API Connect Hands-On Lab

Prerequisites

Needless to say you'll need a laptop! Any OS is fine, but make sure to install the following prior to the session:

- Git or "brew install git"
- The `cf` CLI from [https://github.com/cloudfoundry/cli#downloads] (https://github.com/cloudfoundry/cli#downloads) - download the latest version that is appropriate for your laptop and follow the instructions in README.txt.

  OR

  from http://docs.cloudfoundry.org/devguide/installcf/install-go-cli.html.
- Install npm and API connect Developer kit - Install `nodejs` and `npm` from https://nodejs.org/en/download/. Install API connect Developer Kit after installing `npm` from [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect)
- A Bluemix hosted instance

During the live HOL we will help you with a hosted environment (the details are below). We've done limited testing of the exercises on the following instance.

- [https://console.ng.bluemix.net/] (https://console.ng.bluemix.net/)

Samples and General Directions

Each directory is in a separate sub-directory. ***Ensure that you're in the sub-directory when you're working on a particular exercise and you're issuing the CLI commands from the subdirectory pertaining to the exercise.***

Recommended Exercises - User Related

It is recommended that you run through these exercises sequentially since they are progressive with some dependencies. Each exercise should take about 5-10 mins. to complete.

- Exercise 1: Install Node.js and the API Connect toolkit
- Exercise 2: Create a Weather API using API Connect on Bluemix
- Exercise 3: Design your OpenAPI Swagger specification
- Exercise 4: Generate a LoopBack application and import your APIs
- Exercise 5: Create a database service on Bluemix

- Exercise 6: Create database CRUD APIs with LoopBack models
- Exercise 7: Test, Explore and Deploy your LoopBack application
- Exercise 8: Explore your deployed APIs with the API Manager on Bluemix
- Exercise 9: Generate a Developer Portal for your APIs

More Resources

Quick tour of API connect at [https://console.ng.bluemix.net/docs/services/apiconnect/index.html] (https://console.ng.bluemix.net/docs/services/apiconnect/index.html)

API Connect Developer Toolkit at [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect)

Contact

Please contact us on Twitter @ragss or @boilerupnc or @sai_vennam.

API Connect Hands-On Labs

Exercise 1: Install Node.js, API Connect Toolkit and create a sample "hello world" API connect project

Prerequisites

If you're working on your own laptop, make sure you've met the following prerequisites (**prerequisites 2-4 may already have been met if you're using a pre-installed laptop. You can verify this by typing `cf` and `apic` commands**).

Prerequisite 1: Registered for a Bluemix account that is **still current** (trial Bluemix accounts are available at http://console.ng.bluemix.net). Contact the instructor for a promotion code for a bump in the quota. Please note down the `username` (or `email`) and `password` which will be used to login via the `cf` CLI.

Prerequisite 2: Installed the Cloud Foundry CLI from https://github.com/cloudfoundry/cli#downloads.

Prerequisite 3: Installed `npm` and `apic`. Refer to instructions from [https://nodejs.org/en/download/] (https://nodejs.org/en/download/) and [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect) respectively.

Prerequisite 4: Installed the Hands-On Labs locally. You can either `git clone` [https://github.com/ragsns/apichol] (https://github.com/ragsns/apichol) or download a zip from the repository.

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex1.

```
cd <path-to-hol-folder>/apichol/exercises/ex1
```

Overview

We will target a Bluemix Cloud Foundry instance using the Bluemix ID created and look at the API Connect service locally.

Target the Bluemix instance

Target the Bluemix Cloud Foundry instance by substituting the URL with the one provided and use the following command.

```
cf api https://api.ng.bluemix.net # to Americas
```

OR

```
cf api https://api.eu-gb.bluemix.net # to Europe
```

The output for the `cf` CLI should look something like below.

```
Setting api endpoint to https://api.ng.bluemix.net...
OK


API endpoint:   https://api.ng.bluemix.net (API version: 2.27.0)
Not logged in. Use 'cf login' to log in.
```

Login to the instance as directed.

```
cf login
```

Substitute the **non-expired** Bluemix account that was created earlier as below.

```
API endpoint: https://api.ng.bluemix.net

Email> <your IBM ID>

Password>
Authenticating...
OK

Targeted org raghsrin@us.ibm.com

Targeted space dev


API endpoint:   https://api.ng.bluemix.net (API version: 2.27.0)
User:           raghsrin@us.ibm.com
Org:            raghsrin@us.ibm.com
Space:          dev
```

List the spaces with the following command

```
cf spaces
```

The output will look something line below.

```
Getting spaces in org raghsrin@us.ibm.com as raghsrin@us.ibm.com...

name
dev
```

If there are no space(s) listed, then create a space `dev` with the following command.

```
cf create-space dev
```

The output will look something like below.

```
Creating space dev in org raghsrin@us.ibm.com as raghsrin@us.ibm.com...
OK
Assigning role SpaceManager to user raghsrin@us.ibm.com in org raghsrin@us.ibm.com / space d
OK
Assigning role SpaceDeveloper to user raghsrin@us.ibm.com in org raghsrin@us.ibm.com / space
OK

TIP: Use 'cf target -o raghsrin@us.ibm.com -s dev' to target new space
```

Issue the command as provided in `TIP` above as below to target the newly created space (if required).

```
cf target -o <your IBM ID> -s dev
```

The output will look something like below.

```
API endpoint:   https://api.ng.bluemix.net (API version: 2.27.0)
User:           raghsrin@us.ibm.com
Org:            raghsrin@us.ibm.com
Space:          dev
```

List the apps by issuing the following command.

```
cf apps
```

The output will look something like below.

```
Getting apps in org raghsrin@us.ibm.com / space dev as raghsrin@us.ibm.com...
OK

No apps found
```

Next we will create a simple `hello-world` project using API Connect.

Create a "hello world" API connect project

Create a Loopback application. Pick the defaults except as noted below.

```
apic loopback --name notes
```

Pick the defaults except for the sample application notes instead of the default option.

```
? What kind of application do you have in mind? notes (A project containing a basic working
y database)
```

Change to the project directory

```
cd notes
```

Start the API connect services locally

```
apic start
```

Ensure the service is running via the command

```
curl -l localhost:4001
```

Which should display how long the service has been running

You can try other options as available in the following command

```
apic --help
```

You can post a note via the following `cURL` command

```
curl -k -X POST https://localhost:4002/api/Messages -H 'X-IBM-Client-Id: default' -H 'X-IBM-
```

You should see an output that looks something like below which indicates that a message with an ID of 1 was created.

```
* Closing connection 0
{"id":1,"greeting":"Hello World"}
```

You can retrieve the message via the following command.

```
curl -k --request GET --url 'https://localhost:4002/api/Messages' --header 'accept: applicat
```

You can explore further by invoking the following command

```
SKIP_LOGIN=true apic edit
```

This will launch the API connect GUI in the default browser. You will have to override the certificates manually for the API connect product in order to invoke the `cURL` commands. For now, feel free to wander around. We will take a more formal guided tour in the subsequent exercise.

Finally, you can stop the service as below.

```
apic stop
```

Which should show the service being stopped.

You can clean up by deleting the sub-directory if you prefer.

```
cd ..
rm -rf notes
```

We will dive into API Connect in the subsequent exercises.

Summary of Exercise

After installing all the prerequisites, we explored how to target a Cloud Foundry instance and use the API connect product locally. Next we will look into how to use the product on Bluemix.

API Connect Hands-On Labs

Exercise 2: Create a Weather API using API Connect on Bluemix

Prerequisites

Make sure you've met the following prerequisites.

Prerequisite 1: Completed Exercise 1.

Overview

In the previous exercise we looked at how to use loopback framework on the `localhost` which is great for development. In this exercise, we will now look at how to use the API Connect service as a service on Bluemix. The service provides an API connect manager which provides the same tasks that were available via the API Connect GUI locally.

The service also enables publication of the APIs to a developer portal which will be covered in a subsequent exercise.

Login to Bluemix and instantiate the API Connect Service

Login to Bluemix by providing the credentials that you used during the registration process.

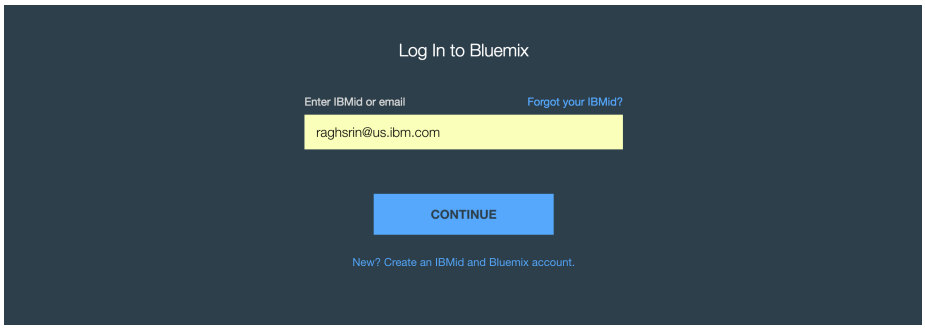

Figure 1:

Once logged in, click on `Catalog` tab.

Pick the API Connect service tile

Pick the defaults for the service

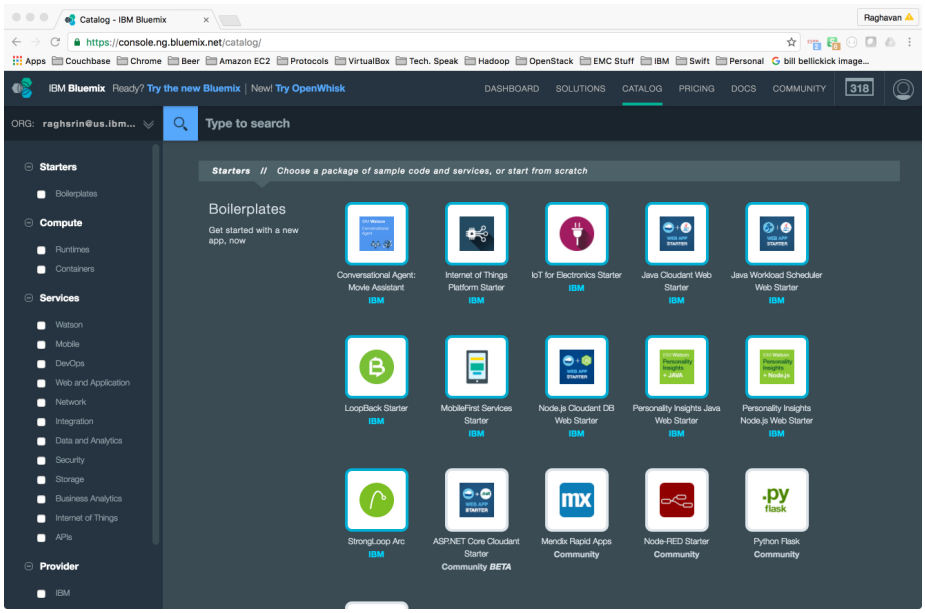

Figure 2:

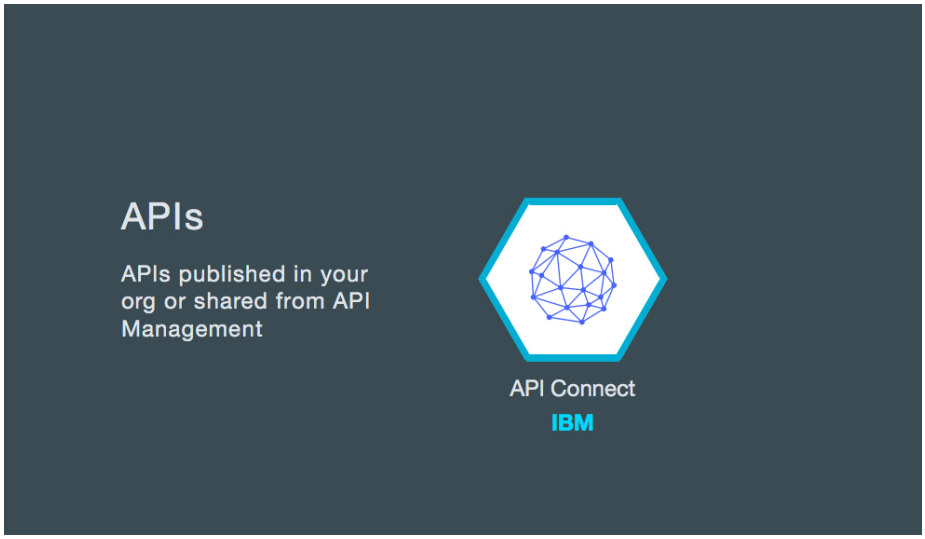

Figure 3:

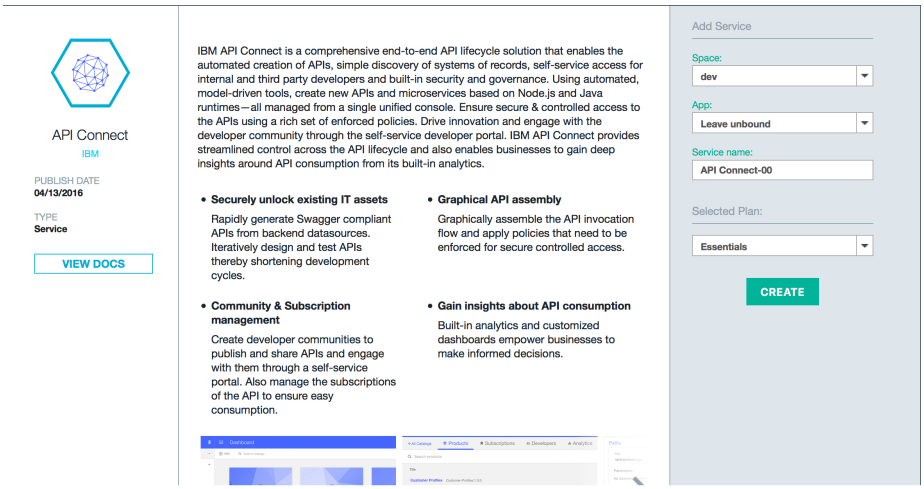

Figure 4:

You will presented tith a Drafts API screen that looks like below.

