currently empty. The Request Example section contains the following JSON object:

```
[{"FahrenheitTemperature": "25"}]
```

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

The screenshot shows the IBM Bluemix API Manager interface. A POST operation named /FahrenheitToCelsius is selected. The Request tab is active. In the Request Headers section, there is one entry: a header named 'Header' with the value 'Content-Type: application/json'. The Swagger Schema Object section is currently empty. The Request Example section contains the following JSON object:

```
[{"FahrenheitTemperature": "25"}]
```

Also Add Response Headers, the request type is SOAP, but we would like to use JSON as the data format.

The screenshot shows the IBM Bluemix API Manager interface. A new operation is being created with the following details:

- Method:** POST
- Path:** /FahrenheitToCelsius
- Description:** Temperature Conversion
- Response:** 200

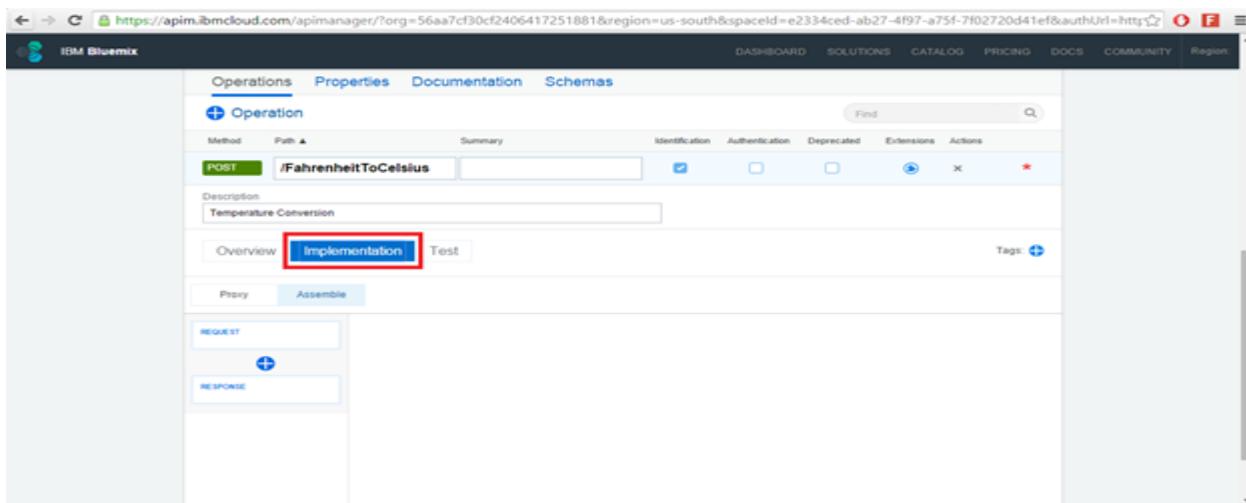
The "Response" tab is selected. The response body contains the value "200".

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

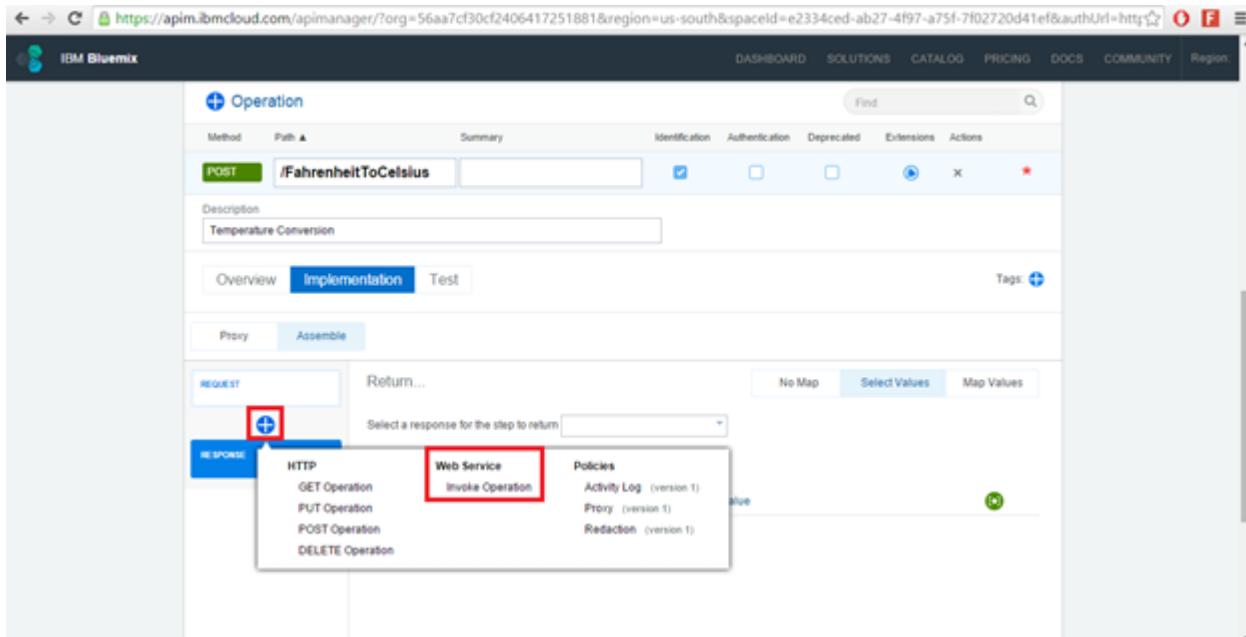
The screenshot shows the configuration of a response header for the "200" status code. The "Header" section is expanded, showing the following details:

- Header Name:** CelsiusTemperature
- Swagger Schema Object:** An empty JSON object ({}).
- Response Example:** {"CelsiusTemperature": "65"}

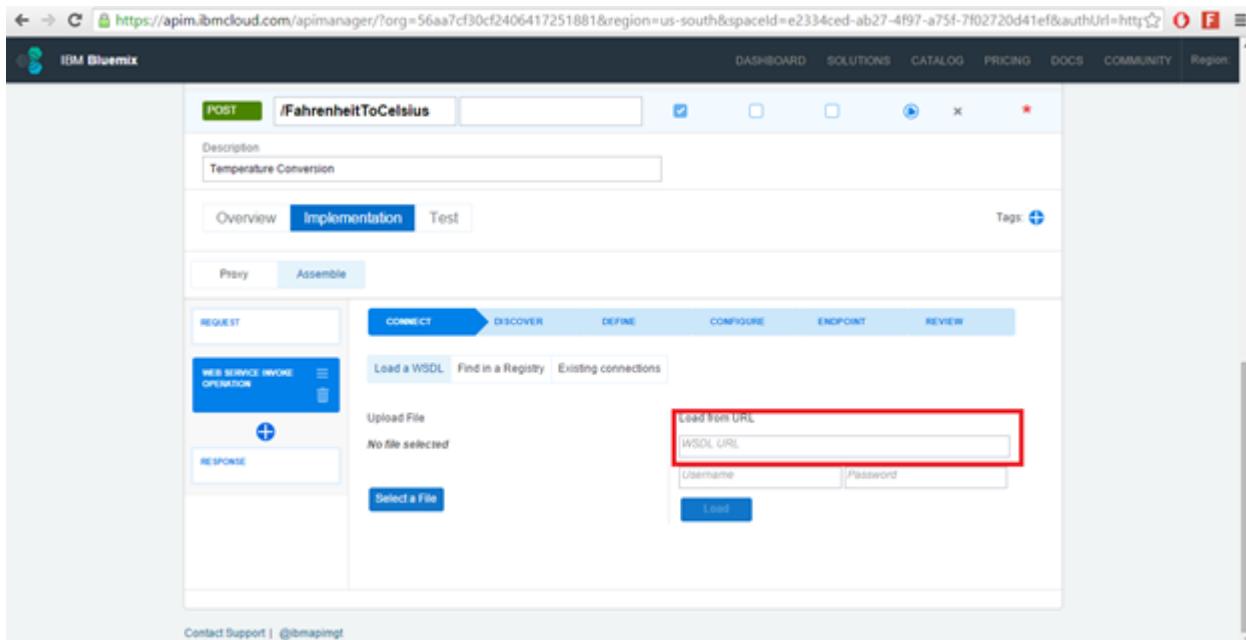
Switch back to API Manager Console, and click on **Implementation** tab for invoking the web Service.



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	<a href="http://webservice.daehosting.com/services/TemperatureConversions.wso?WSDL">http://webservice.daehosting.com/services/TemperatureConversions.wso?WSDL</a>
Username	Leave blank
Password	Leave blank

[Overview](#)
[Implementation](#)
[Test](#)

Tags: +

[Proxy](#)
[Assemble](#)

REQUEST
CONNECT
DISCOVER
DEFINE
CONFIGURE
ENDPOINT
REVIEW

[Load a WSDL](#)
[Find in a Registry](#)
[Existing connections](#)

Upload File

No file selected

Select a File

Load from URL

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.

[Overview](#)
[Implementation](#)
[Test](#)

Tags: +

[Proxy](#)
[Assemble](#)

REQUEST
CONNECT
DISCOVER
DEFINE
CONFIGURE
ENDPOINT
REVIEW

[Available Objects](#)
[Available Fields](#)

[Select All](#)
[Clear All](#)

Use	Field	Type	Sample Data
<input type="checkbox"/>	body	object	...
<input type="checkbox"/>	FahrenheitToCelcius	object	...
<input checked="" type="checkbox"/>	nFahrenheit	number	...
<input type="checkbox"/>	header	object	...
<input type="checkbox"/>	headers	object	...
<input type="checkbox"/>	Security	object	...
<input type="checkbox"/>	UsernameToken	object	...
<input type="checkbox"/>	Username	string	...
<input type="checkbox"/>	Password	string	...

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. Here you can see input parameters of your API and the input variables of the Web service are mapped automatically.

Overview    **Implementation**    Test    Tags: +

Proxy    Assemble

REQUEST    CONNECT    DISCOVER    DEFINE    **CONFIGURE**    ENDPOINT    REVIEW

WEB SERVICE INVOKE OPERATION

RESPONSE

Available values		Transformation	Input variables	
Field	Sample		Field	Sample
* Request			* Request	
* body			* body	
* JSON			* FahrenheitToCelcius	
xy			4.5 $\times$ nFahrenheit	
EnterTemperatureInFahrenheit	65	(Yellow line with a plus sign)		

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.

The screenshot shows the Implementation tab of the Miracle Software Systems interface. On the left, there's a sidebar with tabs for Overview, Implementation (which is selected), and Test. To the right of the sidebar, there are two main sections: REQUEST and RESPONSE. The REQUEST section contains a 'WEB SERVICE INVOKE OPERATION' step. The RESPONSE section contains several output fields: Response, body, FahrenheitToCelsiusResponse, FahrenheitToCelsiusResult, header, and headers. At the top of the main area, there are five buttons: CONNECT, DISCOVER, DEFINE, CONFIGURE, and ENDPOINT, followed by a large blue 'REVIEW' button which is highlighted with a red box.

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

This screenshot shows the Implementation tab of the IBM Bluemix API Manager interface. It has a similar structure to the Miracle interface, with REQUEST and RESPONSE sections. The REQUEST section includes a 'WEB SERVICE INVOKE OPERATION' step. The RESPONSE section includes input fields for Status, body, JSON, and Headers. The 'Implementation' tab is selected at the top. The URL in the browser is https://apim.ibmcloud.com/apimanager/?org=56aa7cf30cf2406417251881&region=us-south&spaceId=e2334ced-ab27-4f97-a75f-7f02720d41ef&authUrl=https://apim.ibmcloud.com/authn

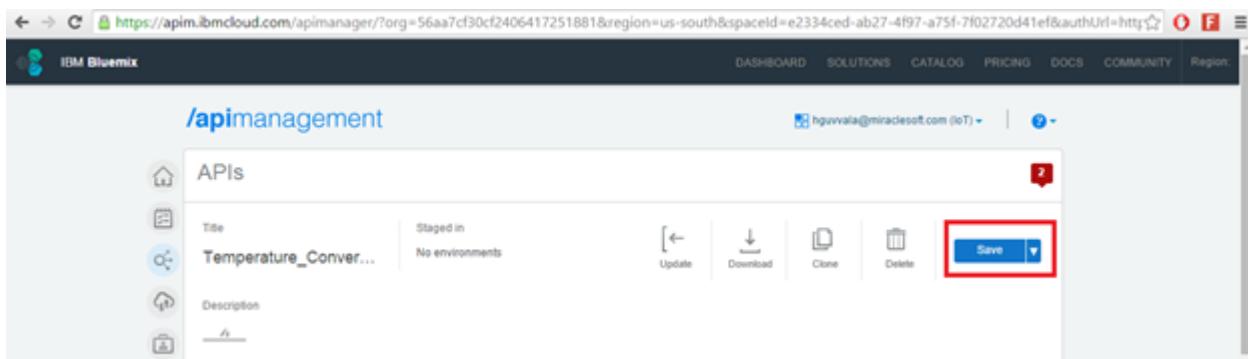
Select **Map Values** to see the alternative way to create mappings using the graphical mapping tool.

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.

The screenshot shows the MIRACLE software interface for mapping API responses. The top navigation bar has tabs for 'Proxy' and 'Assemble'. The 'Assemble' tab is selected. Below the tabs, there's a 'REQUEST' section and a 'WEB SERVICE INVOKE OPERATION' section. A red box highlights the 'RESPONSE' section. In the 'RESPONSE' section, there's a dropdown menu set to '200'. To the right of the dropdown, there are three buttons: 'No Map', 'Select Values', and 'Map Values', with 'Map Values' being highlighted by a red box. Below these buttons is a table with three columns: 'Available values', 'Transformation', and 'Input variables'. The 'Available values' column lists various fields like 'Status', 'body', 'JSON', 'FahrenheitToCelsiusResult', and 'Headers'. The 'Transformation' column contains a yellow mapping diagram showing a curved line connecting a source field to a target field. The 'Input variables' column shows the current values for each field. The entire mapping diagram area is also highlighted with a large red box.

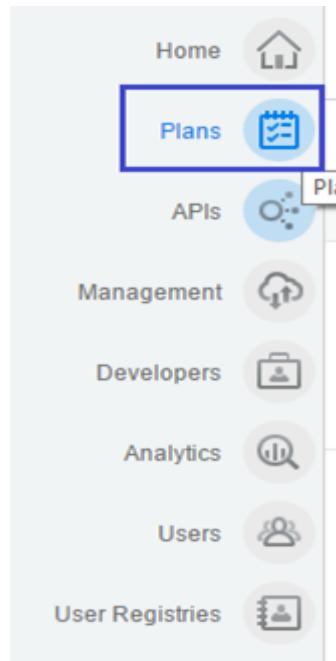
Click **Save** button at the top of the API editor.



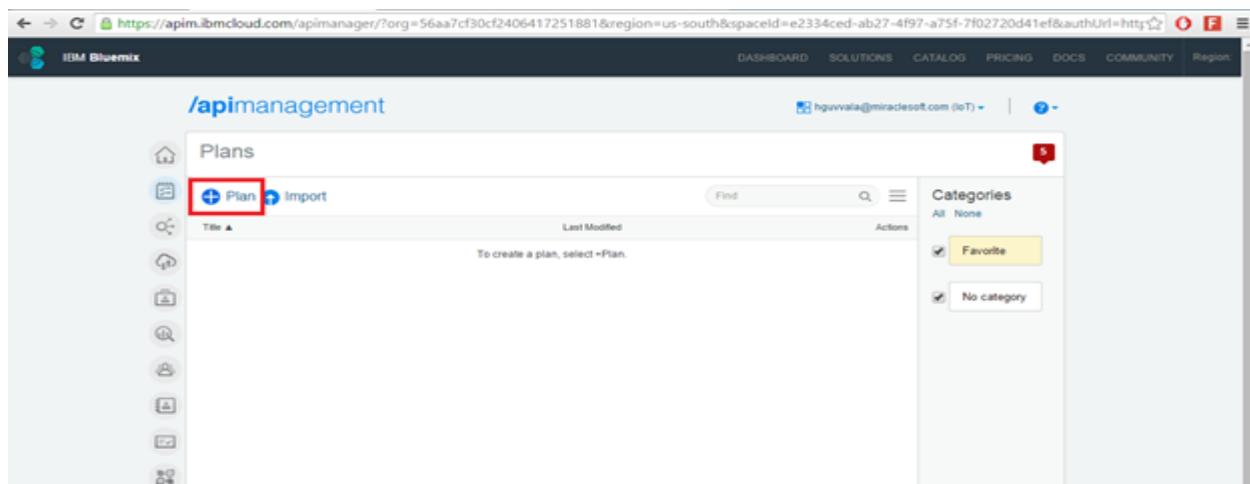
You have now defined your **CurrencyConverter\_Mapping** REST API. Before you can test the resource from within the API Manager UI you will need to add it to a Plan.

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

Select **Plans** from the Navigation Pane.

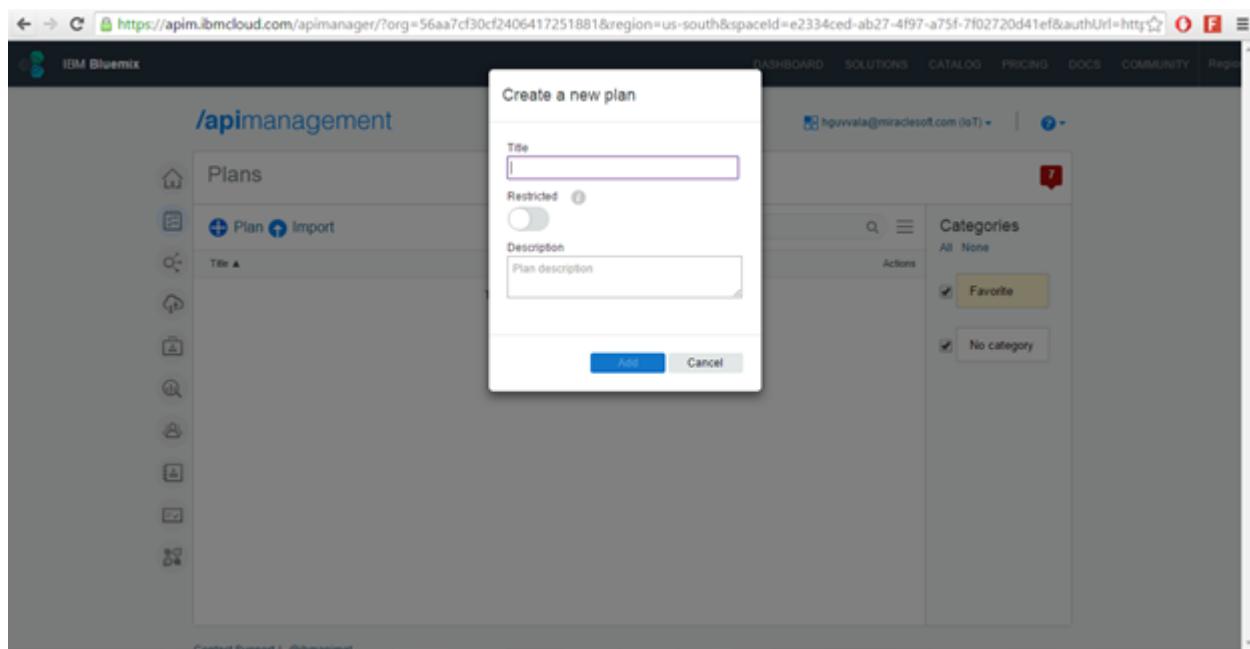


Click on **+Plan** for creating Plan.

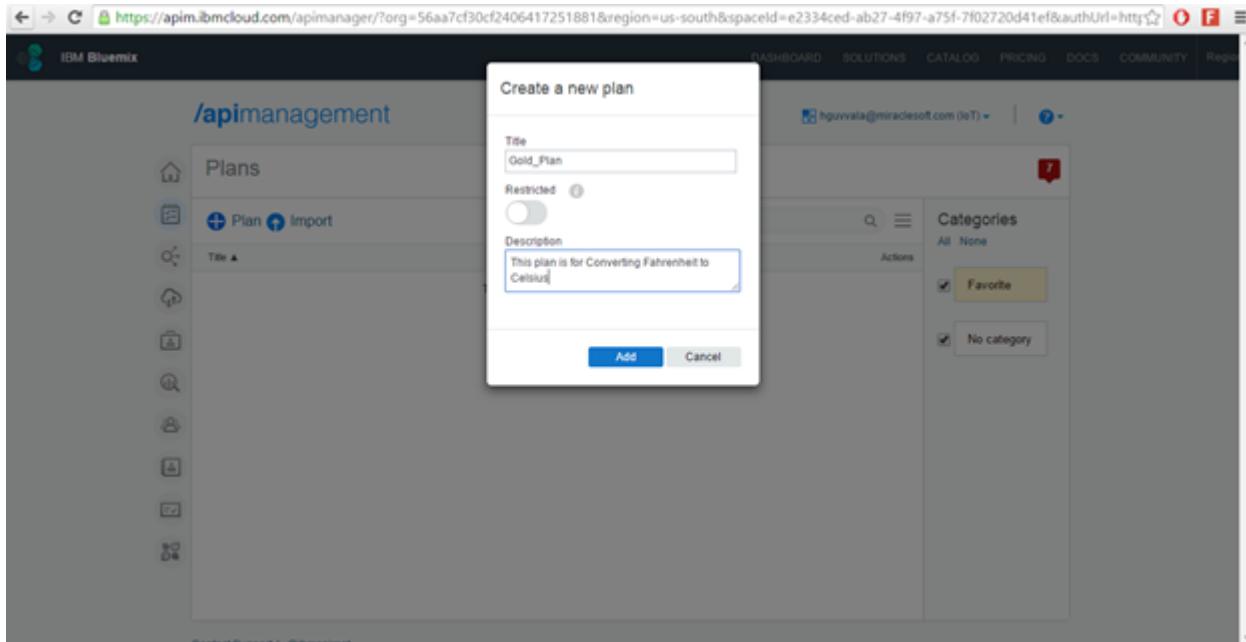


The screenshot shows the 'Plans' section of the API management interface. On the left, there's a sidebar with various icons. In the center, a table lists 'Plans'. At the top of the table, there's a button labeled '+ Plan' with a plus sign icon, which is highlighted with a red box. To the right of the table, there are sections for 'Categories' (with 'Favorite' and 'No category' options) and 'Actions' (with a search bar and a refresh icon).

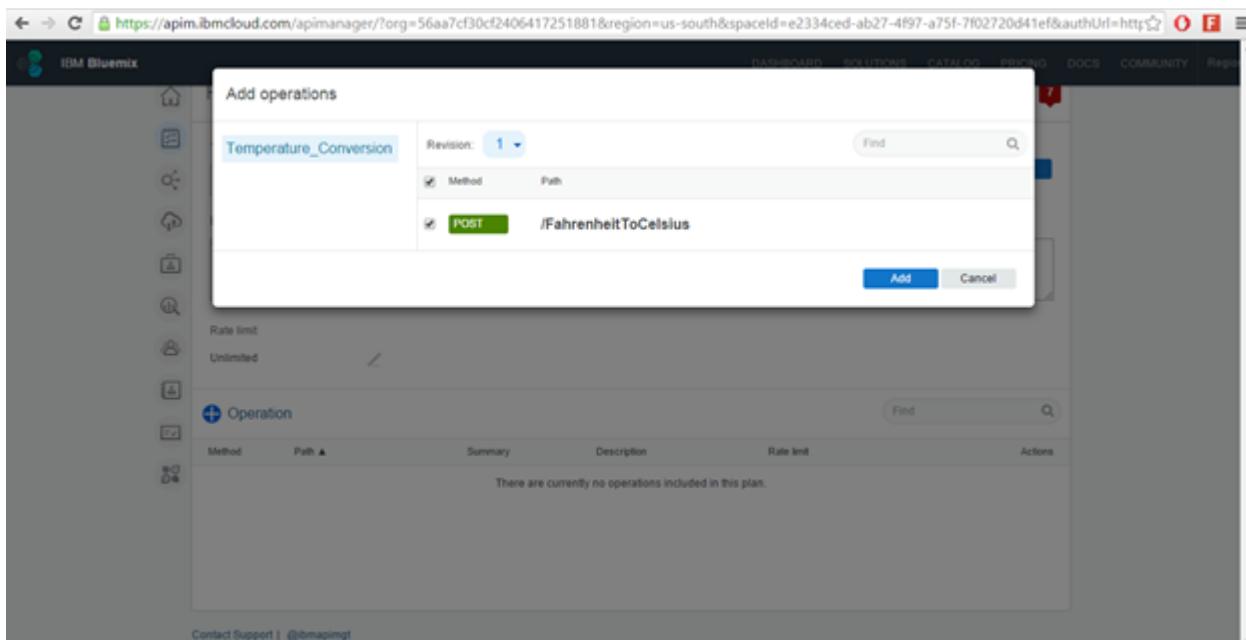
Specify the Plan name and Description for creating the plan.



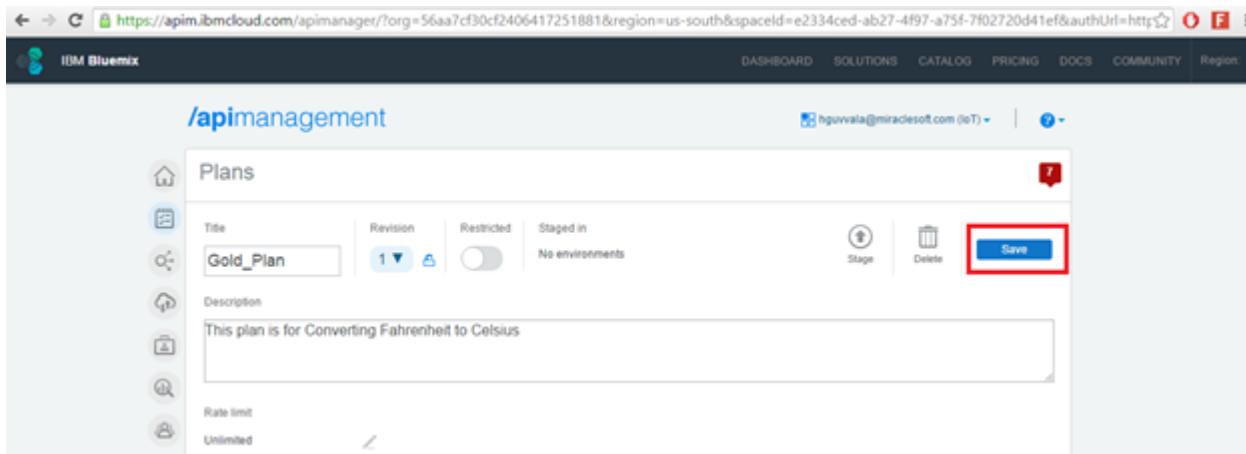
The screenshot shows a modal dialog titled 'Create a new plan'. It has fields for 'Title' (an empty input box), 'Restricted' (a toggle switch that is off), and 'Description' (a text box containing 'Plan description'). At the bottom of the dialog are two buttons: 'Add' (which is highlighted with a blue box) and 'Cancel'.