Click on “Got it” and proceed.

You will be presented with the Drafts screen that looks like below.

If you do not have the “Getting Started” window, click on ? and “Turn on Guided Tour” as shown below.

We will follow the “Getting Started” tour essentially from now on.

A tour of the API Connect Manager via the sample Weather API

We start off by clicking on “Import API” in the “Getting Started” window and “Import a Sample API” as prompted.

In the Dialog box shown below pick the “Climbing Weather” project and hit “Import” as shown below.

This will prompt you about the “API Editor” as shown below. Hit “Got it!”.

This will bring up the Design view similar to what you saw earlier, this time on Bluemix.

Scroll down to the `getWeather` method which looks like below.

Click “Generate and Publish” in the “Getting Started” window picking “Generate a Default Product” as shown below.

You will see that a default plan will be used and that you will publish the product to the Sandbox catalog. Click on “Generate” as shown below.

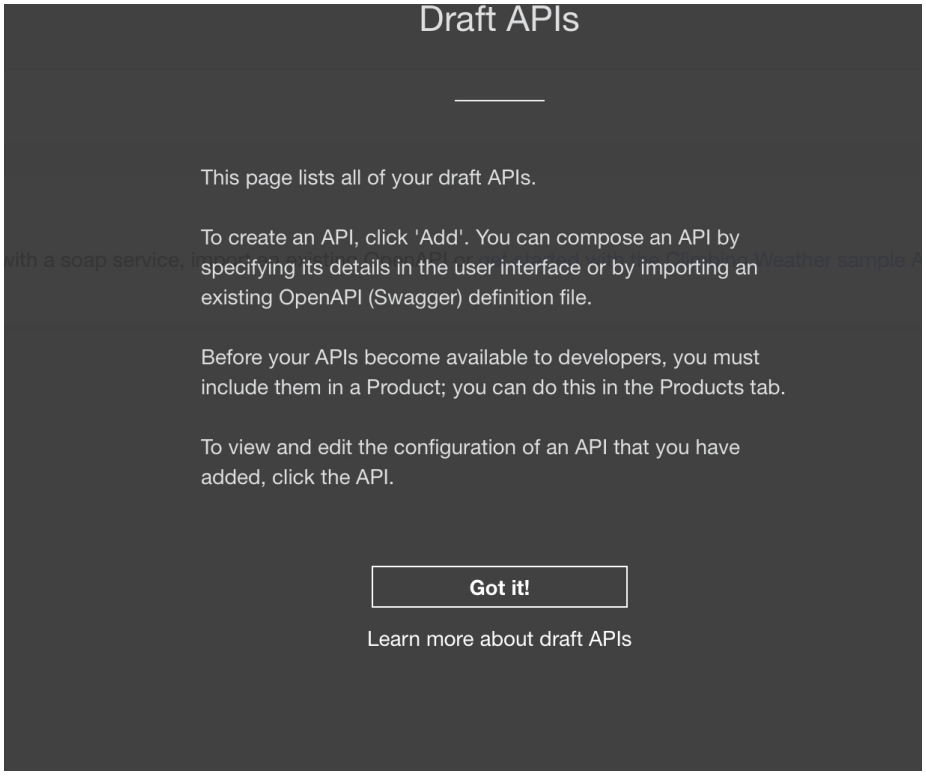

Figure 5:

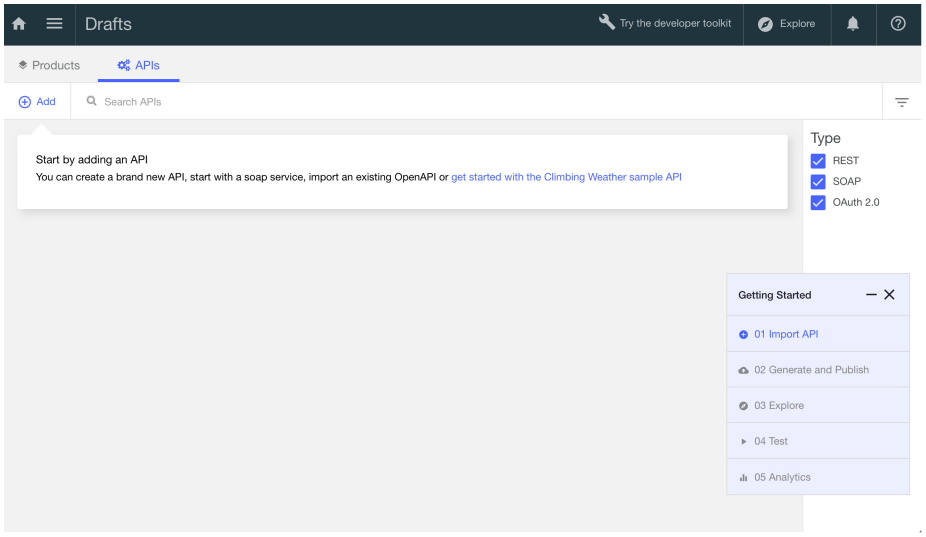

Figure 6:

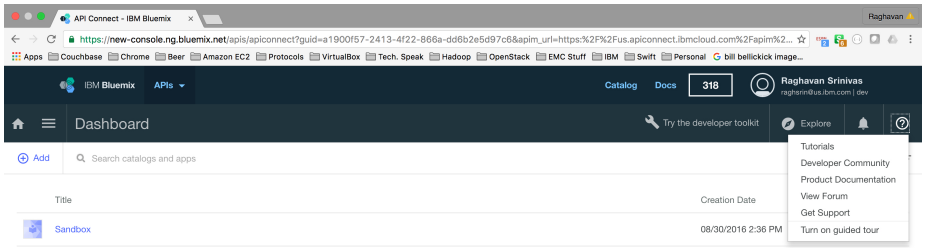

Figure 7:

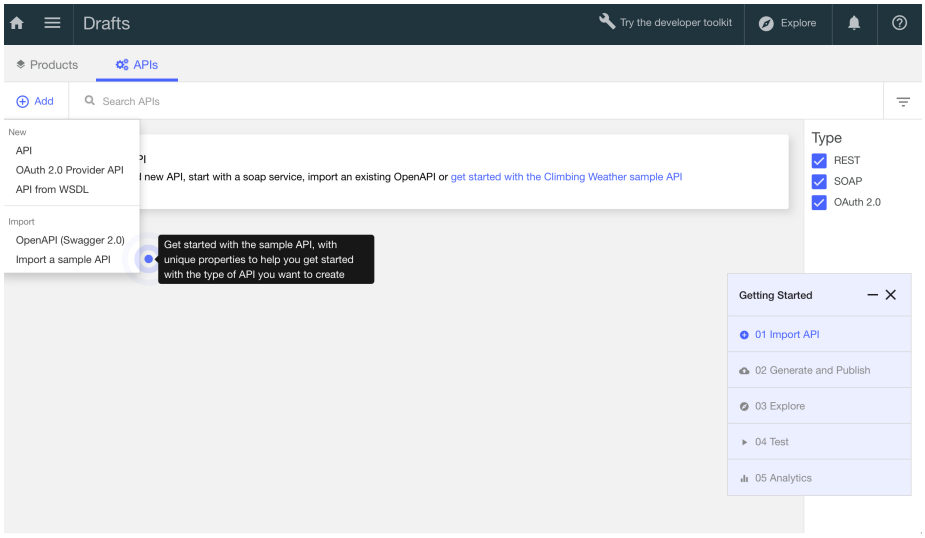

Figure 8:

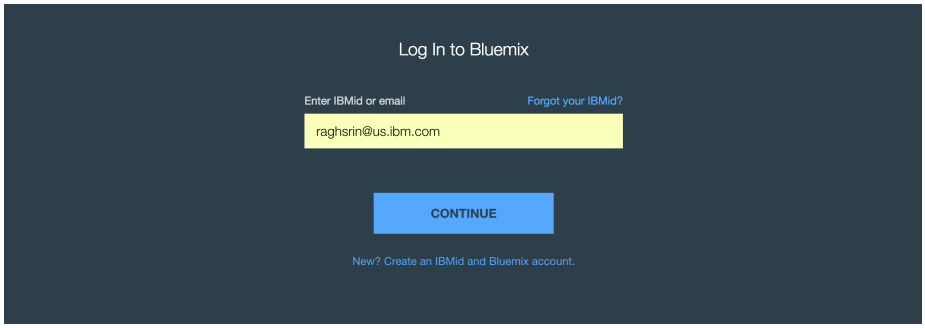

Figure 9:

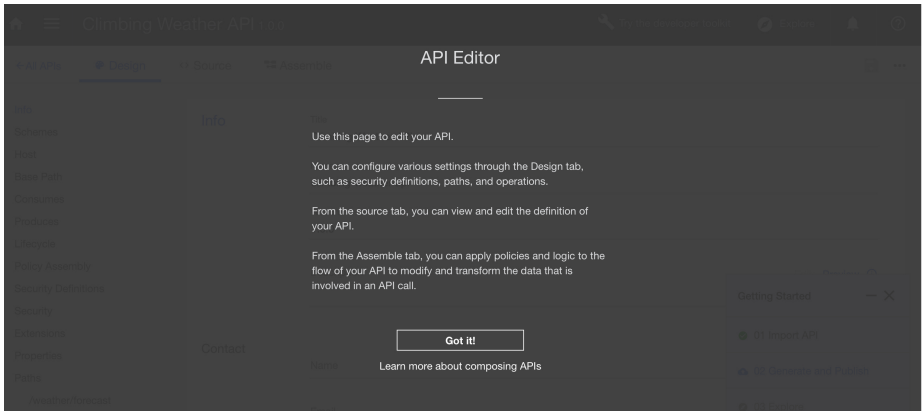

Figure 10:

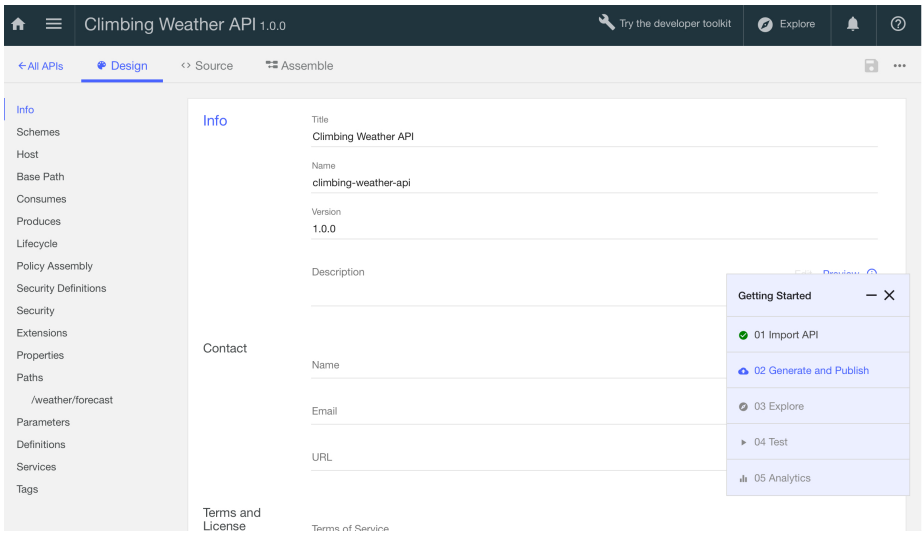

Figure 11:

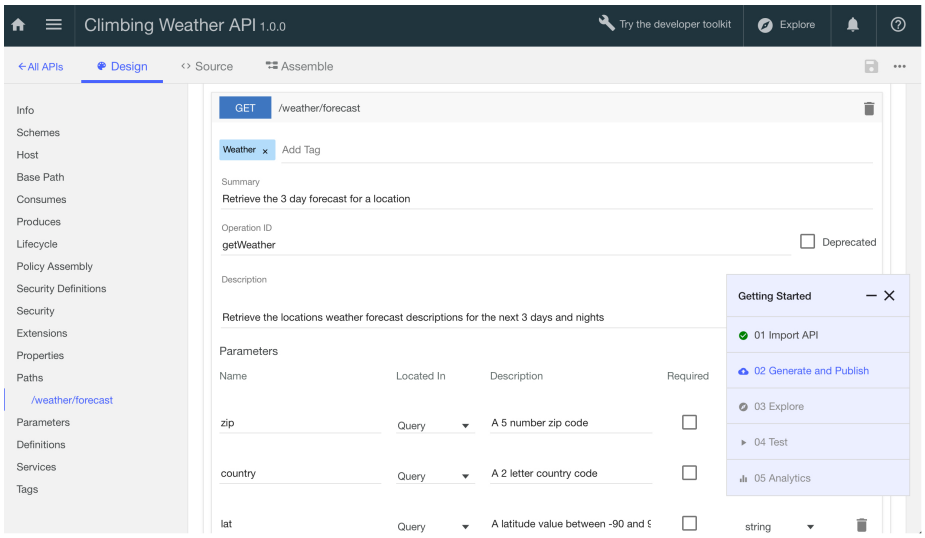

Figure 12:

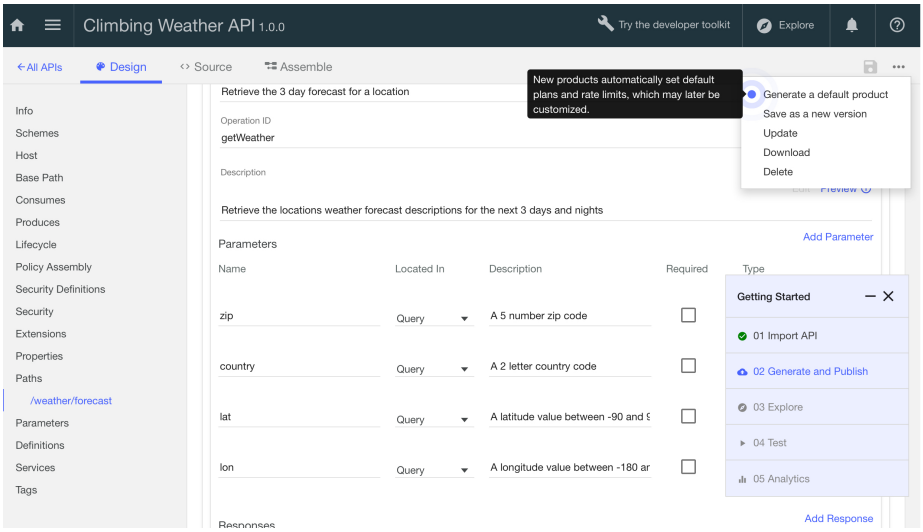

Figure 13:

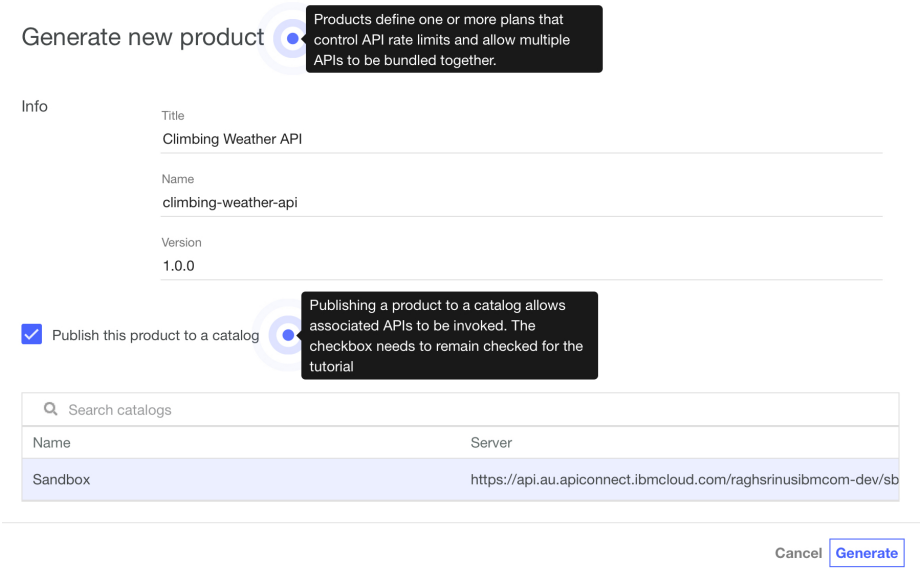

Figure 14:

Next, click “Explore” in the “Getting Started” window as shown below.

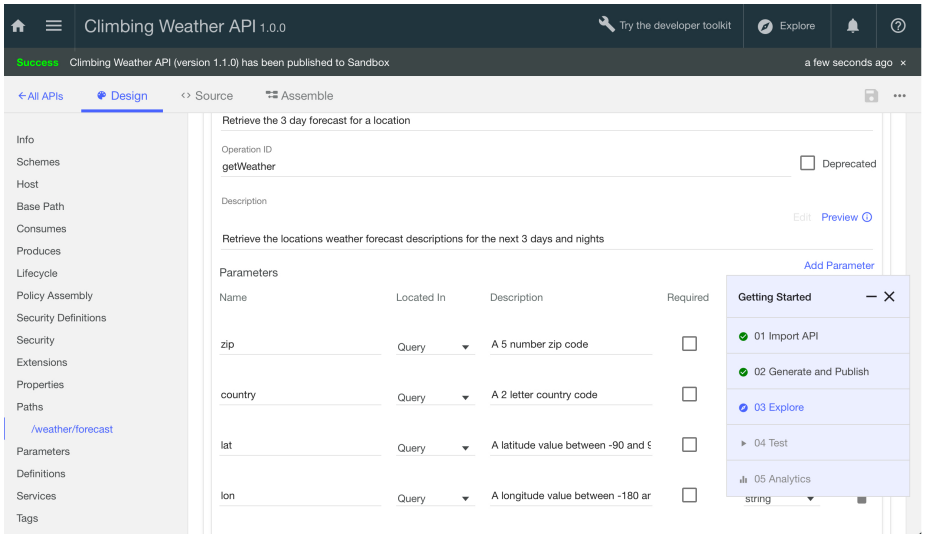

Figure 15:

Pick “Sandbox” as prompted and as whown below.

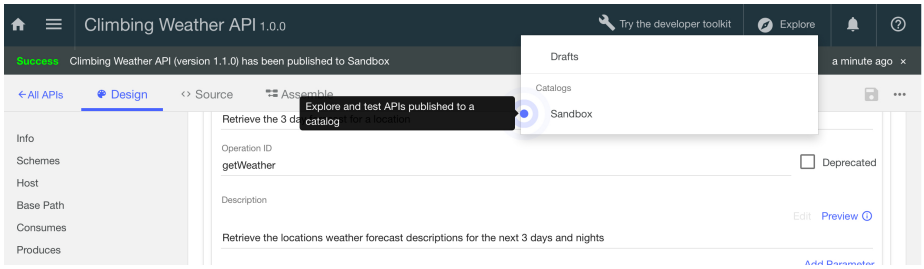

Figure 16:

You will see the sample API as shown below.

We will invoke the API by providing the following APIs to country code and zip. Feel free to substitute values.

If the call succeeds, you should see a response that looks like below with the weather data for the country and zip code.

If the call fails as shown below, click on the link and accept the certificates (we’re overriding the Cross-Origin Resource Sharing - CORS error)

You can click “Test” on the “Getting Started” window and finally on “Analytics” which should result in something like below.

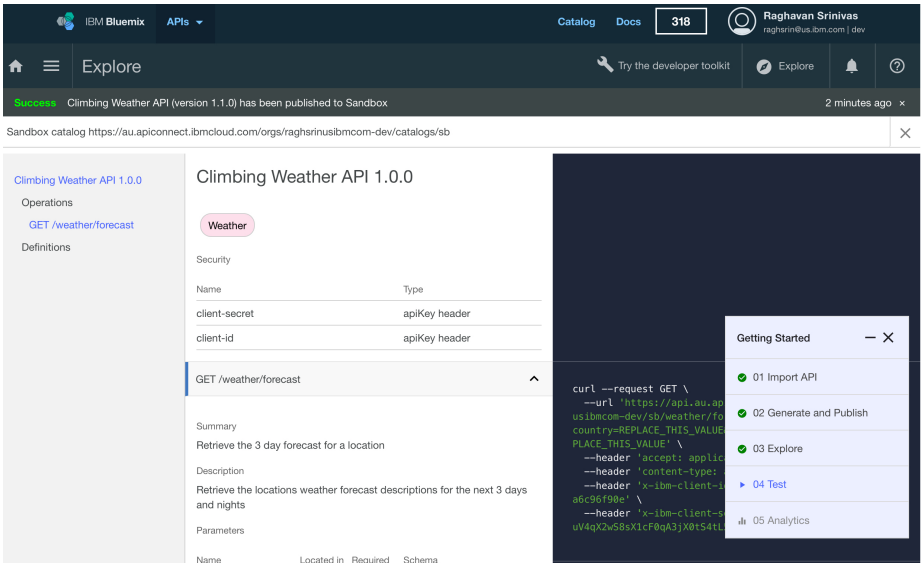

Figure 17:

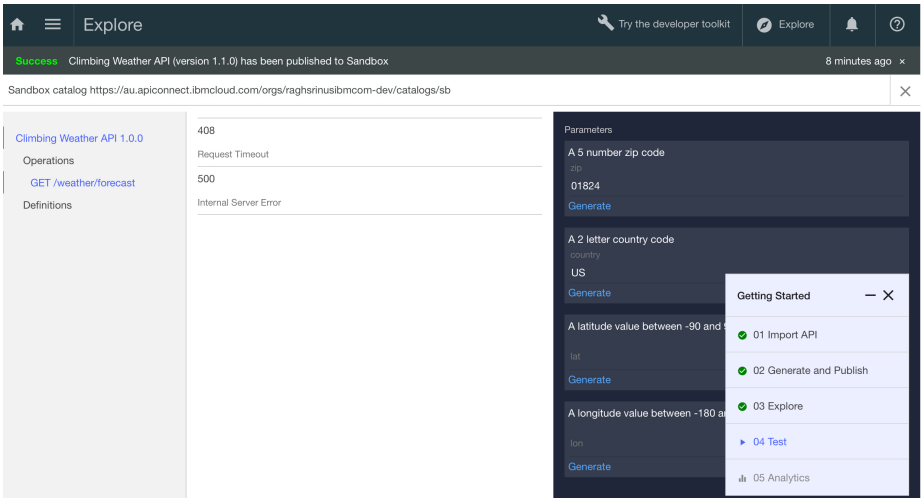

Figure 18:

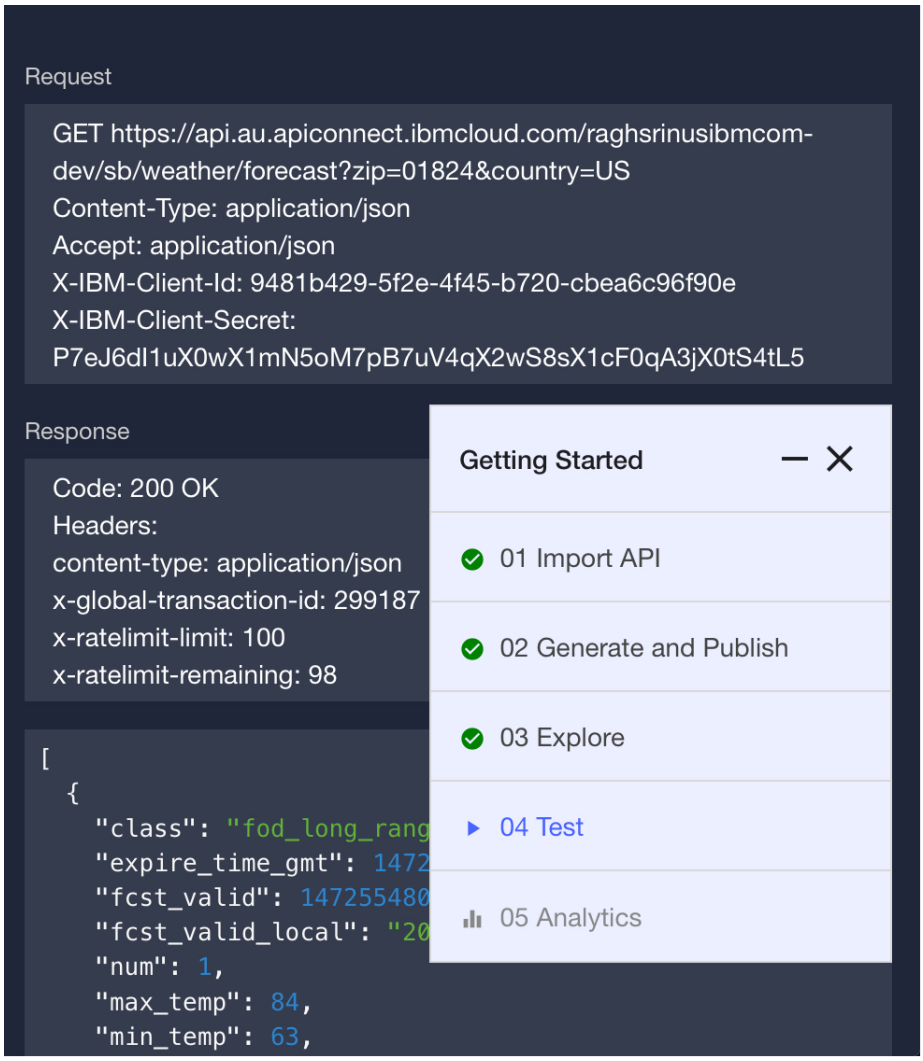
Request
GET https://api.au.apiconnect.ibmcloud.com/raghsrinusibmcom-dev/sb/weather/forecast?zip=01824&country=US
Content-Type: application/json
Accept: application/json
X-IBM-Client-Id: 9481b429-5f2e-4f45-b720-cbea6c96f90e
X-IBM-Client-Secret: P7eJ6dI1uX0wX1mN5oM7pB7uV4qX2wS8sX1cF0qA3jX0tS4tL5
Response
Code: 200 OK
Headers:
content-type: application/json
x-global-transaction-id: 299187
x-ratelimit-limit: 100
x-ratelimit-remaining: 98
[
{
"class": "fod_long_rang
"expire_time_gmt": 1472
"fcst_valid": 147255480
"fcst_valid_local": "20
"num": 1,
"max_temp": 84,
"min_temp": 63,
Getting Started
01 Import API
02 Generate and Publish
03 Explore
04 Test
05 Analytics