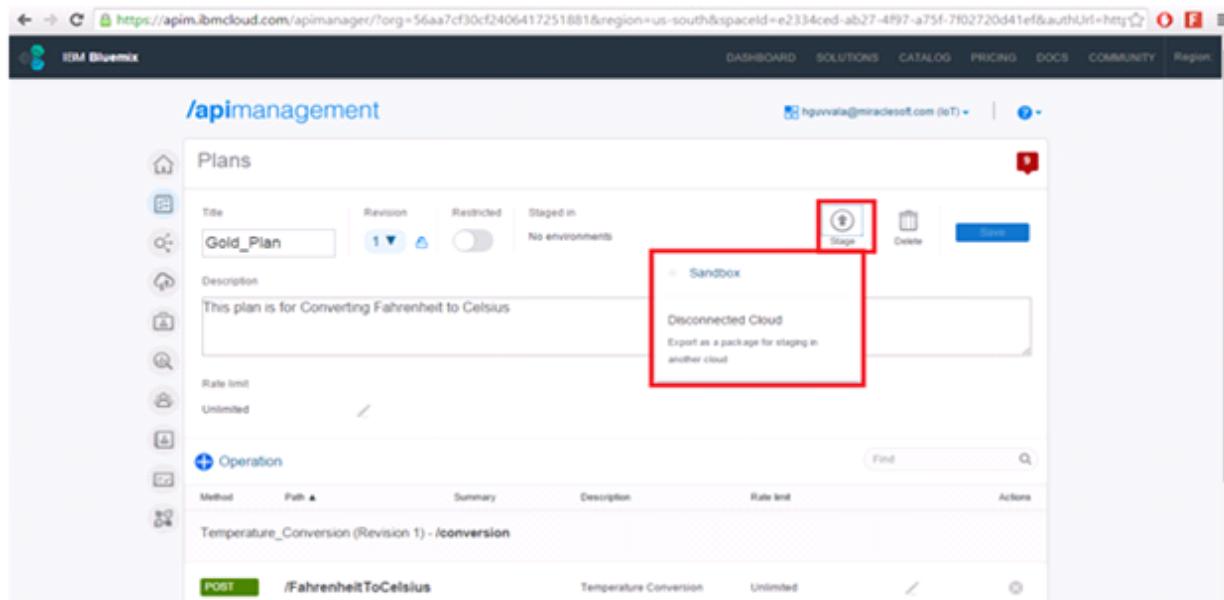
In order to expose the resource you have created you need to add it to the Public plan. Click **+ Operation**.



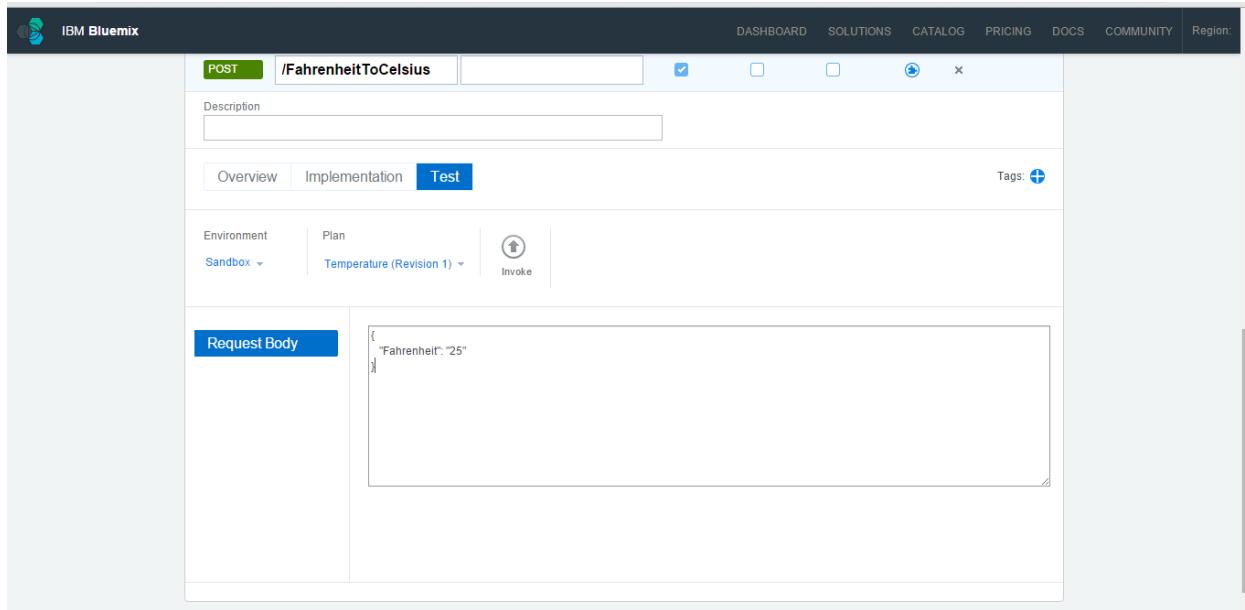
Click on Save to save the Plan.



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

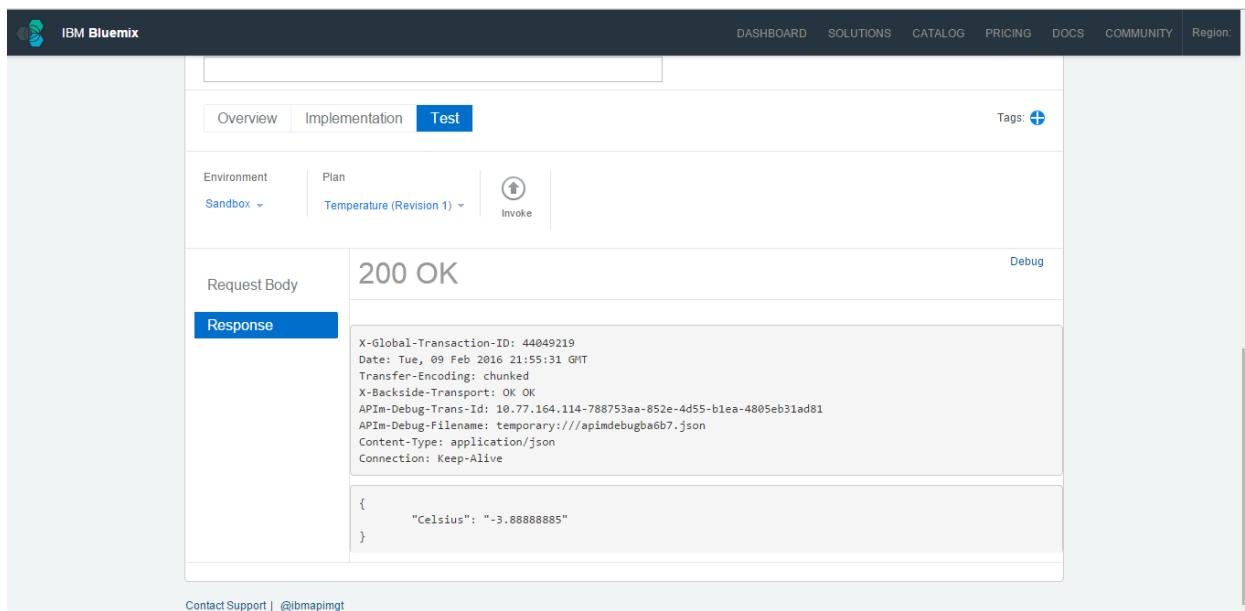


To test your REST resource to ensure that it is defined and implemented correctly, Select APIs from the navigation pane. Click the **Edit icon** for the POST quote resource. Now, Test your API.



The screenshot shows the IBM Bluemix API Test interface. At the top, there's a header with the IBM Bluemix logo and navigation links: DASHBOARD, SOLUTIONS, CATALOG, PRICING, DOCS, COMMUNITY, and Region. Below the header, a card displays a POST request to the endpoint /FahrenheitToCelsius. The Request Body contains the JSON payload: { "Fahrenheit": "25" }. The Test tab is selected in the navigation bar below the card.

Click on Invoke to invoke your API.



The screenshot shows the IBM Bluemix API Test interface after invoking the API. The status is 200 OK. The Response section shows the detailed HTTP headers and the JSON response body: { "Celsius": "-3.88888885" }.

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

The screenshot shows the IBM Bluemix API Management interface. On the left is a sidebar with icons for Home, Environment, API, Data, Functions, Pipelines, and Watson. The main area is titled '/apimanagement' and shows an 'Environments' section. A specific environment named 'Sandbox APIMOMT\_GATEWAY' is selected. Below it, there are tabs for Configuration, Portal, and Permissions. The Configuration tab is active, showing fields for Name (Sandbox), a checked checkbox for Sandbox (which forces staging and publishing actions), a dropdown for Gateway Cluster (set to APIMOMT\_GATEWAY), and an API Endpoint section with a Base URL. A red box highlights the 'Portal' tab at the top of the configuration panel.

Click on **Portal** Tab.

This screenshot shows the same API Management interface, but the Portal tab is now active. The configuration fields remain the same, but the options under the Portal tab are visible. It includes sections for Basic Developer Portal (with a Portal URL of https://developer.apim.ibmcloud.com/hguvalamiraclesoftcom-iot/sb and a Customize URL of https://example.com), Advanced Developer Portal (with a URL of https://example.com), and Other (with a URL of https://example.com). A red box highlights the 'Portal' tab at the top of the configuration panel.

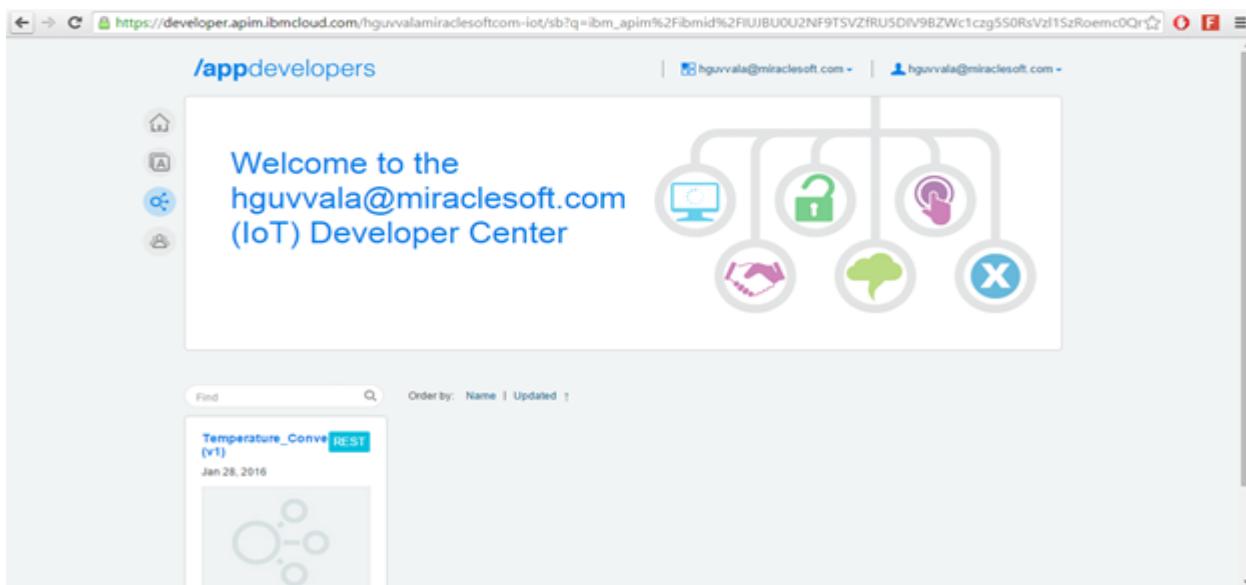
Click on the Basic Developer Portal checkbox and open the URL which was specified.

The screenshot shows the IBM Bluemix API Management interface. On the left, there's a sidebar with various icons. In the center, under 'Environments', there's a section for 'Sandbox APIMGMT\_GATEWAY'. Below this, there are tabs: Configuration, Portal (which is selected and highlighted with a red border), and Permissions. Under the Portal tab, there's a section for 'Basic Developer Portal' with a checked checkbox. The 'Portal URL' field contains 'https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb'. There are also fields for 'Customize URL' (set to 'https://example.com') and 'Customize Portal'. Below these, there are sections for 'Advanced Developer Portal' and 'Other', each with a URL field set to 'https://example.com'. At the bottom, there's a link 'User Registration and Invitation'.