Figure 19:

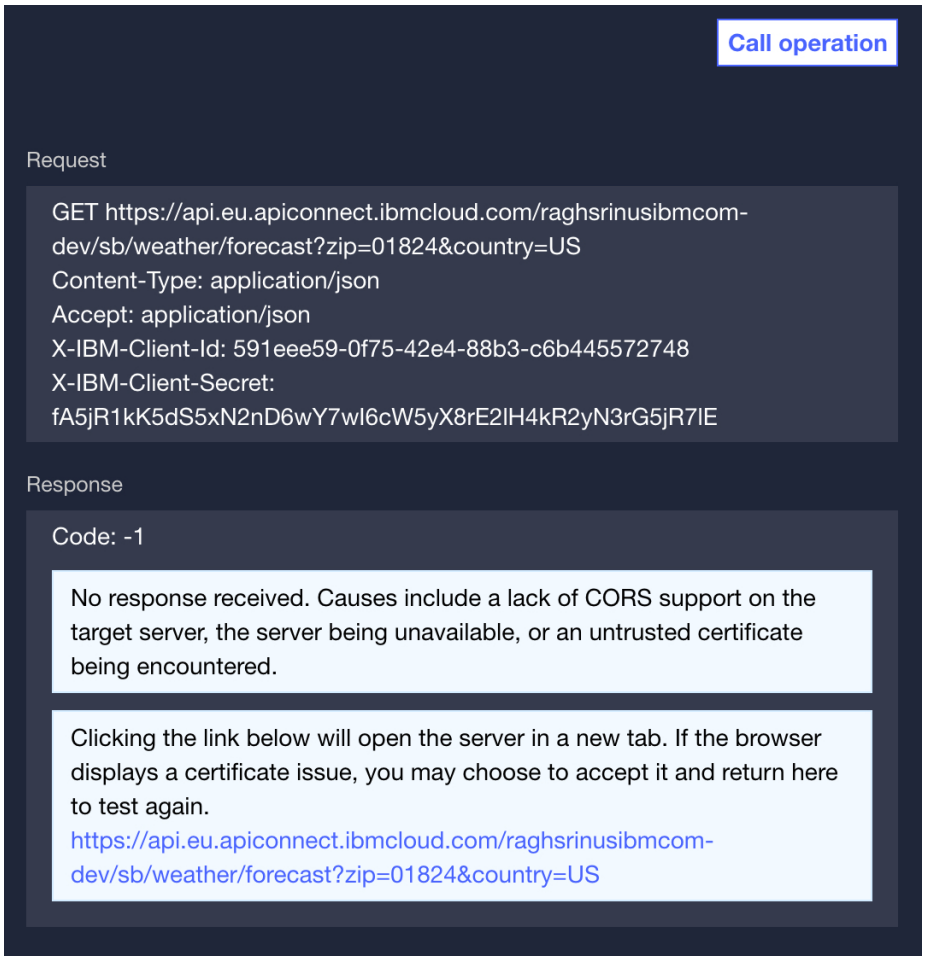
Call operation
Request
GET https://api.eu.apiconnect.ibmcloud.com/raghsrinusibmcom-dev/sb/weather/forecast?zip=01824&country=US
Content-Type: application/json
Accept: application/json
X-IBM-Client-Id: 591eee59-0f75-42e4-88b3-c6b445572748
X-IBM-Client-Secret: fA5jR1kK5dS5xN2nD6wY7wI6cW5yX8rE2lH4kR2yN3rG5jR7lE
Response
Code: -1
No response received. Causes include a lack of CORS support on the target server, the server being unavailable, or an untrusted certificate being encountered.
Clicking the link below will open the server in a new tab. If the browser displays a certificate issue, you may choose to accept it and return here to test again.
https://api.eu.apiconnect.ibmcloud.com/raghsrinusibmcom-dev/sb/weather/forecast?zip=01824&country=US

Figure 20:

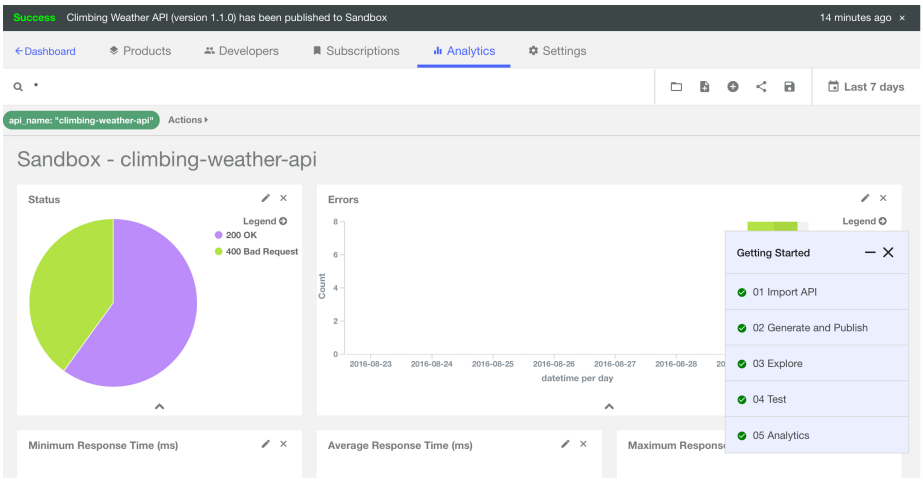

Figure 21:

Which should show the usage statistics for the product.

Summary of Exercise

We started with the API connect product locally and looked at how to use the same service on Bluemix via the Sample. In a later exercise, we will publish the product to a developer portal that is an adjunct service with the API connect service on Bluemix.

API Connect Hands-On Labs

Exercise 4: Generate a LoopBack application and import your APIs

Prerequisites

Make sure you've met the following prerequisites.

Prerequisite 1: Registered for a Bluemix account that is **still current** (trial Bluemix accounts are available at http://console.ng.bluemix.net). Contact the instructor for a promotion code for a bump in the quota. Please note down the `username` (or `email`) and `password` which will be used to login via the `cf` CLI.

Prerequisite 2: Installed the Cloud Foundry CLI from https://github.com/cloudfoundry/cli#downloads.

Prerequisite 3: Installed `npm` and `apic`. Refer to instructions from [https://nodejs.org/en/download/] (https://nodejs.org/en/download/) and [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect) respectively.

Prerequisite 4: Installed the Hands-On Labs locally. You can either `git clone` [https://github.com/ragsns/apichol] (https://github.com/ragsns/apichol) or download a zip from the repository.

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex4.

```
cd <path-to-hol-folder>/apichol/exercises/ex4
```

Goals

For this exercise, we'll:

1. Learn about how to create a LoopBack application
2. Learn how to consume an OpenAPI specification to empower your LoopBack application

LoopBack API

Navigate (cd loopback) to the empty loopback folder within the ex4 directory and execute the following command line syntax:

```
apic loopback
```

You'll want to give your application a name and select the **empty-server** option to create an empty LoopBack API.

Next, we'll learn how to easily create a Bluemix MySQL service instance.

API Connect Hands-On Labs

Exercise 5: Create a database service on Bluemix

Prerequisites

Make sure you've met the following prerequisites.

Prerequisite 1: Registered for a Bluemix account that is **still current** (trial Bluemix accounts are available at http://console.ng.bluemix.net). Contact the instructor for a promotion code for a bump in the quota. Please note down the `username` (or `email`) and `password` which will be used to login via the `cf` CLI.

Prerequisite 2: Installed the Cloud Foundry CLI from https://github.com/cloudfoundry/cli#downloads.

Prerequisite 3: Installed `npm` and `apic`. Refer to instructions from [https://nodejs.org/en/download/] (https://nodejs.org/en/download/) and [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect) respectively.

Prerequisite 4: Installed the Hands-On Labs locally. You can either `git clone` [https://github.com/ragsns/apichol] (https://github.com/ragsns/apichol) or download a zip from the repository.

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex5.

```
cd <path-to-hol-folder>/apichol/exercises/ex5
```

Verify your Target within Bluemix instance

Verify your Target within Bluemix Cloud Foundry instance by issuing the following command.

```
cf target
```

The output for the `cf` CLI should look something like below.

```
API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           <bluemix_email_id>
Org:            <bluemix_email_id>
Space:          dev
```

Creating a database service

IBM Bluemix offers a wide array of data service options that are just a few clicks/commands away.

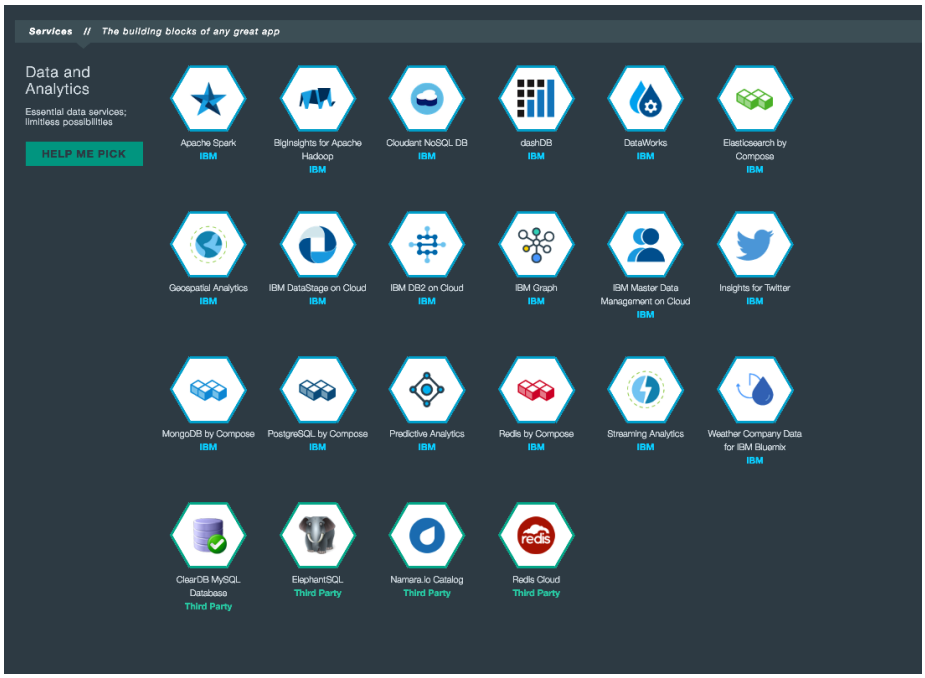

Figure 1: Service catalog

For this exercise, we'll be instantiating a MySQL database service offered on the Bluemix platform. MySQL is an open-source relational database management system (RDBMS).

Creating a Bluemix service instantiation requires input of three fields:

- Name of the service offering
- Plan of the service offering
- Instance name of your instantiated service

Name of the service offering: Defined by the service provider when registering their service with Bluemix.

Plan of the service offering: Defined by the service provider based on varying levels of quality of service (QoS), resource throttling and price.

Instance name: Defined by the user. This will be the canonical name that uniquely identifies this service instantiation. It is typically used when defining

service dependencies within application manifests or Bluemix CLI operations such as service binding.