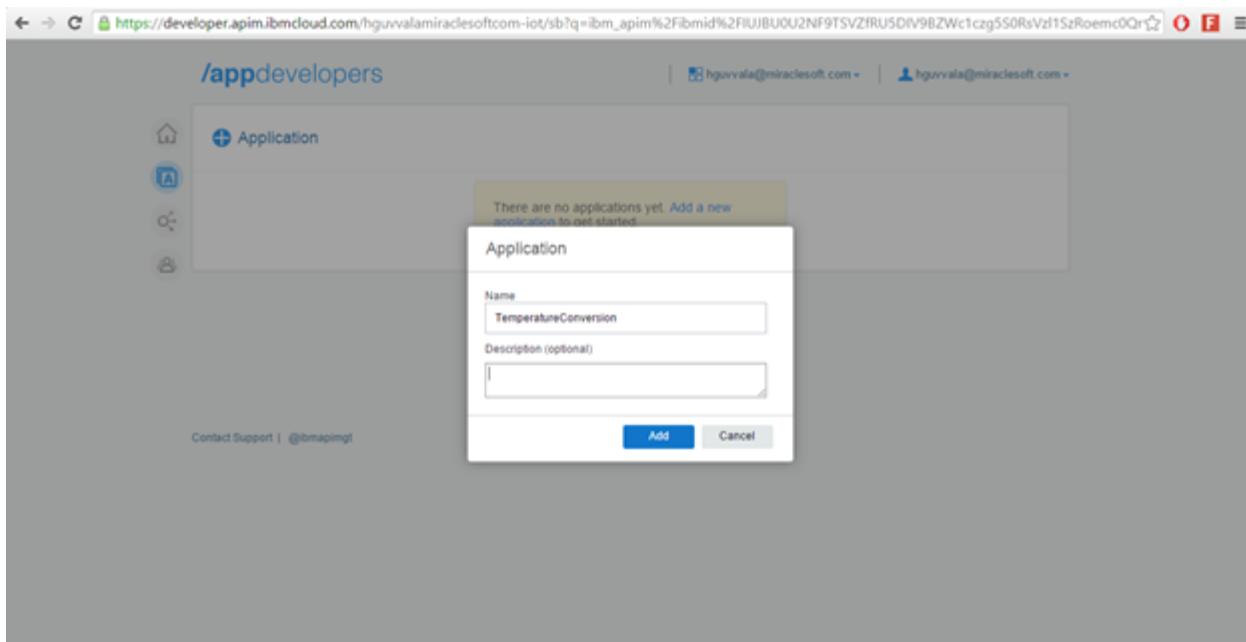
After clicking the URL you will be redirecting to API Developer Portal. Click on **SignIn**, it will automatically synchronize your credentials into Developer Portal and lets you login into that.

The screenshot shows the API Developer Portal at the URL 'https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb#apis/appdevelopers'. The page has a header with 'Sign in' and 'Join' buttons. Below the header, there's a welcome message: 'Welcome to the hguvvala@miraclesoft.com (IoT) Developer Center'. To the right of the message is a circular diagram with icons representing different developer tools or services. At the bottom of the page, there's a search bar and a list of APIs, with one API named 'Temperature\_Conver (v1)' shown in detail.

Click on **SignIn**.



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



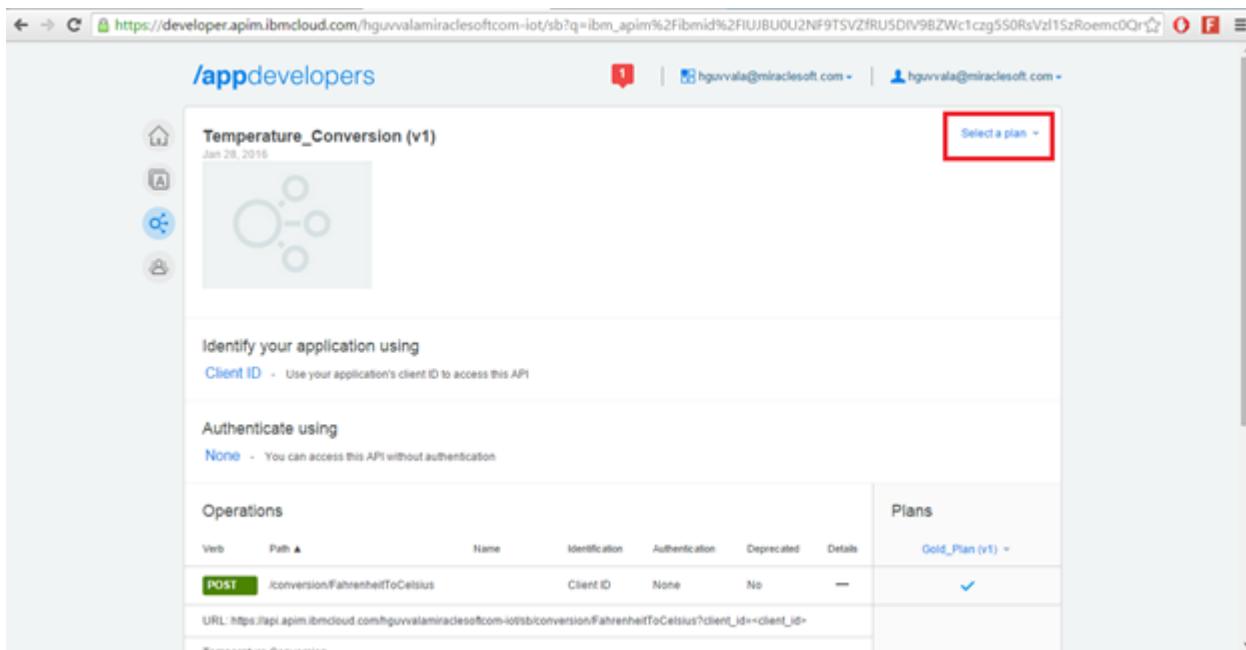
Click on the required API and copy the client ID as shown below.

The screenshot shows the IBM API Management developer portal at [https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm\\_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI1SzRoemc0Qr](https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI1SzRoemc0Qr). The left sidebar has icons for Home, Application, API, and User. The main area shows an application named "TemperatureConversion" with a blue icon. The "Client ID" field is highlighted with a red border. Below it, there are "Show" and "Reset" buttons. Other fields include "Client Secret" (with a "Show" button), "Subscribed Plans: None", "Redirection URL for OAuth (optional): https://localhost", and a "Description" field.

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

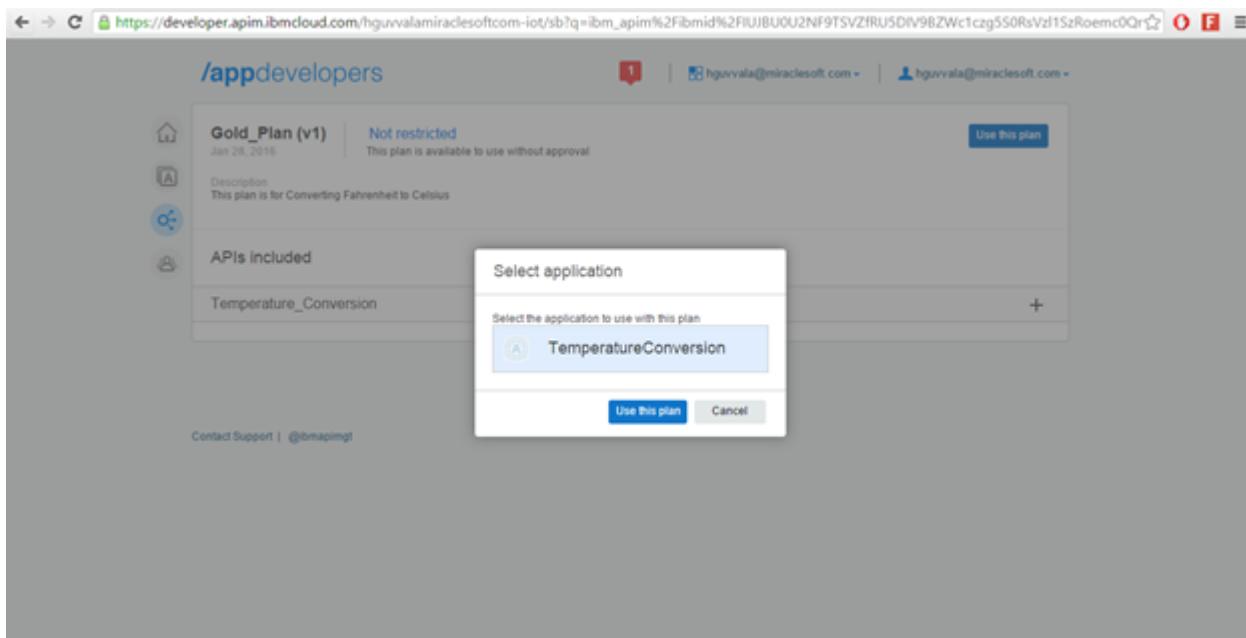
The screenshot shows the IBM API Management developer portal at [https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm\\_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI1SzRoemc0Qr](https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI1SzRoemc0Qr). The left sidebar has icons for Home, Application, API, and User. The main area shows an API named "Temperature\_Conversion (v1)" last updated on Jan 28, 2016. It includes sections for "Identify your application using Client ID" and "Authenticate using None". The "Operations" table lists a single endpoint: "POST /conversion/FahrenheitToCelsius". The "Plans" column shows "Gold\_Plan (v1)".

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



The screenshot shows the IBM API Management Developer Portal interface. At the top, there's a navigation bar with icons for back, forward, search, and user authentication. The URL in the address bar is [https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm\\_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI5zRoemc0QrL](https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI5zRoemc0QrL). Below the address bar, the page title is "/appdevelopers". On the left, there are four icons: a house (Home), a gear (APIs), a lock (Authentications), and a person (Applications). The main content area displays the "Temperature\_Conversion (v1)" API, which was last updated on Jan 28, 2016. It includes a small icon of three circles connected by lines. To the right of the icon is a "Select a plan" button, which is highlighted with a red box. Below the icon, there are sections for identifying the application using a Client ID and authenticating using None. The "Operations" table lists a single endpoint: a POST method for the path /conversion/FahrenheitToCelsius, requiring a Client ID and no authentication. The "Plans" column for this endpoint shows "Gold\_Plan (v1)" with a checkmark. At the bottom of the table, the URL is listed as [https://api.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb/conversion/FahrenheitToCelsius?client\\_id=<client\\_id>](https://api.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb/conversion/FahrenheitToCelsius?client_id=<client_id>).

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



The screenshot shows the IBM API Management Developer Portal interface. The URL in the address bar is [https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm\\_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI5zRoemc0QrL](https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DfV9BZWc1czg550RsVzI5zRoemc0QrL). The page title is "/appdevelopers". On the left, there are four icons: a house (Home), a gear (APIs), a lock (Authentications), and a person (Applications). The main content area displays the "Gold\_Plan (v1)" plan, which was last updated on Jan 28, 2016. The plan is described as "Not restricted" and "This plan is available to use without approval". There is a "Select application" button. A modal dialog box titled "Select application" is open, showing a list with "TemperatureConversion" selected. Below the list are "Use this plan" and "Cancel" buttons. At the bottom of the page, there is a "Contact Support | @ibmapimgt" link.

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.

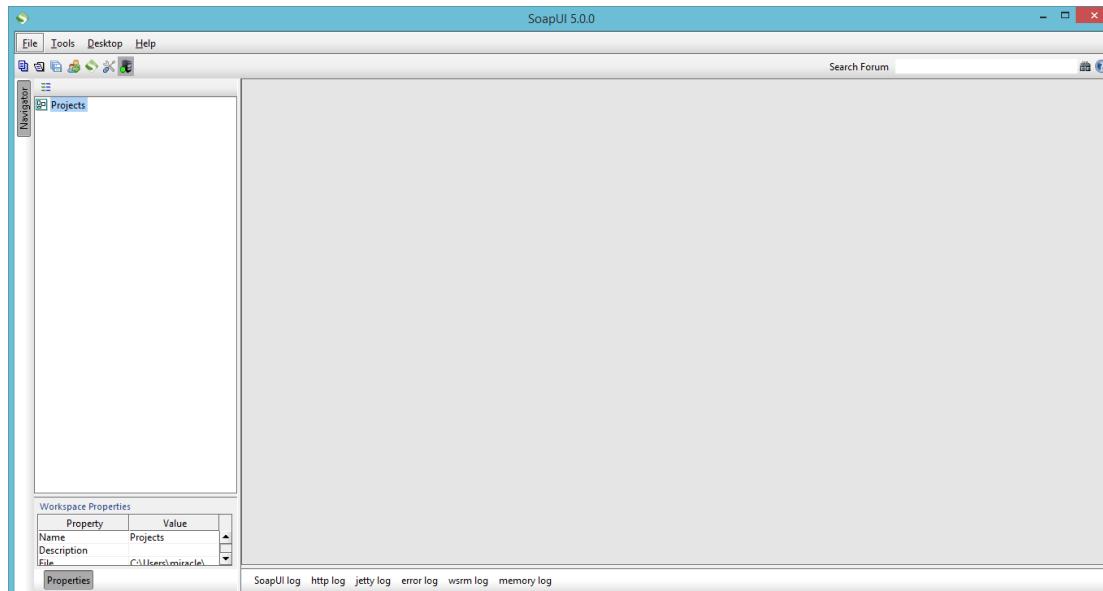
The screenshot shows the IBM API Management interface. The URL in the address bar is [https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm\\_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DIV9BVytBWIE1ZzIKRXIYWDQzMnpHK](https://developer.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb?q=ibm_apim%2Fibmid%2FIUJBU0U2NF9TSVZfRU5DIV9BVytBWIE1ZzIKRXIYWDQzMnpHK). The page title is /appdevelopers. The main content area displays the 'Temperature\_Conversion (v1)' API, which was created on Feb 1, 2016. It includes sections for identifying the application using Client ID and authenticating using None. The Operations table lists a POST operation for the path /conversion/FahrenheitToCelsius. A red box highlights the '+' button in the Details column of this row. The Plans section shows a single plan named 'Gold\_Plan (v1)'. At the bottom left, there is a link to Contact Support (@ibmapimg).