Identifying the name and plan of a service offering

Cloud Foundry offers a listing of the entire service marketplace registered within the platform. Beware, this can be a **lengthy query** (~2-5 mins) proportional to the size of the platform catalog. For the curious, the command is:

```
cf marketplace
```

The output for the `cf` CLI would look something like below.

```
Getting services from marketplace in org xxx@xxx.xxx / space dev as xxx@xxx.xxx ...
OK

service                                          plans
APIConnect                                       Essentials, Professional*, Enterprise*, Pro
AdvancedMobileAccess                             Gold*, Bronze*
Application Security on Cloud                    free, standard*
Auto-Scaling                                     free

[....]
```

As you can see, the first two columns contain 2 out of 3 of parameters that we need. In particular, there are two MySQL services of interest to us:

- **Cleardb**
- **MySQL**

While the marketplace command contains both names and plans, there may be times where you know the name of the service and simply want to enumerate the latest plan options. If so, there is a convenient shortcut CLI command to discover only the associated plans. **Nota Bene: This convenience does not extend to services designated as experimental within the catalog**

For example, the command to discover the Cleardb plans are:

```
cf marketplace -s cleardb
```

The output for the `cf` CLI should look something like below.

```
Getting service plan information for service cleardb as xxx@xxx.xxx...
OK

service plan   description
spark          Great for getting started and developing your apps
boost          Best for light production or staging your applications
shock          Designed for apps where you need real MySQL reliability, power and throughput
amp            For apps with moderate data requirements
```

Based on the above commands, we can now discern the key details for both MySQL service providers:

Service Provider	Service Name	Service Plan (Free)
ClearDB	cleardb	spark
MySQL (Community)	mysql	100

We are now ready to create our MySQL service instance by issuing either of the following commands:

```
cf create-service cleardb spark workshopmysql
```

OR

```
cf create-service mysql 100 workshopmysql
```

The output for the `cf` CLI should look something like below.

```
Creating service instance workshopmysql in org xxx@xxx.xxx / space dev as xxx@xxx.xxx...
OK
```

Congratulations, you now have a MySQL service instance exclusively for your application and API development needs. At this point, you may be wondering about discovery of connection information to access this service. We need to create a service key credential set for our newly minted MySQL service. We can do this by issuing the following command:

```
cf create-service-key workshopmysql connectioncreds
```

The output for the `cf` CLI should look something like below.

```
Creating service key connectioncreds for service instance workshopmysql as xxx@xxx.xxx...
OK
```

Finally, we can examine the generated credentials by executing the following command:

```
cf service-key workshopmysql connectioncreds
```

The output for the `cf` CLI should look something like below.

```
{
 "host": "192.168.255.255",
 "hostname": "192.168.255.255",
 "name": "dc9caxxxxxxxxxxxxxxxxxxe4ec",
 "password": "p9c#&$^&*@TN",
 "port": 3307,
 "uri": "mysql://uevWFw0FThyL9:p9c#&$^&*@TN@192.168.255.255:3307/dc9caxxxxxxxxxxxxxxxxxxe4ec
 "user": "uevWFw0FThyL9",
 "username": "uevWFw0FThyL9"
}
```

Armed with this connection information, we are now able to interact with our new MySQL server.

In the next exercise, we will dive into creation of database CRUD APIs.

API Connect Hands-On Labs

Exercise 6: Create database CRUD APIs with LoopBack models

Prerequisites

To run through this exercise, you will need to have done the following steps:

Prerequisite 1 Installed the API Connect toolkit (Exercise 2)

Prerequisite 2 Generated a LoopBack app (Exercise 4)

Prerequisite 3 Created a database service on Bluemix and connected it to your LoopBack app (Exercise 5)

Ensure that you are in the LoopBack application directory

Ensure that you are in the LoopBack directory you created in Exercise 4

```
cd <path-to-loopback-folder>
```

Launch the API Connect Designer (Developer toolkit)

The API Connect Designer is a GUI that allows developers to graphically create and manage their APIs.

```
apic edit
```

After a brief pause, the following message is displayed.

```
Express server listening on http://127.0.0.1:9000
```

The API Designer opens in your default web browser. If it prompts you to login, use your IBM Bluemix credentials.

Create a database connection

Click on the Data Sources tab. Hit the "Add" button, and choose name for your database - "mysql-db".

In the connector tab, choose "MySQL". It'll prompt you to install the connector; simply follow the prompts.

Enter the database credentials you noted in Exercise 5. Enter the `uri` credential into the `url` field and your database name, `employees` into the `Database` field.

Hit the Save button on the top-right. This should test your database connection and alert you if your credentials are incorrect or if the connection was unable to be made.

Create Models to work with your database

Create an "Employee" model so that you're able to perform CRUD (Create/Read/Update/Delete) operations against your MySQL database.

TODO: Add screenshot.

That's it! Once a model is created, the APIs to represent that model are automatically generated for you.

Next steps

In the next exercise, we will test your new APIs by starting the LoopBack application locally and use an interactive OpenAPI explorer to call your APIs!

Next up, Exercise 7: Test, Explore and Deploy your LoopBack application

API Connect Hands-On Labs

Exercise 7: Test, Explore and Deploy your LoopBack application

Prerequisites

To run through this exercise, you will need to have done the following steps:

Prerequisite 1 Installed the API Connect toolkit (Exercise 1)

Prerequisite 2 Generated a LoopBack app (Exercise 4)

Prerequisite 3 Created a database service on Bluemix and connected it to your LoopBack app (Exercise 5)

Prerequisite 4 Created database CRUD APIs in the API Designer (Exercise 6)

Ensure that you are in the LoopBack application directory

Ensure that you are in the LoopBack directory you created in Exercise 2

```
cd <path-to-loopback-folder>
```

Launch the API Connect Designer (Developer toolkit)

The API Connect Designer is a GUI that allows developers to graphically create and manage their APIs.

```
apic edit
```

After a brief pause, the following message is displayed.

```
Express server listening on http://127.0.0.1:9000
```

The API Designer opens in your default web browser. If it prompts you to login, use your IBM Bluemix credentials.

Start your LoopBack application

On the bottom left of the API Designer, hit the `Play` button to start your application. After a short delay, your application will change to "Running", and you should see two links: Micro Gateway and Application.

TODO: Add screenshot

The application link corresponds to the LoopBack application you created in the earlier exercises. It hosts the CRUD APIs you created in exercise 6. The Micro Gateway link corresponds to a fully-featured API gateway which proxies requests to your LoopBack application, allowing you test your gateway policies.

Test and Explore your Swagger-based APIs

Now that the application is running, let's try calling some of the APIs!

On the top right of the `API Designer`, hit the `Explore` button. This takes you to an API Explorer, allowing you to explore the APIs defined in your generated Swagger doc.

TODO: Add screenshot

Along the left side, you should see a number of operations for the `Employee` model you created in exercise 6. Let's try calling a series of these operations.

TODO: Add screenshot

GET /Employees

Navigate to the operation `GET /Employees`. Along the right side, there is a black section which shows you how to call that operation, provides boiler code, and has a button "Call Operation". Hit the button to call your GET operation.

TODO: Add screenshot

You should see a `200 OK` response, along with an empty array in the response body: `[]`. This is because you haven't stored anything in the database yet!

TODO: Add screenshot

POST /Employees

Navigate to the operation `POST /$Employees` to create a database entry. Scroll down to the "Call Operation" button, enter some data into the Paramters section (or use the `Generate` button), and hit call Operation.

TODO: Add screenshot

You should see a `200 OK` response, as well as a response body indicating that the database update has succeeded.

TODO: Add screenshot

Deploy your APIs to IBM Bluemix

Once you're happy with the APIs you've created, you can push them the Bluemix, IBM's PaaS (Platform as a Service). Bluemix will host your APIs and allow you to graphically manage them.

Start by hitting the Publish button on the top right of the API Designer.

TODO: Add screenshot

Choose `Add and Manage Targets` and `Add IBM Bluemix target`.

TODO: Add screenshot

Ensure that the organization is correct – it should correspond to your Bluemix email. Choose the `Sandbox` catalog.

TODO: Add screenshot

Type a new application name: `EmployeeAPI`. Then hit the `(+)` button, and hit `Save`.

TODO: Add screenshot

You've now created a publish target; deploy to it by hitting the `Publish` button and choosing the target you just created.

Choose both `Publish Application` and `Stage or Publish` products. Hit `Publish`. That's it! Your APIs are now securely pushed to the cloud.

TODO: Add screenshot

Next steps

In the next two exercises, we'll explain how you can test your APIs and create a developer portal so others can consume your APIs.

Next up, Exercise 8: Explore your deployed APIs with the API Manager on Bluemix

API Connect Hands-On Labs

Exercise 8: Explore your deployed APIs with the API Manager on Bluemix

Prerequisites

To run through this exercise, you will need to have done the following steps:

Prerequisite 1 Installed the API Connect toolkit (Exercise 1)

Prerequisite 2 Generated a LoopBack app (Exercise 4)

Prerequisite 3 Created a database service on Bluemix and connected it to your LoopBack app (Exercise 5)

Prerequisite 4 Created database CRUD APIs in the API Designer (Exercise 6)

Prerequisite 5 Deployed your LoopBack application to Bluemix (Exercise 7)

Accessing your deployed APIs

In exercise 7, you tested your APIs locally and deployed them to Bluemix. To see where you deployed the APIs, we'll access the API Manager you used in exercise 2.

Navigate to Bluemix: https://new-console.ng.bluemix.net and ensure you are logged in. Choose APIs using the drop-down at the top, and select your API Connect service. This launches the API Manager.

Double check that you're in the `Dashboard` view by selecting the sandwich button on the top left and clicking Dashboard.

TODO: Add screenshot

Click the `Sandbox` Catalog, where you should see the Employee API you deployed earlier.

In this view, you can edit the lifecycle of the API, control its visibility and see analytics. For now, we'll simply Explore the APIs. Hit the `Explore` button on the top right, and choose your `Sandbox` catalog.

TODO: Add screenshot

In this view, we can explore all the APIs in the chosen catalog.

GET /Employees

Navigate to the operation `GET /Employees`. Along the right side, there is a black section which shows you how to call that operation, shows a sample response, and has a button "Call Operation". Hit the button to call your GET operation.

TODO: Add screenshot

You should see a `200 OK` response, along with an empty array in the response body: `[]`. This is because you haven't stored anything in the database yet!

POST /Employees

Navigate to the operation `POST /Employees` to create a database entry. Scroll down to the "Call Operation" button, enter some data into the Paramters section (or use the `Generate` button), and hit call Operation.

TODO: Add screenshot

You should see a `200 OK` response, as well as a response body indicating that the database update has succeeded. You can now call the `GET /Employees` operation again to see the employee entry you just created.

TODO: Add screenshot

Next steps

In the next exercises, we'll create a developer portal so others can consume your APIs.

Next up, Exercise 9: Generate a Developer Portal for your APIs

API Connect Hands-On Labs

Exercise 9: Generate a Developer Portal for your APIs

Prerequisites

Make sure you've met the following prerequisites.

Prerequisite 1: Completed Exercise 2.

Overview

We will pick up from where we left off at exercise 2. We followed the "Getting Started" tour to use the API Connect manager and invoke the APIs within the API Connect manager.

In this exercise, we will publish the APIs, with an associated plan and so on to a developer portal which is an adjunct to the Bluemix API Connect service. These APIs can be invoked by any REST-based client.

Publishing the Sample API to the Developer Portal

Let's head back to the API Connect Manager on IBM Bluemix and to the sandbox that we finished off in exercise 2 as we see below. You can click on the Dashboard tab on the Bluemix console and then on the API connect service as shown below.

This will bring up the Dashboard for the API Connect manager as shown below.

Select the "Settings" tab, pick "Portal" and pick the "IBM Developer Portal". Finally click Save icon on the top right as shown below.

Wait for the email to arrive that the developer portal is ready as the dialog box below indicates.

Clicking on the link will take you to the developer portal. Reset the password for `admin` as shown below.

You will see the "Climbing Weather API" as seen below.

We will next publish this API to Bluemix.

Publishing the Sample API to Bluemix

We will go back to the API Connect Manager. If required, re-login to the API Connect Manager as shown below.