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

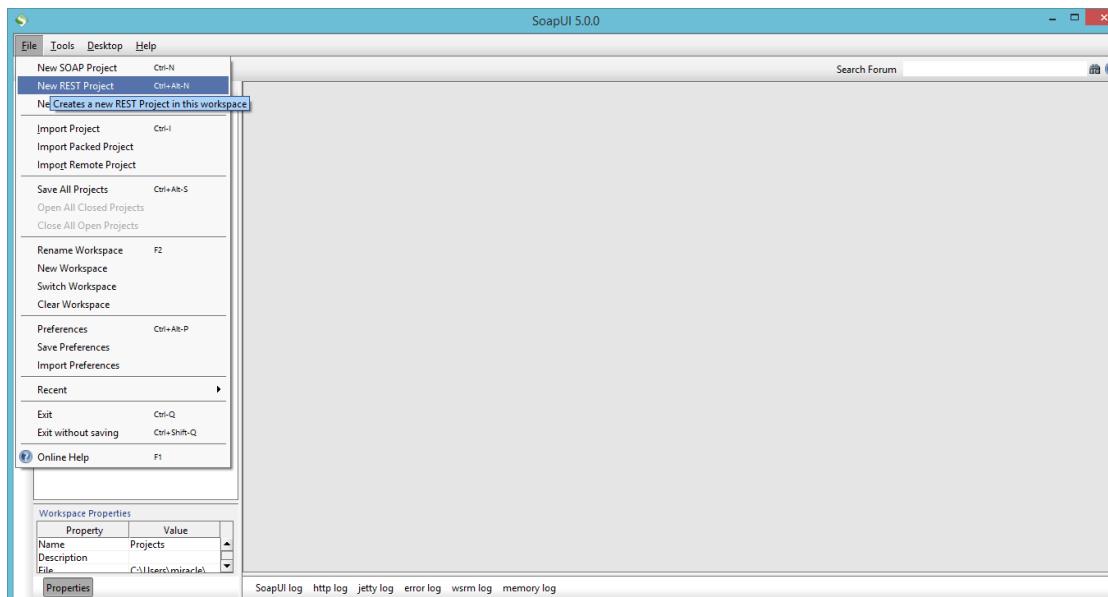
This screenshot shows the same IBM API Management interface as the previous one, but it highlights the URL for the POST operation. The URL [https://api.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb/conversion/FahrenheitToCelsius?client\\_id=<client\\_id>](https://api.apim.ibmcloud.com/hguvvalamiraclesoftcom-iot/sb/conversion/FahrenheitToCelsius?client_id=<client_id>) is highlighted with a red box. The rest of the interface, including the operations table and request body example, remains visible.

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,



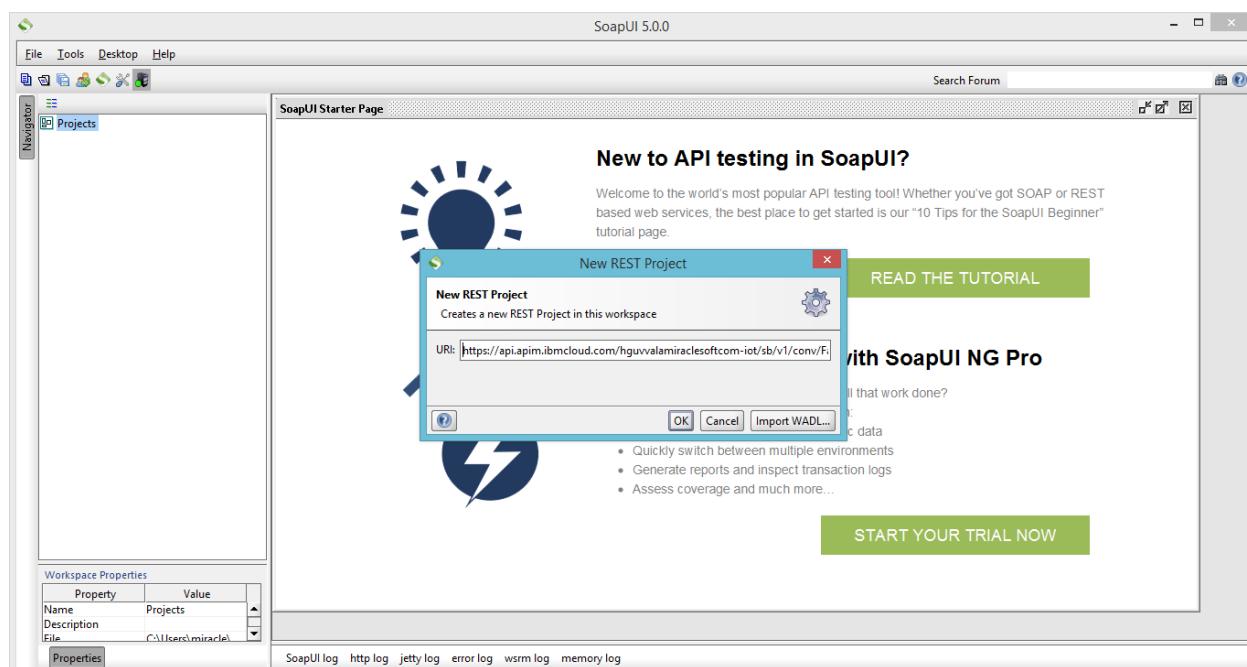
Create a new REST project from the File menu by choosing the "New REST Project" option in the File menu.



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

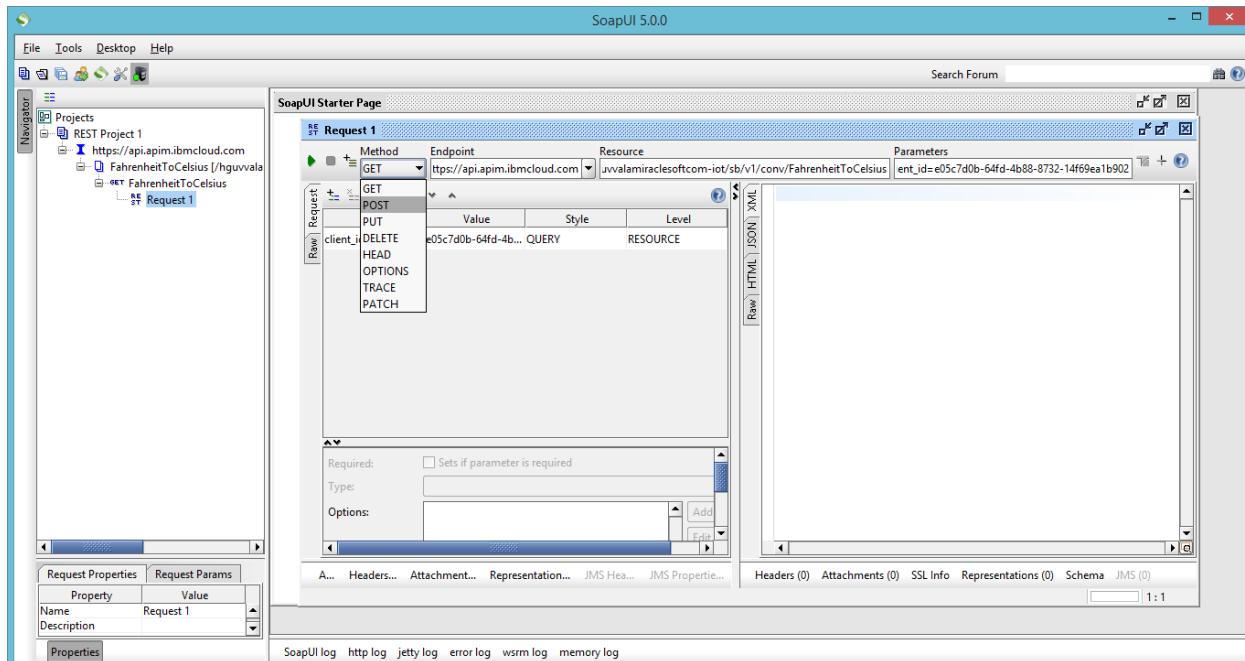
Specify the Client ID which was generated in IBM Developer Portal and the process okay.



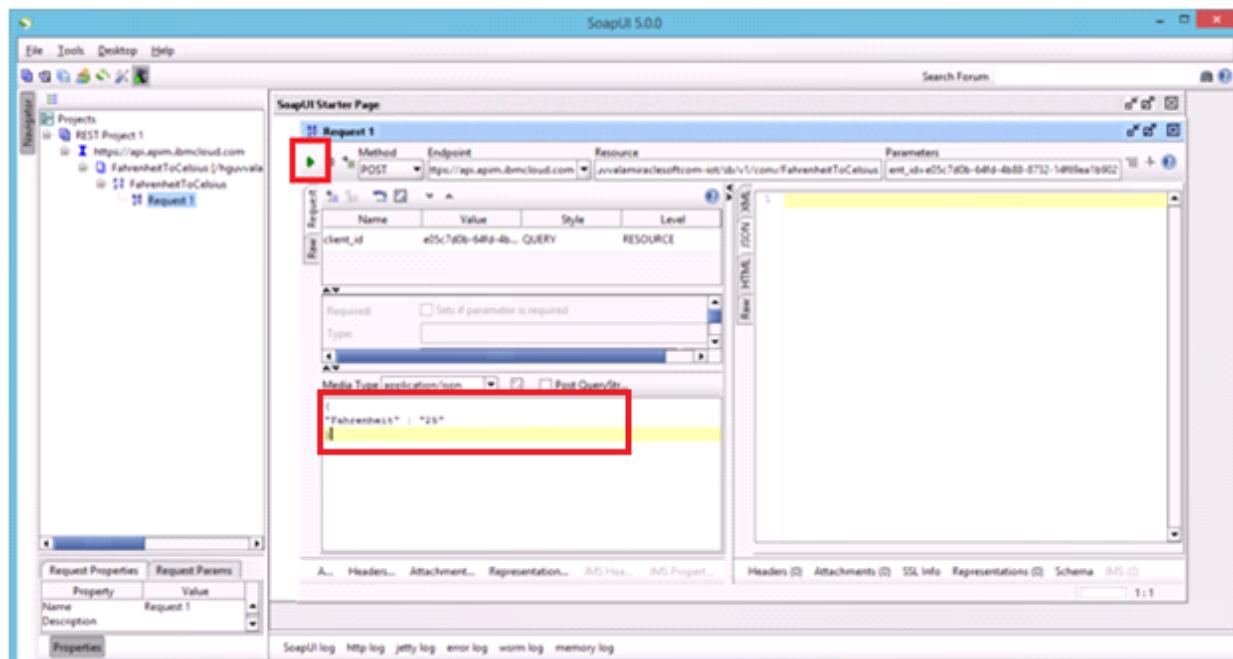
SOAPUI creates the project completely with a Service, Resource, Method and the actual Request and opens the Request editor.

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

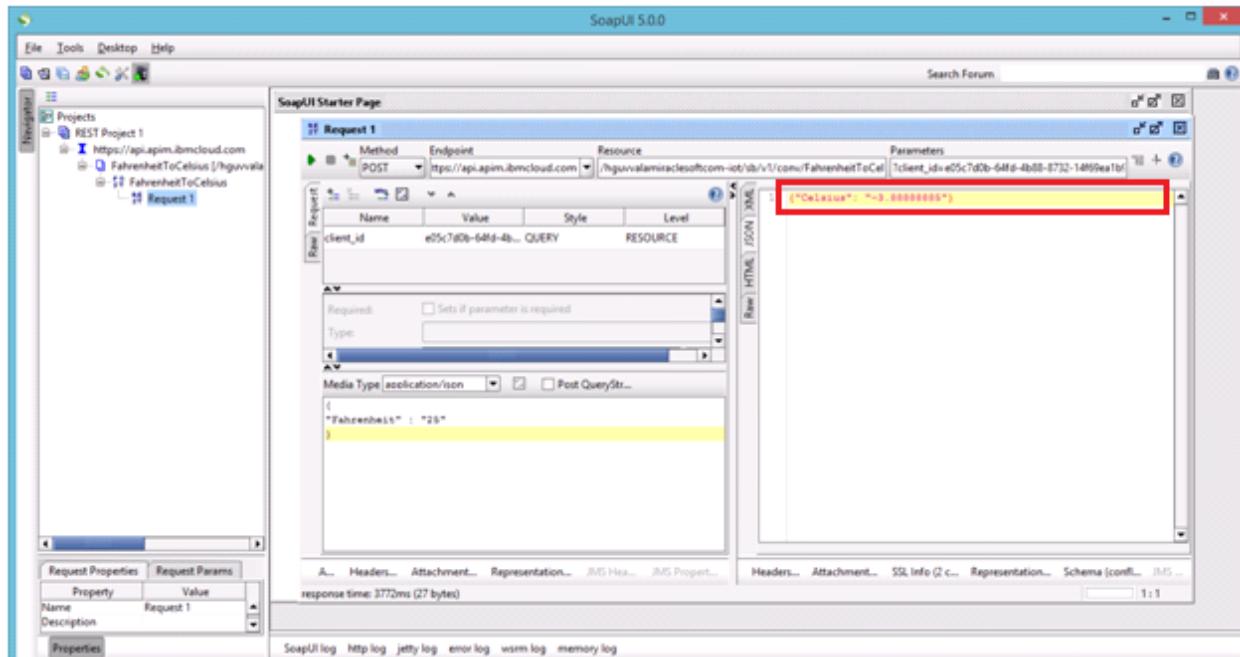
Click on JSON tab in the Right Navigation Pane for getting the result in JSON format.



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

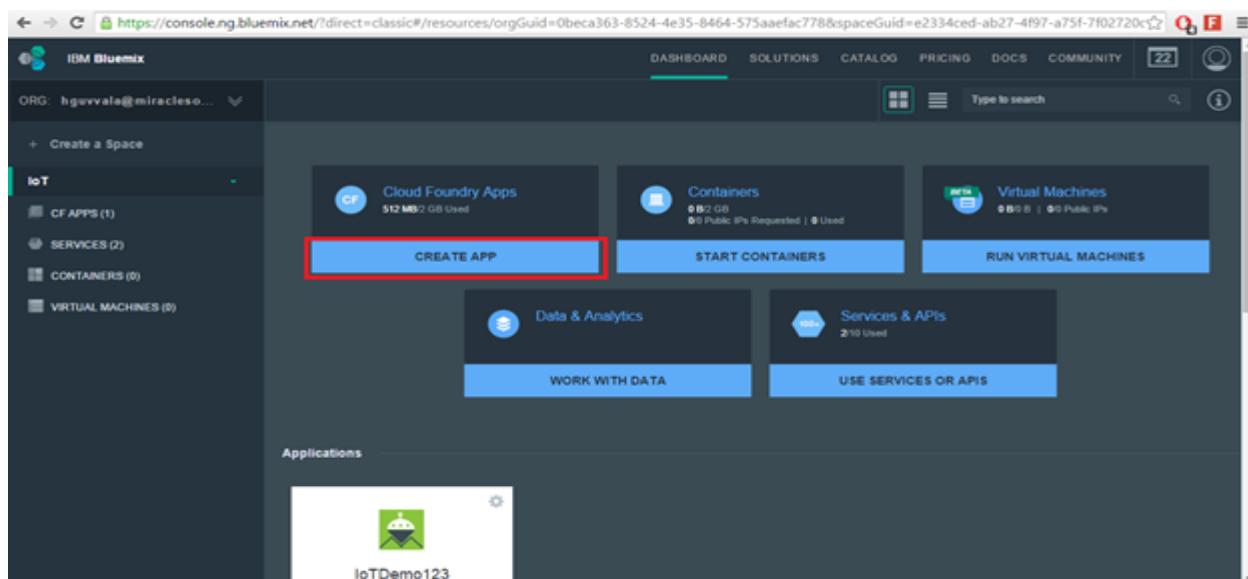


Press the green arrow at the top left in the Request editor and you can see the JSON output returned by the service.

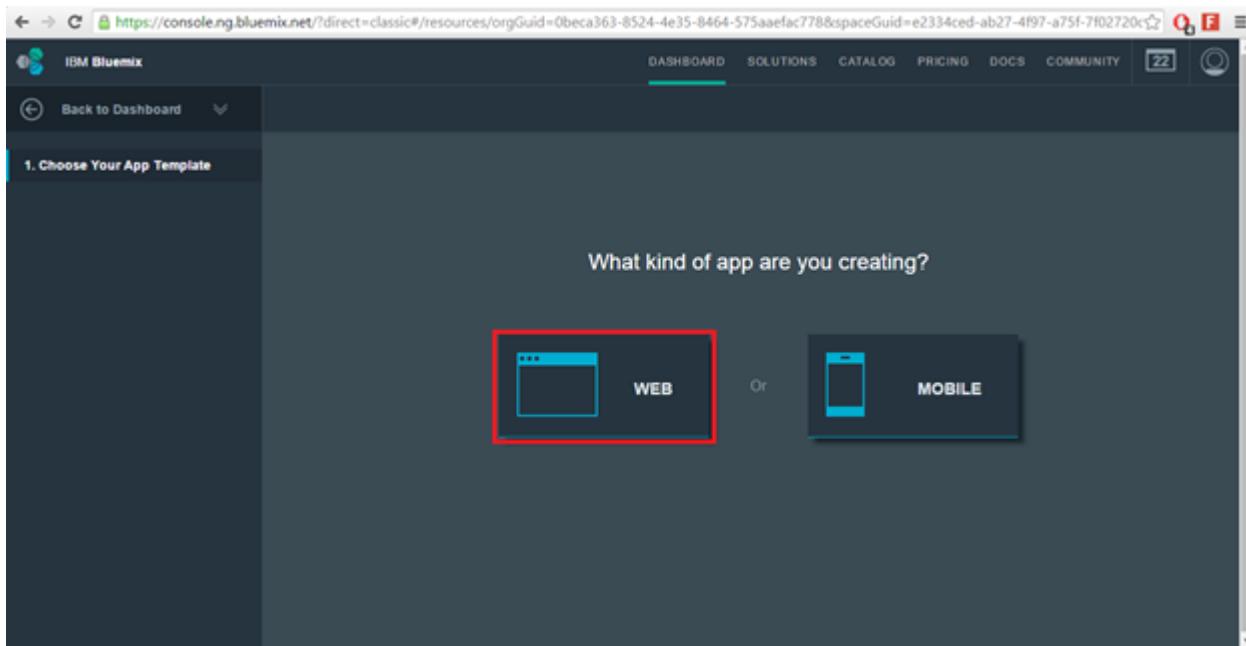


## #5 | Create Application in Bluemix

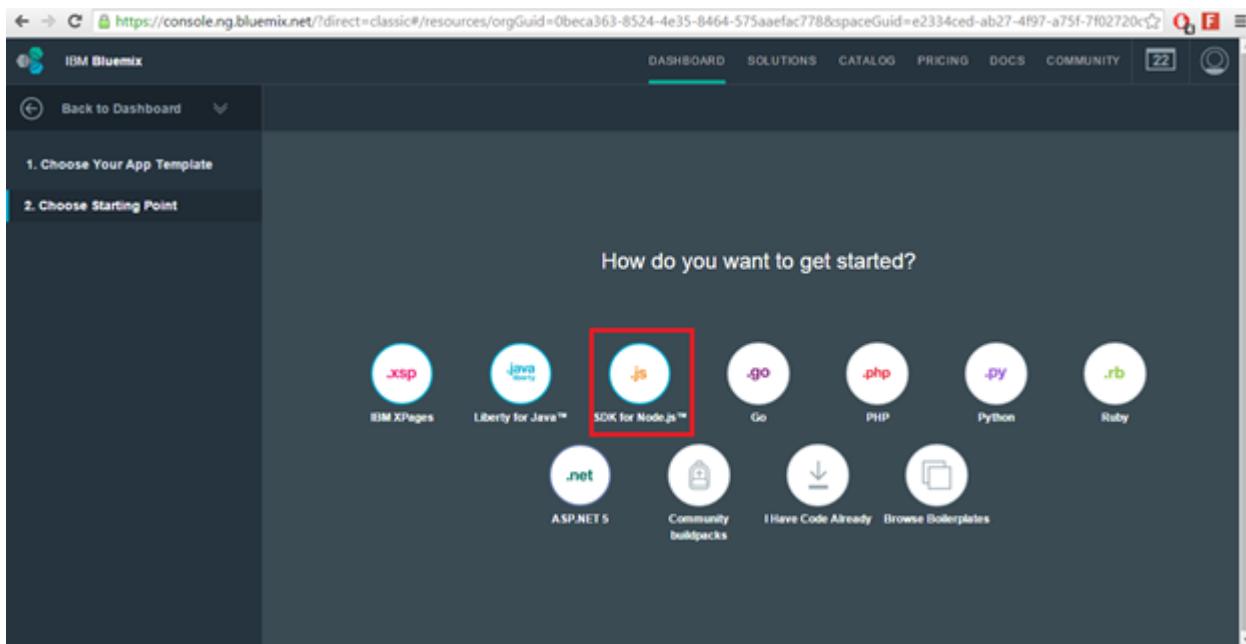
Click on the Cloud Foundry Apps module on the Dashboard to start creating the application.



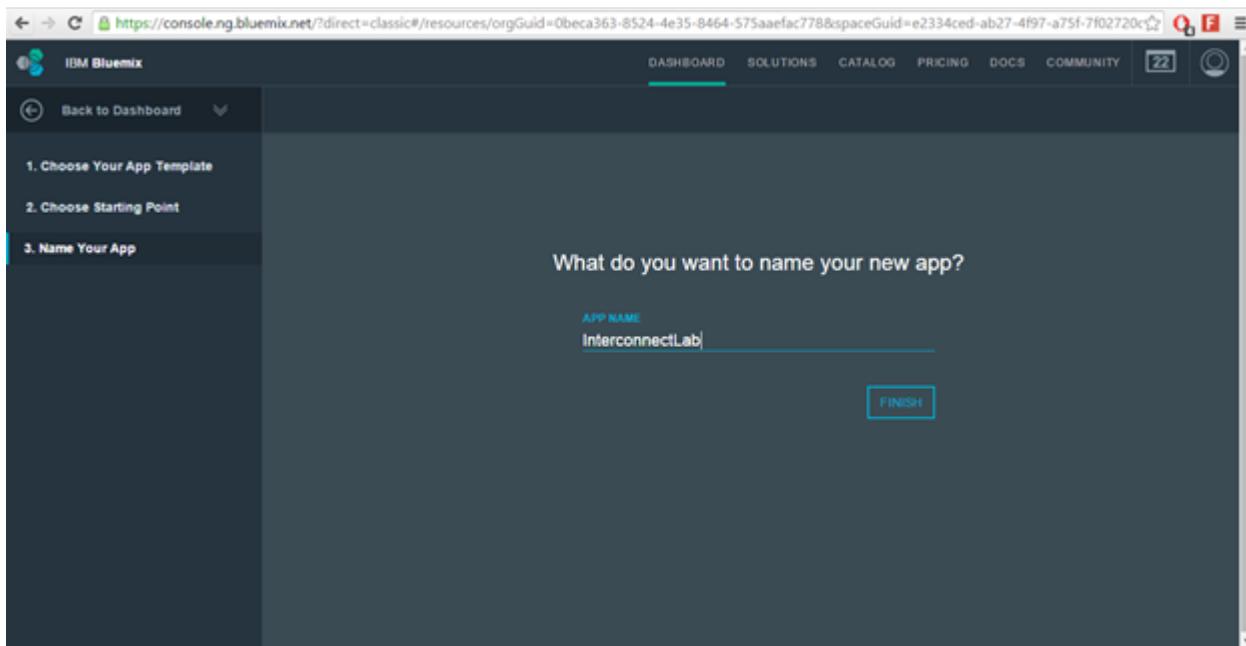
When prompted for the kind of application you want to create, please select Web Application.



Select “SDK for Node.js” in the page.

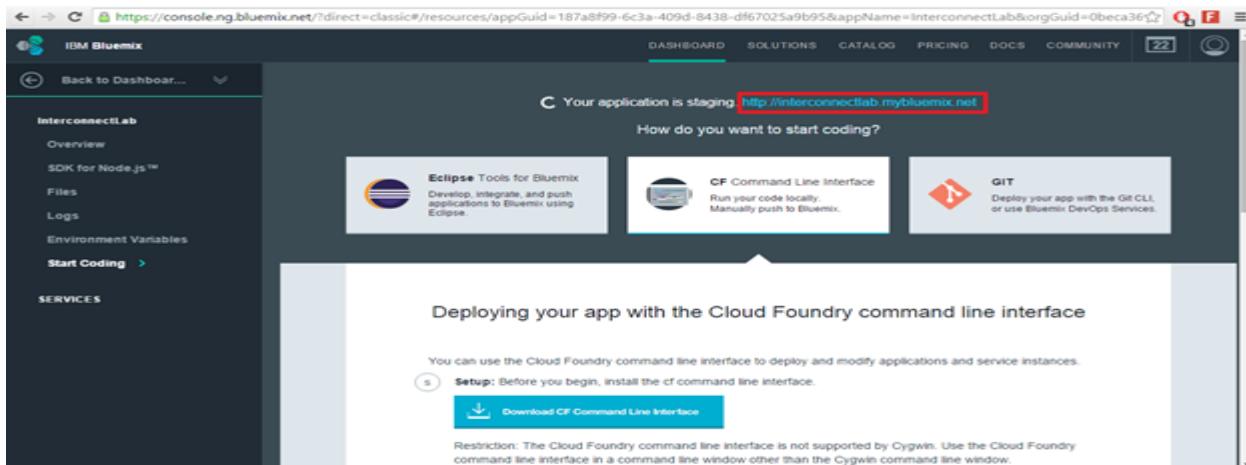


Click on continue and enter an application name and click finish. Application names must be unique as they will be on a public domain.

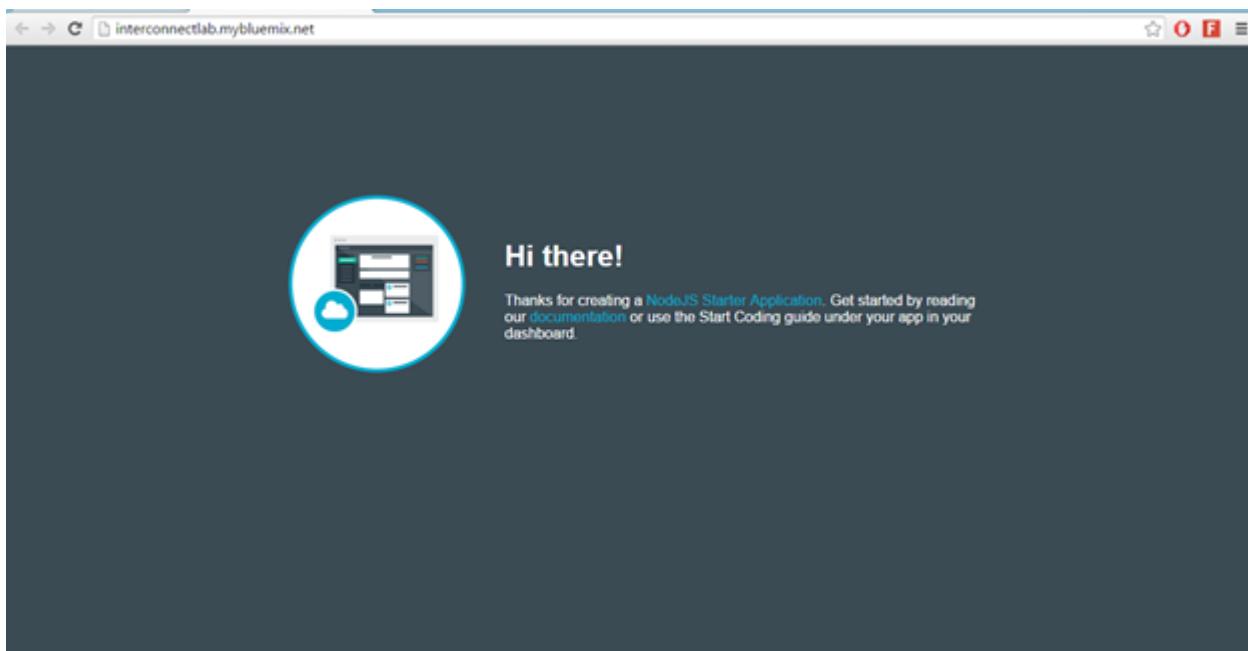


**Note :** Once created application will take about 2 minutes for staging and starting the application.

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.