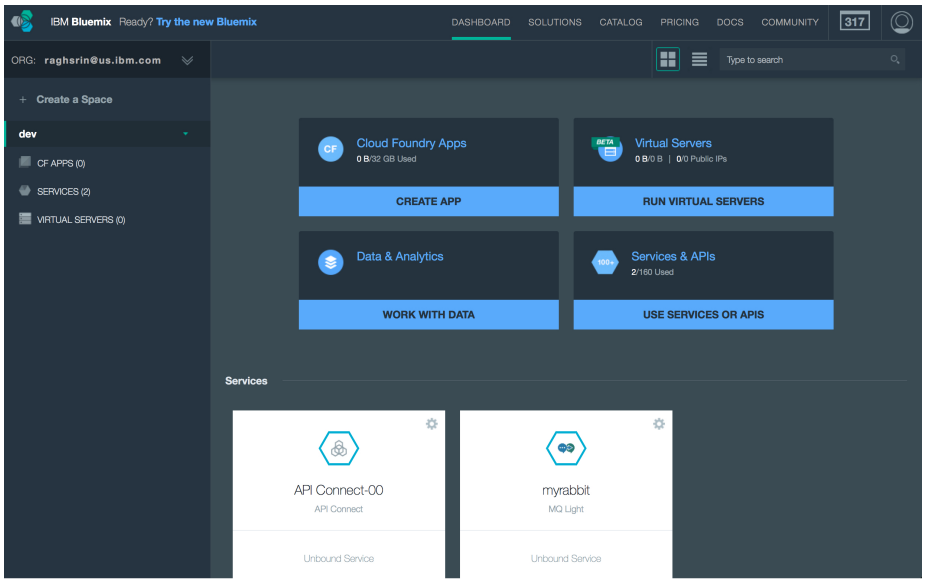

Figure 1:

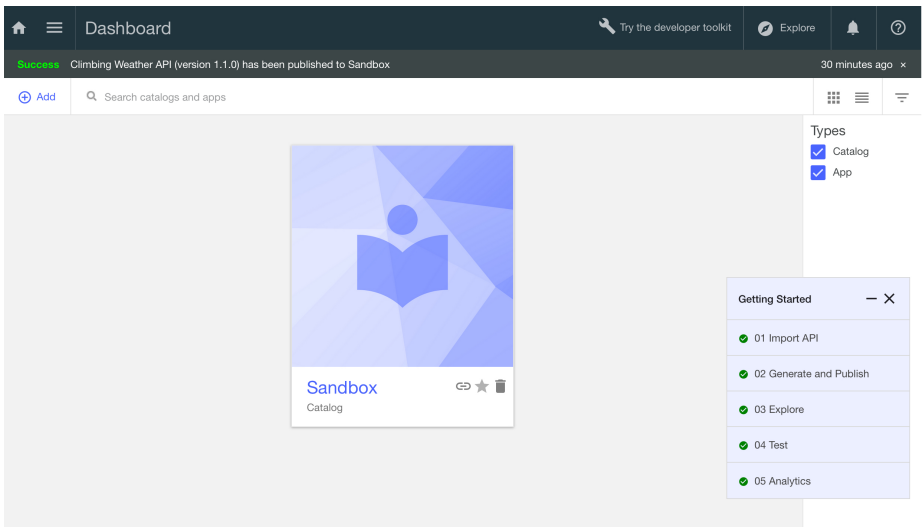

Figure 2:

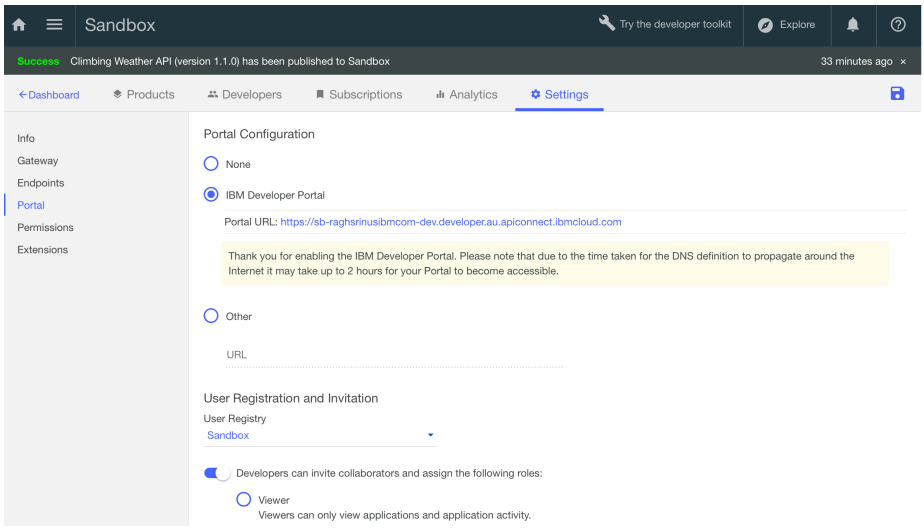

Figure 3:

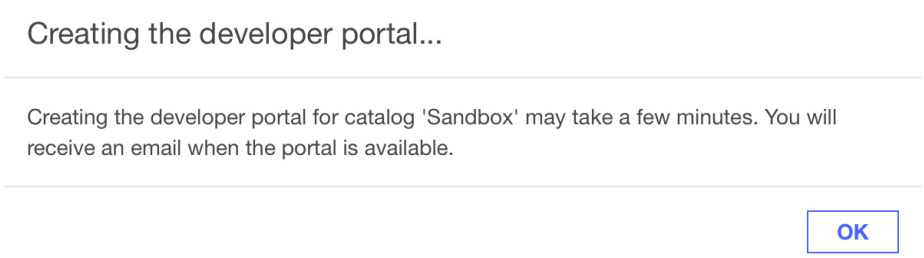

Figure 4:

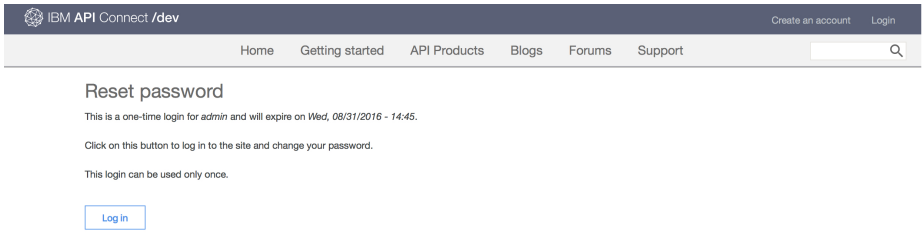

Figure 5:

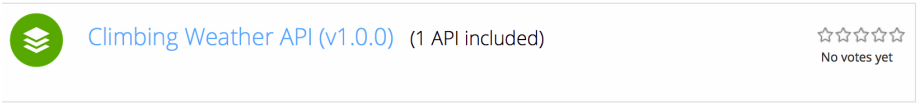

Figure 6:

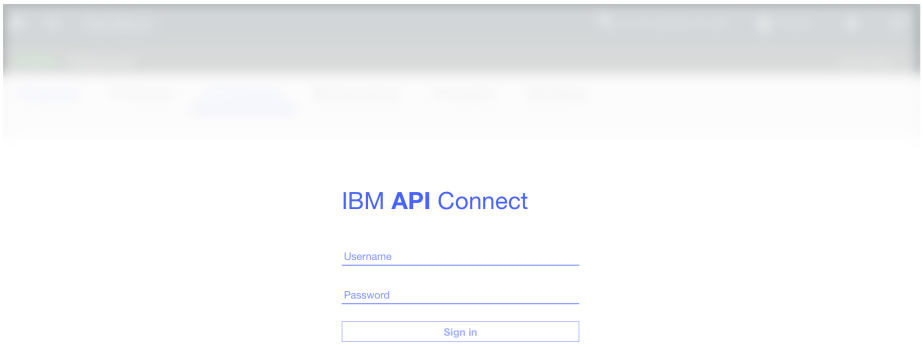

Figure 7:

In the API Connect Manager on Bluemix click on “Developers” in the navigation pane and “Add Bluemix organization” as shown below.

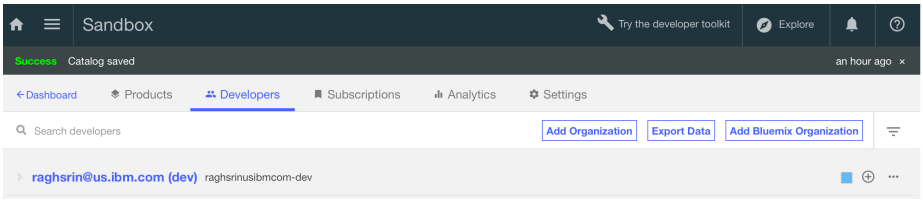

Figure 8:

Provide the email address substituting the address with your Bluemix email address as shown below.

Select the Bluemix org and hit “Confirm” as shown below.

Now back in the API Connect Manager click on the “Products” tab and click on “. . . ” tab which should bring up the popup menu as shown below and clik on “Edit visibility”.

Change the visibility to “Custom” and add the organization and Bluemix. Finally hit “Republish” as shown below.

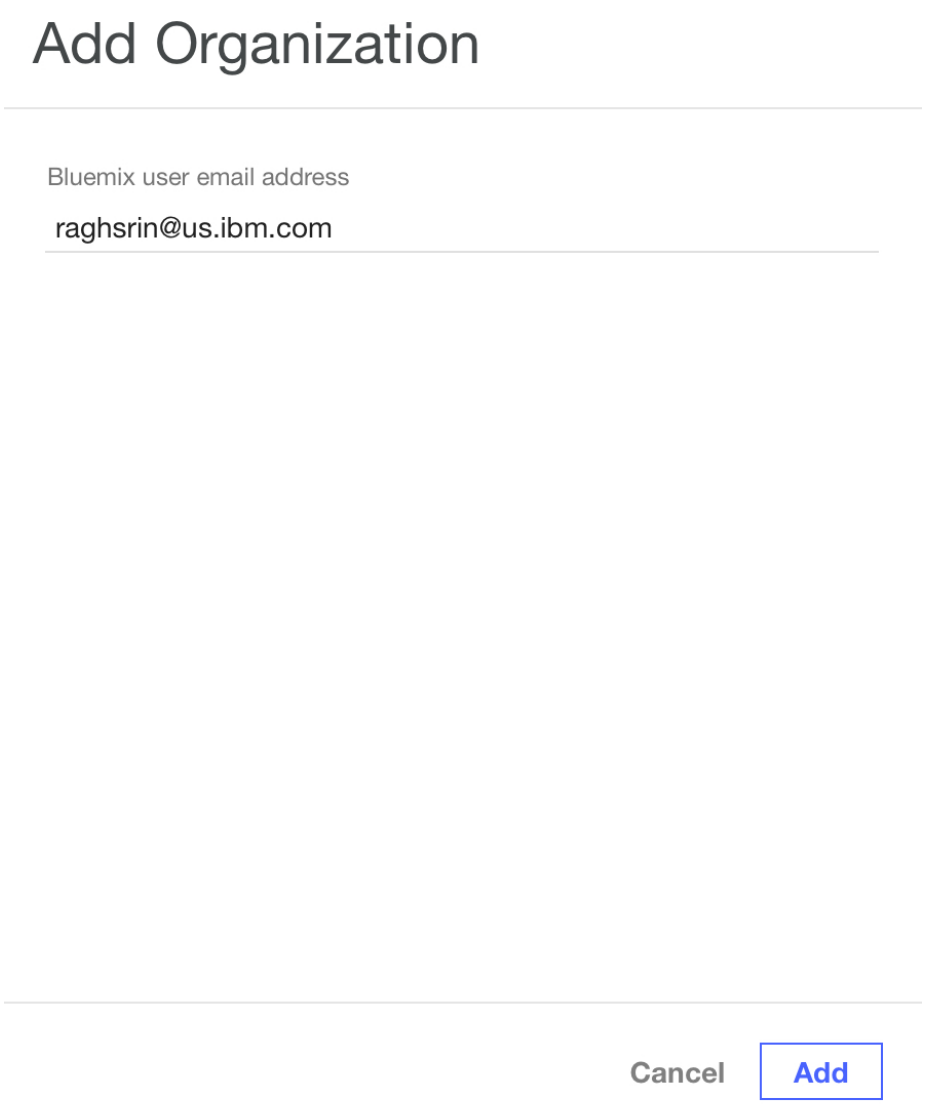
Add Organization
Bluemix user email address
raghsrin@us.ibm.com
Cancel
Add

Figure 9:

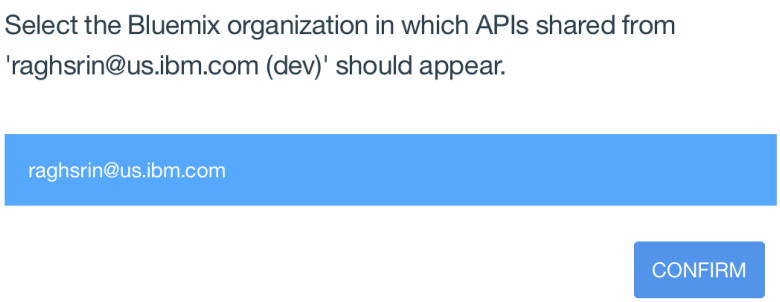

Figure 10:

The Climbing Weather API product is displayed in the Explore APIs tab of the API Connect Dashboard. If you click, Climbing Weather App 1.0.0 it will take you to the API in the Developer Portal.

Registering an app for the sample API in the Developer Portal

Back in the Developer Portal logout from admin as shown below.

Click on "Create an account" as shown below.

Complete the form and ensure that you use a different email address.

You should see an account ativation complete as shown below.

Login with the account which will bring up the portal as below.

Click the API Products and Climbing Weather API. You can see that Climbing Weather API belongs to a default Plan with a rate limit of 100 calls per minute as shown below.

Click on the "Climbing Weather API 1.0.0" tab as shown below.

Click on the "Apps"navigation pane of the Developer Portal, click "Register a new Application" as shown below.

Fill in the details and hit "Submit".