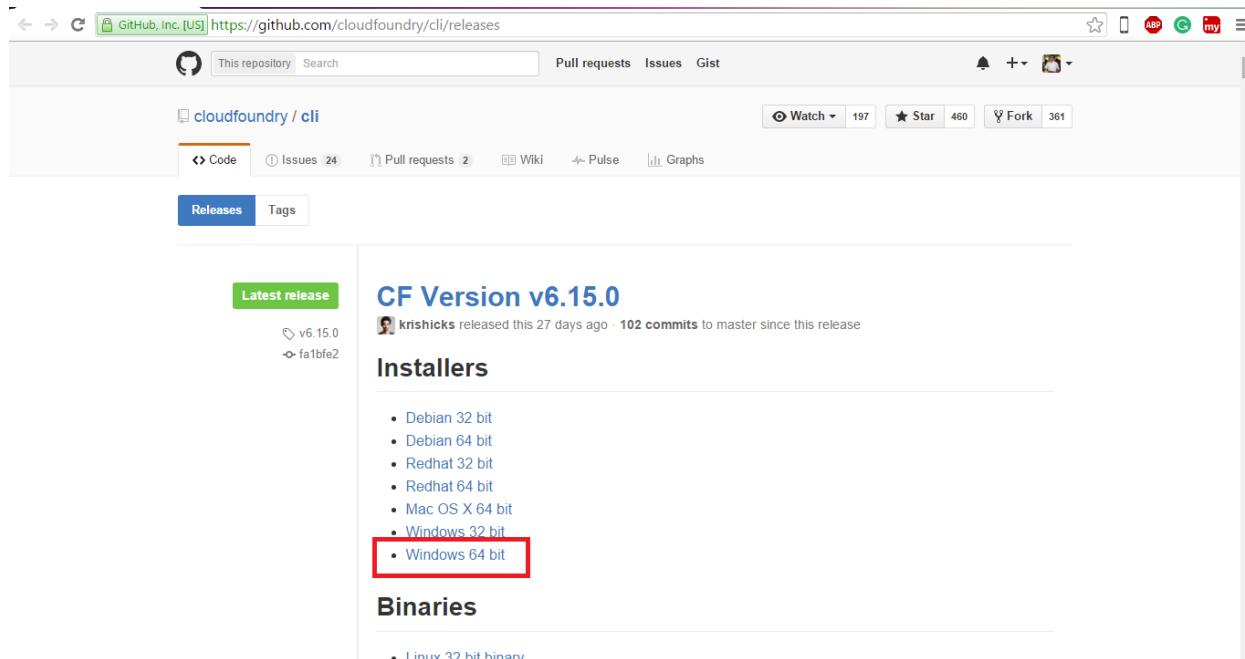


## #6 | Download and Edit Starter Code

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.

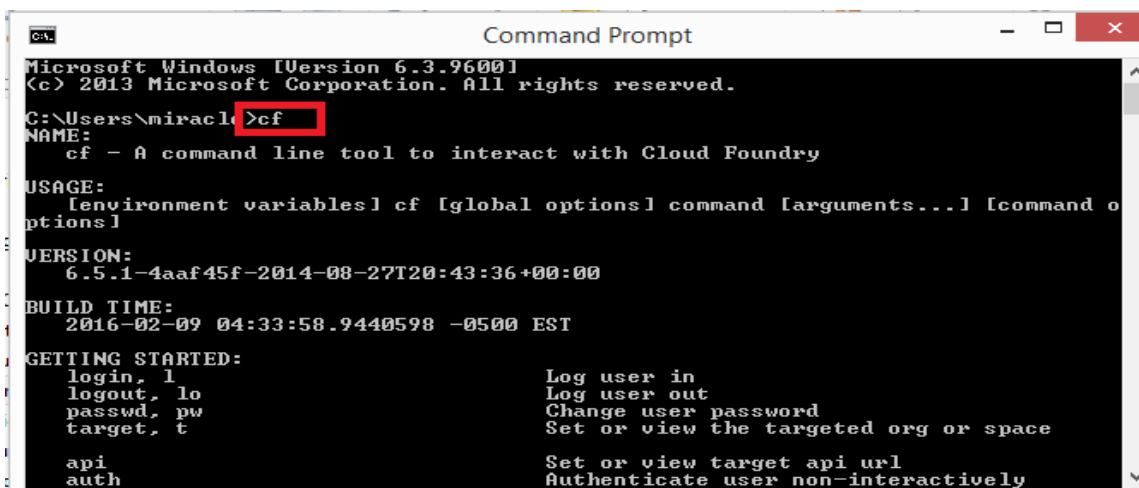
A screenshot of the IBM Bluemix dashboard. The left sidebar shows 'InterconnectLab' with options like 'Overview', 'SDK for Node.js™', 'Files', 'Logs', 'Environment Variables', 'Start Coding &gt;', and 'SERVICES'. The main content area has a dark grey header with tabs: DASHBOARD (highlighted), SOLUTIONS, CATALOG, PRICING, DOCS, and COMMUNITY. Below the header, there are three cards: 'Eclipse Tools for Bluemix' (with a blue icon), 'CF Command Line Interface' (with a grey icon), and 'GIT' (with a red icon). The 'CF Command Line Interface' card is expanded. It contains the text 'Deploying your app with the Cloud Foundry command line interface' and 'You can use the Cloud Foundry command line interface to deploy and modify applications and service instances.' It includes a step-by-step process: 'Setup: Before you begin, install the cf command line interface.' with a 'Download CF Command Line Interface' button (which is highlighted with a red box), and 'Restriction: The Cloud Foundry command line interface is not supported by Cygwin. Use the Cloud Foundry command line interface in a command line window other than the Cygwin command line window.' Below this, it says 'After the cf command line interface is installed, you can get started:' with a numbered list: '1. Download your starter code.' and a 'Download Starter Code' button.

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.



The screenshot shows the GitHub release page for the Cloud Foundry CLI. The page title is "CF Version v6.15.0". It includes a list of "Installers" with several options like Debian 32 bit, Debian 64 bit, Redhat 32 bit, Redhat 64 bit, Mac OS X 64 bit, Windows 32 bit, and Windows 64 bit. The "Windows 64 bit" option is highlighted with a red box.

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.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\miracle>cf
NAME:
  cf - A command line tool to interact with Cloud Foundry

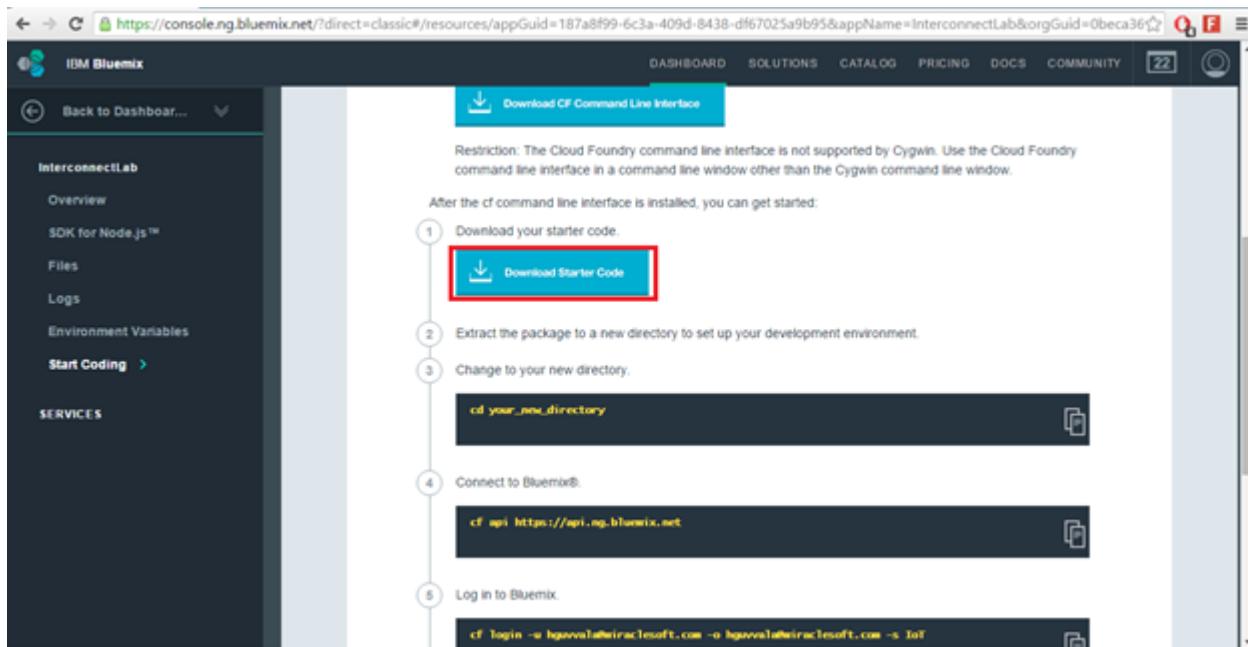
USAGE:
  [environment variables] cf [global options] command [arguments...] [command options]

VERSION:
  6.5.1-4aaf45f-2014-08-27T20:43:36+00:00

BUILD TIME:
  2016-02-09 04:33:58.9440598 -0500 EST

GETTING STARTED:
  login, l                               Log user in
  logout, lo                             Log user out
  passwd, pw                           Change user password
  target, t                            Set or view the targeted org or space
  api                                Set or view target api url
  auth                               Authenticate user non-interactively
```

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 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.

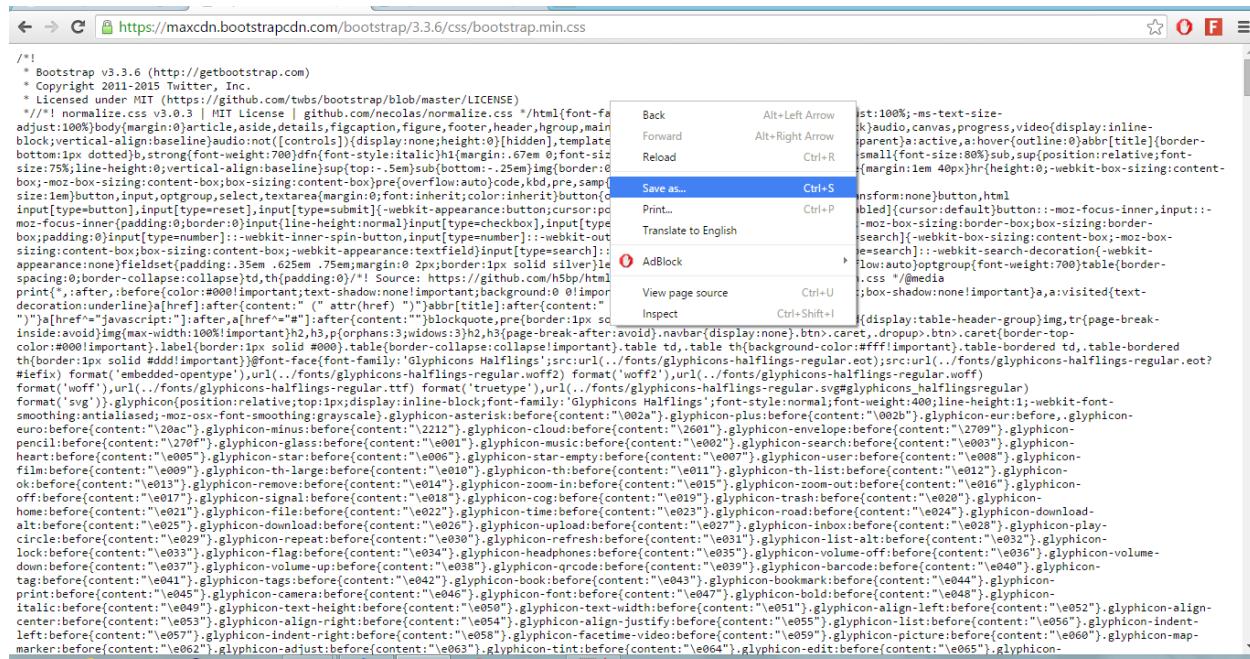
```
.modal-footer { border-top: 0px; }
```

Go to this URL,

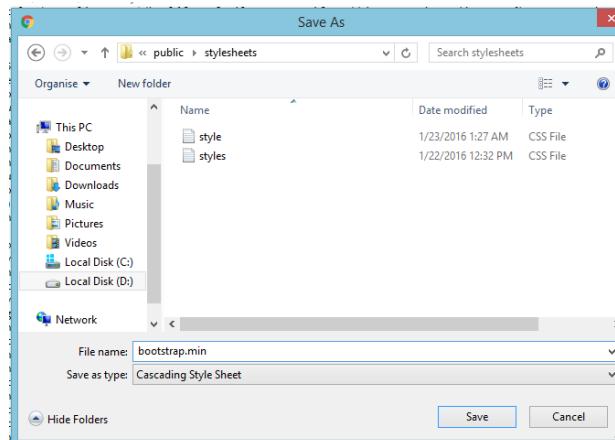
<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css>

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.



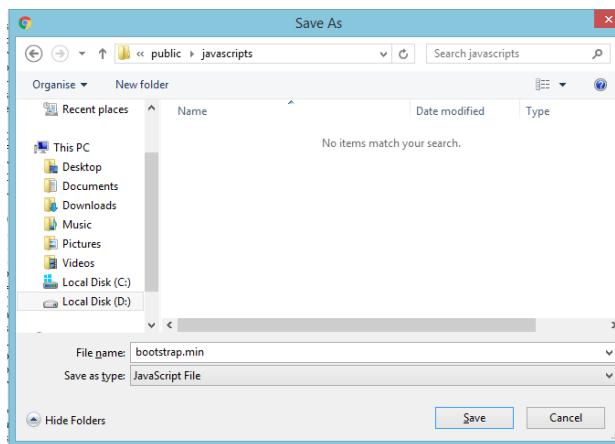
Go to this URL,

<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 the 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>
</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

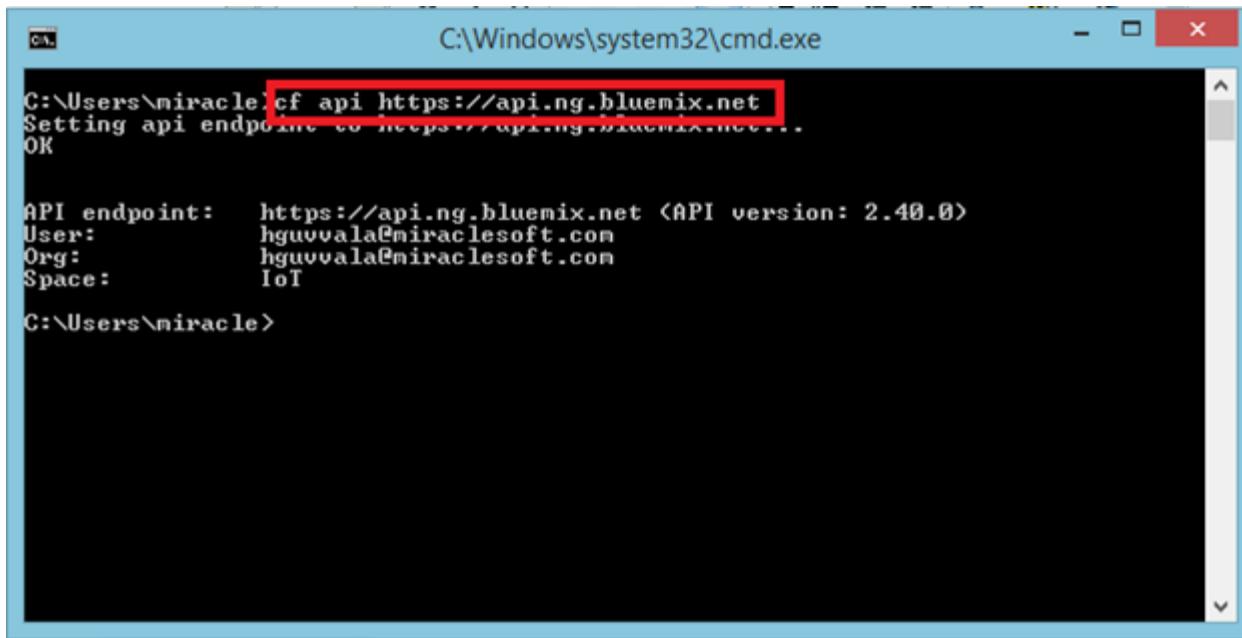
The next step will be to take your application and deploy it back to Bluemix so that you can share it with your friends.

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



A screenshot of a Windows Command Prompt window titled 'cmd' with the path 'C:\Windows\system32\cmd.exe'. The window contains the following text:

```
C:\Users\miracle>cf api https://api.ng.bluemix.net
Setting api endpoint to https://api.ng.bluemix.net...
OK

API endpoint: https://api.ng.bluemix.net (API version: 2.40.0)
User: hguvvala@miraclesoft.com
Org: hguvvala@miraclesoft.com
Space: IoT

C:\Users\miracle>
```

Login to Bluemix using the command,

**cf login**

When prompted enter username and password.

```
C:\Windows\system32\cmd.exe - cf login
OK

API endpoint: https://api.ng.bluemix.net (API version: 2.40.0)
User: hguvvala@miraclesoft.com
Org: hguvvala@miraclesoft.com
Space: IoT
C:\Users\miracle cf login
API endpoint: https://api.ng.bluemix.net
Email> hguvvala@miraclesoft.com

Password>
Authenticating...
OK

Targeted org hguvvala@miraclesoft.com
Select a space (or press enter to skip):
1. IoT
2. Raspberry
3. Raspberry-Pi
Space>
```

Specify the space where you deployed your application.

```
C:\Windows\system32\cmd.exe
Email> hguvvala@miraclesoft.com

Password>
Authenticating...
OK

Targeted org hguvvala@miraclesoft.com
Select a space (or press enter to skip):
1. IoT
2. Raspberry
3. Raspberry-Pi
Space> IoT
Targeted space IoT

API endpoint: https://api.ng.bluemix.net (API version: 2.40.0)
User: hguvvala@miraclesoft.com
Org: hguvvala@miraclesoft.com
Space: IoT
C:\Users\miracle>
```

Make sure that you are in application's directory.

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

```
C:\Windows\system32\cmd.exe - cf push

API endpoint: https://api.ng.bluemix.net (API version: 2.40.0)
User: hguvvala@miraclesoft.com
Org: hguvvala@miraclesoft.com
Space: IoT

C:\Users\miracle>d:
D:\>cd Interconnect_2016
D:\Interconnect_2016>cd InterconnectLab
D:\Interconnect_2016\InterconnectLab>cf push
Using manifest file D:\Interconnect_2016\InterconnectLab\manifest.yml

Updating app InterconnectLab in org hguvvala@miraclesoft.com / space IoT as hguvvala@miraclesoft.com...
OK

Using route interconnectlab.mybluemix.net
Uploading InterconnectLab...
Uploading app files from: D:\Interconnect_2016\InterconnectLab
Uploading 18.8K, 11 files
```

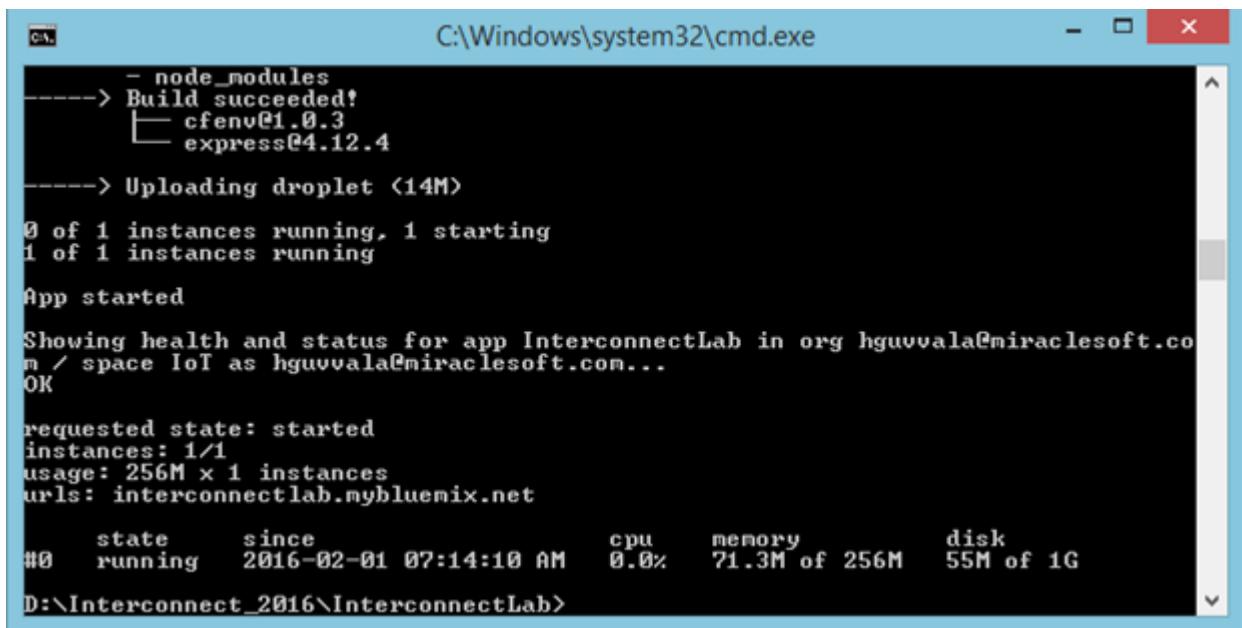
```
C:\Windows\system32\cmd.exe - cf push

----> Installing binaries
  engines.node <package.json>: 0.12.x
  engines.npm <package.json>: unspecified <use default>
  Resolving node version 0.12.x via 'node-version-resolver'
  Installing IBM SDK for Node.js (0.12.9) from cache
  Using default npm version: 2.14.9
----> Restoring cache
  Loading 1 from cacheDirectories <default>:
    - node_modules
----> Building dependencies
  Pruning any extraneous modules
  Installing node modules <package.json>
----> Installing App Management
----> Caching build
  Clearing previous node cache
  Saving 1 cacheDirectories <default>:
    - node_modules
----> Build succeeded!
  └─ cfenv@1.0.3
    └─ express@4.12.4
----> Uploading droplet <14M>

0 of 1 instances running, 1 starting
```

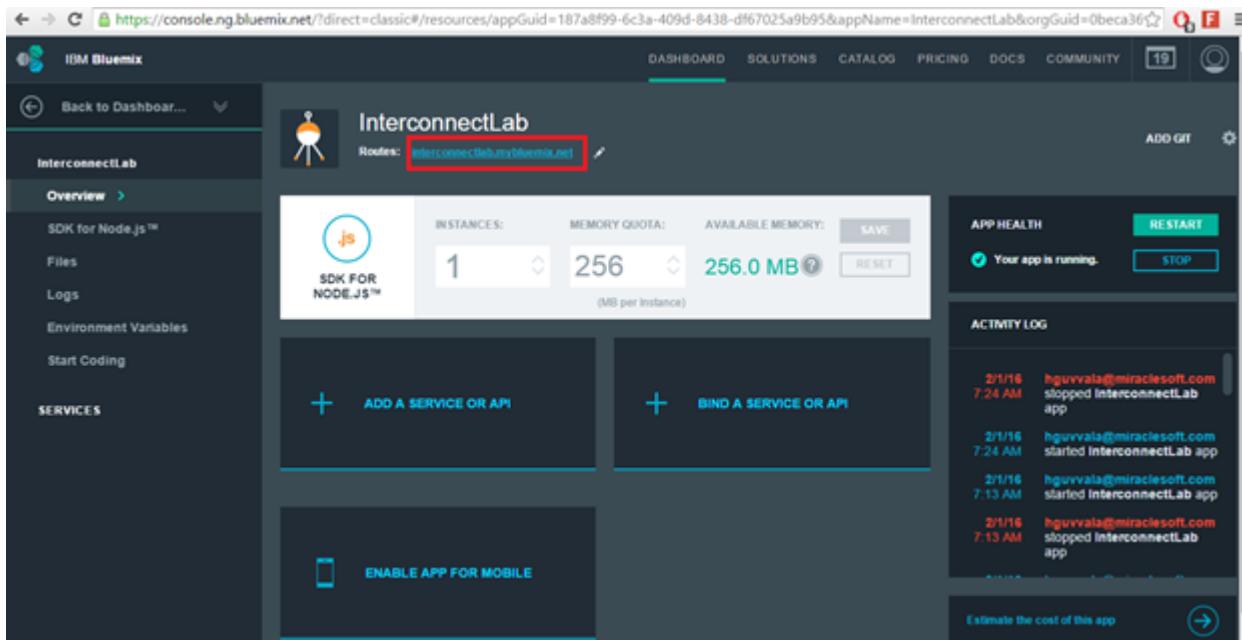
**Note :** This will take around 3 minutes for completion

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



```
C:\Windows\system32\cmd.exe
----> Build succeeded!
    cfenv@1.0.3
        express@4.12.4
----> Uploading droplet (14M)
0 of 1 instances running, 1 starting
1 of 1 instances running
App started
Showing health and status for app InterconnectLab in org hguvvala@miraclesoft.com / space IoT as hguvvala@miraclesoft.com...
OK
requested state: started
instances: 1/1
usage: 256M x 1 instances
urls: interconnectlab.mybluemix.net
#0  state      since
#0  running    2016-02-01 07:14:10 AM  cpu      memory      disk
                                0.0%   71.3M of 256M   55M of 1G
D:\Interconnect_2016\InterconnectLab>
```

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



The screenshot shows the IBM Bluemix Dashboard with the application 'InterconnectLab' selected. The 'Overview' tab is active. Key details shown include:

- Routes:** interconnectlab.mybluemix.net (highlighted with a red box)
- SDK FOR NODE.js™:** An icon showing a JS logo.
- INSTANCES:** 1
- MEMORY QUOTA:** 256 MB
- AVAILABLE MEMORY:** 256.0 MB
- APP HEALTH:** Your app is running.
- ACTIVITY LOG:** A list of log entries:
  - 2/1/16 7:24 AM hguvvala@miraclesoft.com stopped InterconnectLab app
  - 2/1/16 7:24 AM hguvvala@miraclesoft.com started InterconnectLab app
  - 2/1/16 7:13 AM hguvvala@miraclesoft.com started InterconnectLab app
  - 2/1/16 7:13 AM hguvvala@miraclesoft.com stopped InterconnectLab app
- Estimate the cost of this app:** A button with a right-pointing arrow.

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