Click Show to display the "**Client secret**" that we will use subsequently as shown below.

Click "API Products" and the "Climbing Weather API" as shown below.

The details of the Plan that is named Default Plan are displayed. Click "Subscribe" as shown below.

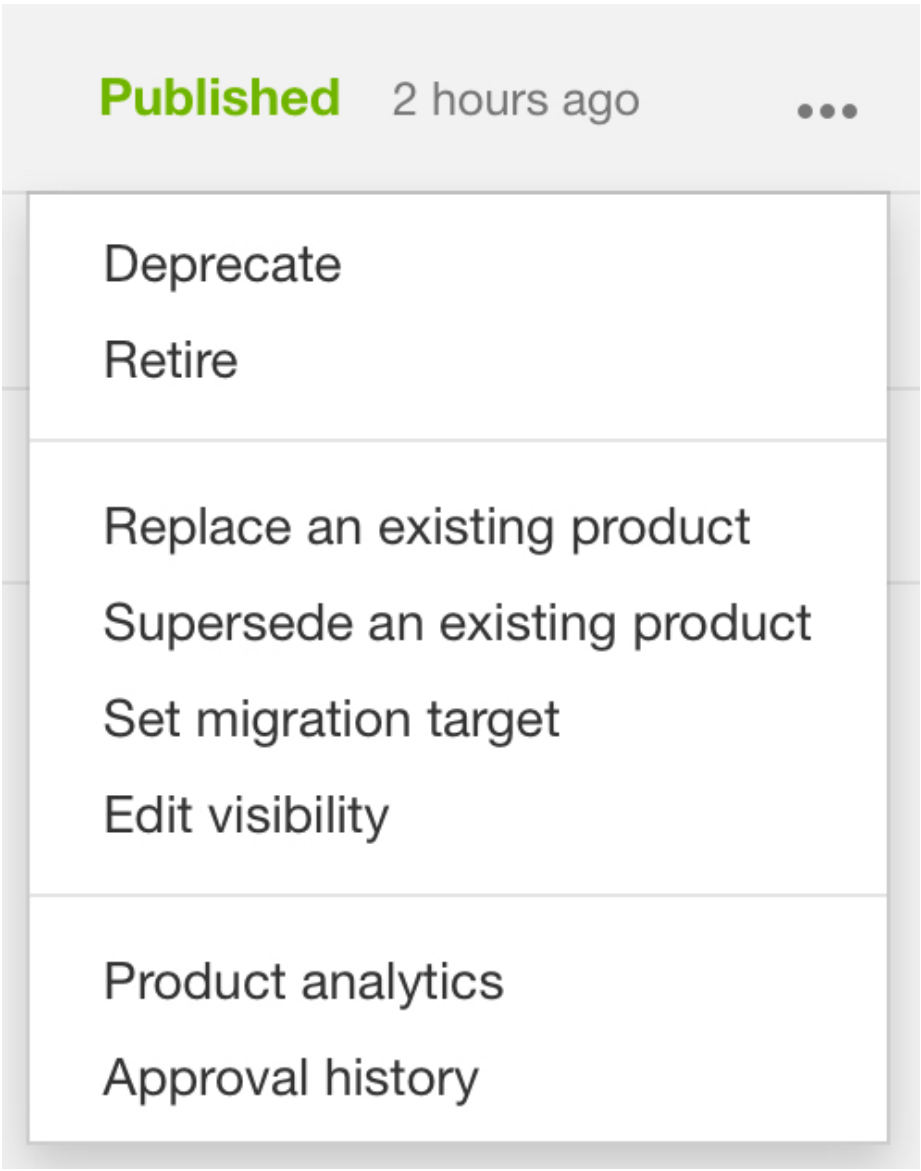
Published
2 hours ago
Deprecate
Retire
Replace an existing product
Supersede an existing product
Set migration target
Edit visibility
Product analytics
Approval history

Figure 11:

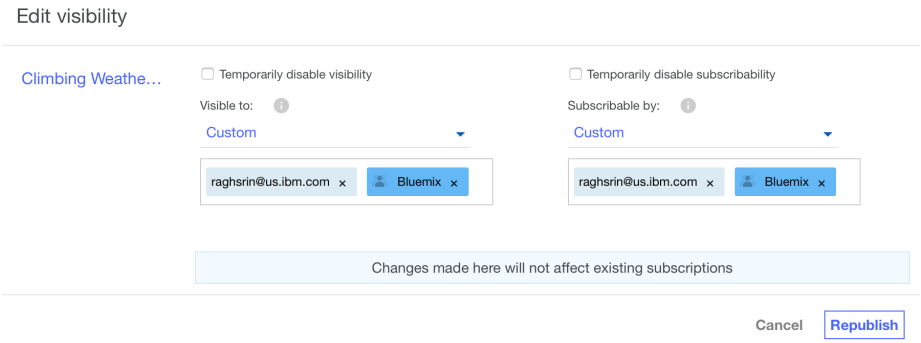

Figure 12:

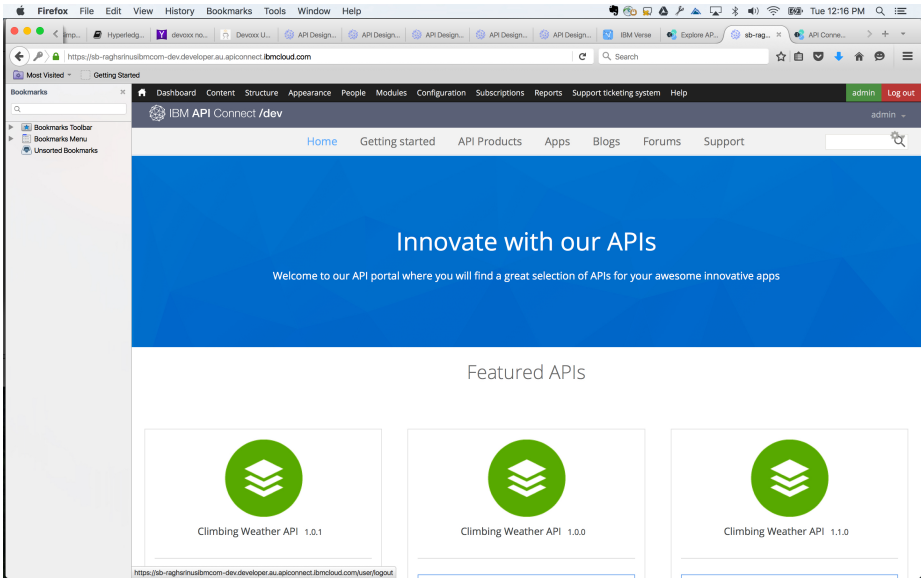

Figure 13:

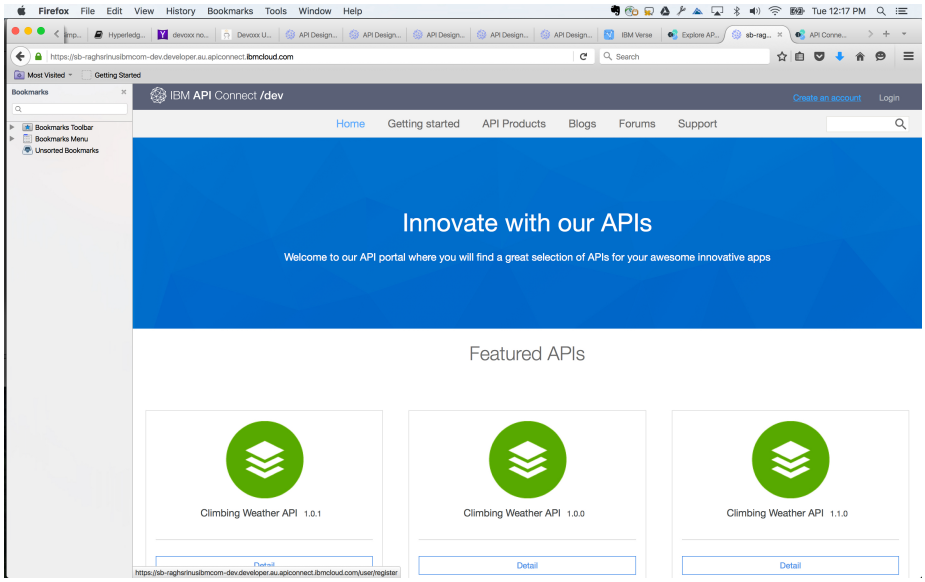

Figure 14:

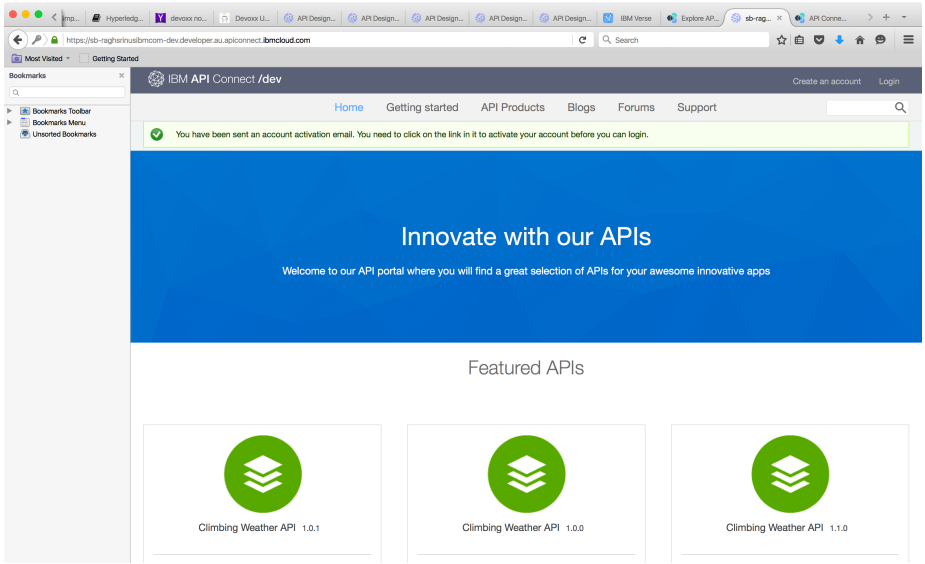

Figure 15:

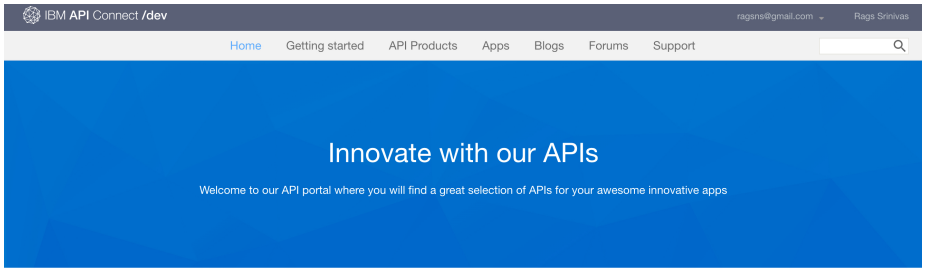

Figure 16:

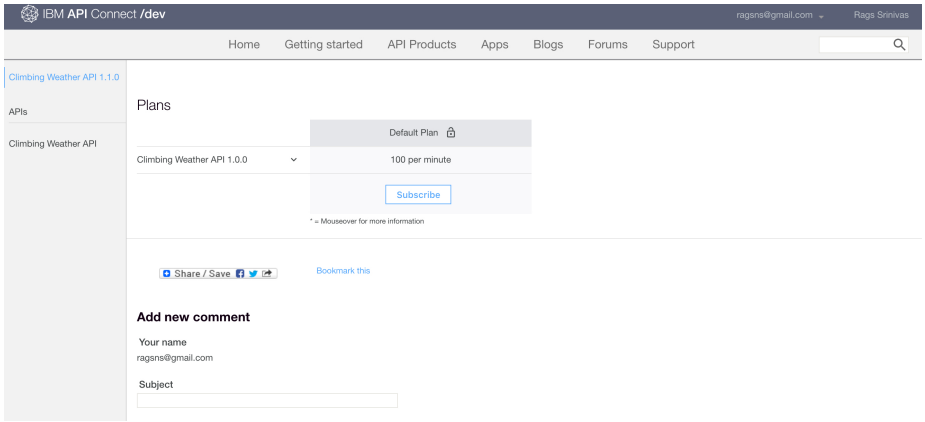

Figure 17:

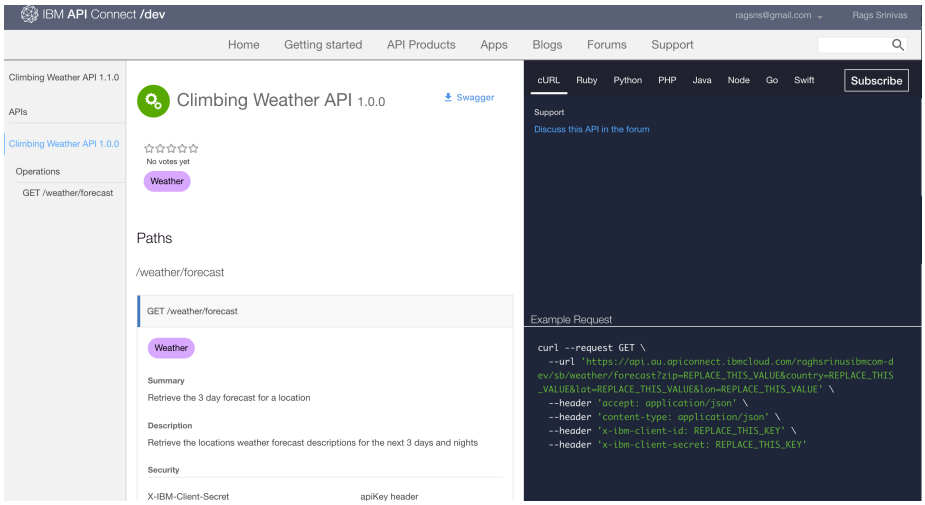

Figure 18:

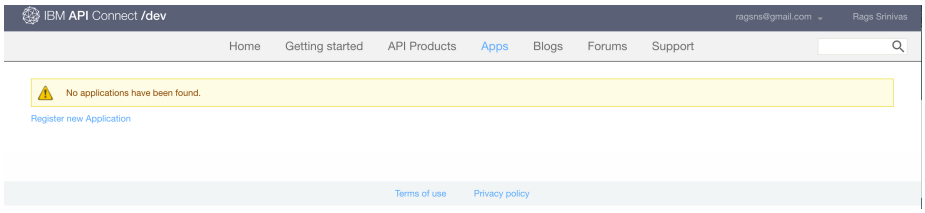

Figure 19:

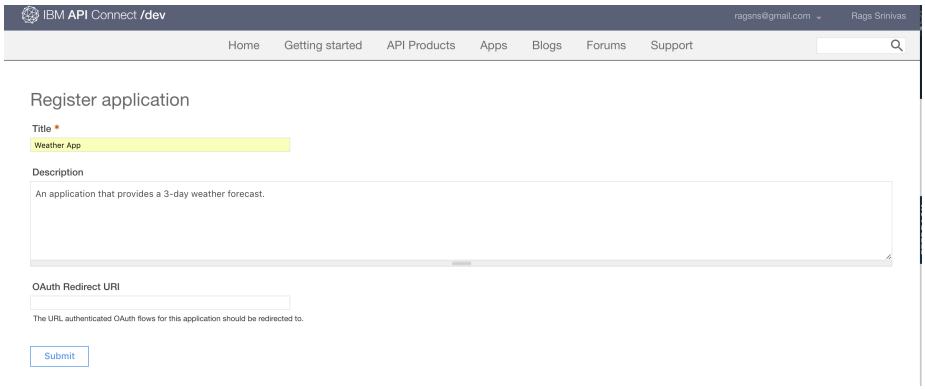

Figure 20:

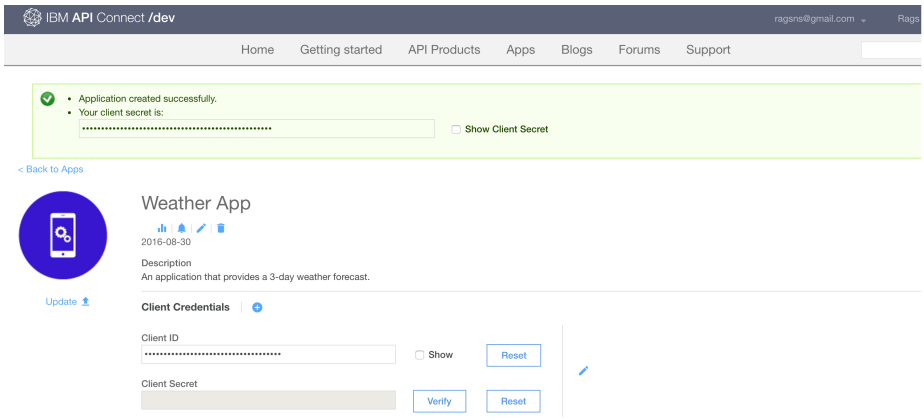

Figure 21:

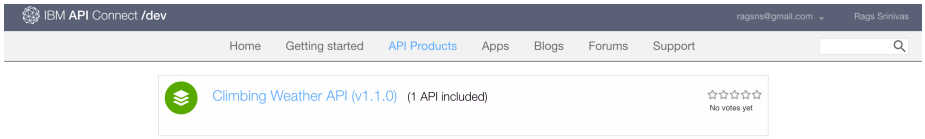

Figure 22:

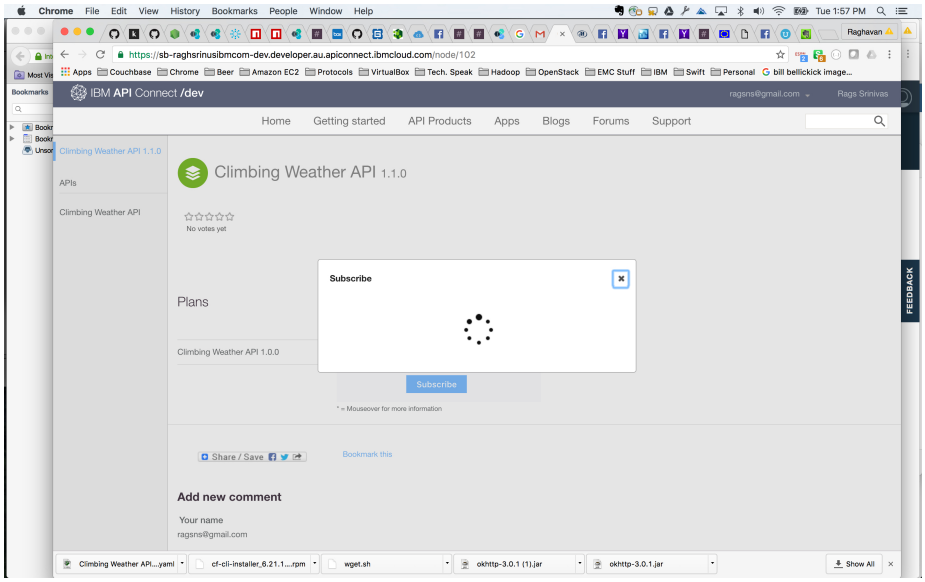

Figure 23:

This will bring up a dialog box. Click on “Subscribe” as shown below.

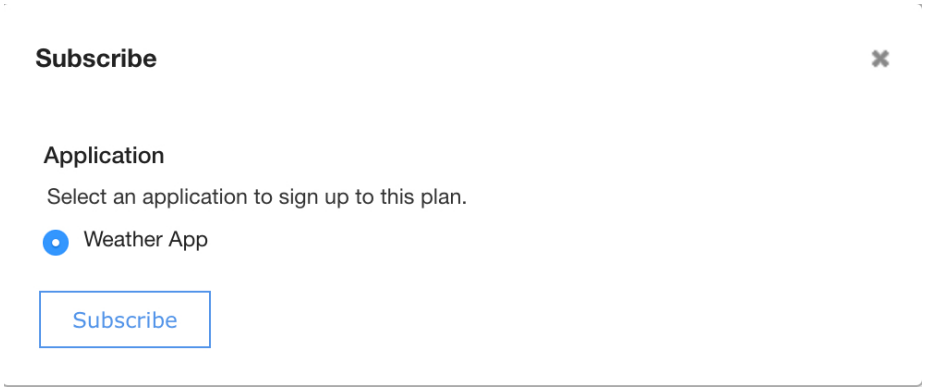

Figure 24:

You have registered to the Climbing Weather App application and subscribed it to a Plan.

Testing the Climbing Weather API in the Developer Portal

Back in the Developer Portal, In the left navigation pane, click Climbing Weather API. In the “Try this operation” section of the console, ensure that the Weather App application is selected and enter the parameters as shown below including the “Client secret” that was noted from the earlier step and click “Call Operation”.

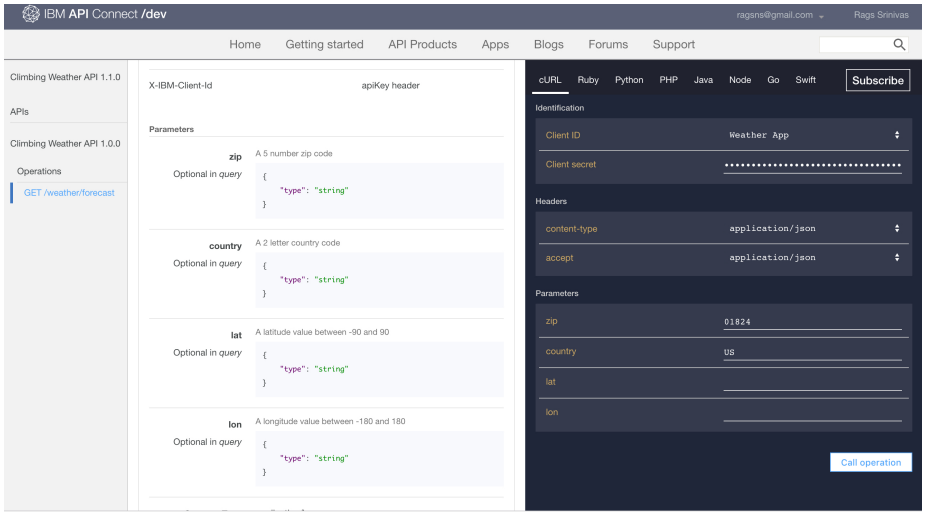

Figure 25:

That should provide a response which is shown below.

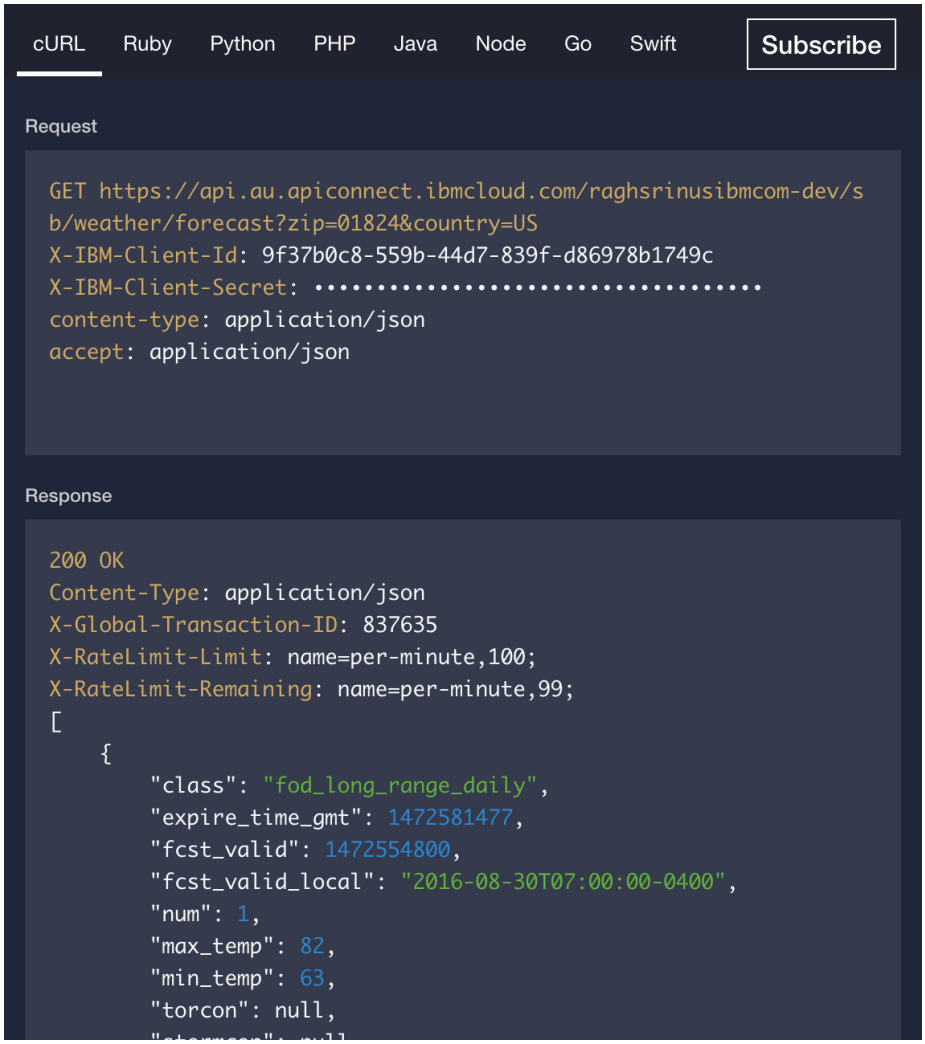

Figure 26:

Summary of Exercise

We started with the API connect product locally in exercise 1 and looked at how to use the same service on Bluemix via the Sample in exercise 2. In this exercise, we published the product to a developer portal that is an adjunct service with the API connect service on Bluemix.