API Connect Hands-On Lab

Prerequisites

Needless to say you'll need a laptop! It is recommended you use a Mac OS or a ubuntu desktop version laptop. Also make sure you install the following software prior to the session:

You can verify the versions of the required or pre-installed software by running the following commands and ensuring that you have the following versions (or higher).

```
git --version
git version 2.7.4
```

```
cf --version
cf version 6.21.1+6fd3c9f-2016-08-10
```

```
node --version
v4.5.0
```

```
npm --version
3.10.7
```

```
apic --version
API Connect: v5.0.3.0 iFix 2
```

The software can also be installed from

- Git from http://git-scm.com/downloads 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 - Install `nodejs` from https://nodejs.org/en/download/ which also installs `npm`.
- Install API connect Developer kit - Install API connect Developer Kit after installing `npm` from [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect)

Sign up for a Bluemix account

- **Sign up for a new account on a Bluemix hosted instance** - It is recommended you create a new account from [https://console.ng.bluemix.net/]

(https://console.ng.bluemix.net/) especially if you have not created this account in the last few days.

Lab software

The software including the instructions is available from [https://github.com/ragsns/apichol] (https://github.com/ragsns/apichol). Install the software locally on your laptop by running the followng command.

```
git clone https://github.com/ragsns/apichol
```

If the lab software has already been installed (you should be able to see a subdirectory named `apichol`) ensure that you have the latest updates by issuing the following command.

```
cd apichol
git pull
```

Samples and General Directions

Each exercise 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 at Twitter @ragss or @boilerupnc or @sai_vennam.

API Connect Hands-On Labs

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

Prerequisites

It is recommended you use a Mac OS or a ubuntu desktop version laptop. Also make sure to install the following software prior to the session:

You can verify the versions of the required or pre-installed software by running the following commands and ensuring that you have the following versions (or higher).

```
git --version
git version 2.7.4
```

```
cf --version
cf version 6.21.1+6fd3c9f-2016-08-10
```

```
node --version
v4.5.0
```

```
npm --version
3.10.7
```

```
apic --version
API Connect: v5.0.3.0 iFix 2
```

The software can also be installed (if you Bring Your Own Laptop) from

- Git from http://git-scm.com/downloads 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 - Install `nodejs` from https://nodejs.org/en/download/ which also installs `npm`.
- Install API connect Developer kit - Install API connect Developer Kit after installing `npm` from [https://www.npmjs.com/package/apiconnect] (https://www.npmjs.com/package/apiconnect)

Sign up for a Bluemix account

Sign up for a new account on a Bluemix hosted instance - It is recommended you create a new account from https://console.ng.bluemix.net/ especially if you have not created this account in the last few days.

Lab software

The software including the instructions is available from https://github.com/ragsns/apichol. Install the software locally on your laptop by running the followng command.

```
git clone https://github.com/ragsns/apichol
```

If the lab software has already been installed (you should be able to see a subdirectory named `apichol`) ensure that you have the latest updates by issuing the following command.

```
cd apichol
git pull
```

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex1.

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

Overview of exercise

In this exercise, we will target a Bluemix Cloud Foundry instance using the Bluemix ID you created and invoke 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 dev a
OK
Assigning role SpaceDeveloper to user raghsrin@us.ibm.com in org raghsrin@us.ibm.com / space de
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 at signup> -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 examp
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 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 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 and next steps

After installing all the prerequisites, we explored how to target a Bluemix/Cloud Foundry instance and use the API connect product locally.

Next we will look into how to use the API connect Manager on Bluemix which provide an interface similar to what we saw locally. We will go through the `Getting Started` tour which will give us a good idea of the different artifacts and how they are connected in Exercise 2.

API Connect Hands-On Labs

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

Prerequisites

Prerequisite 1: Installed the API Connect toolkit (Exercise 1).

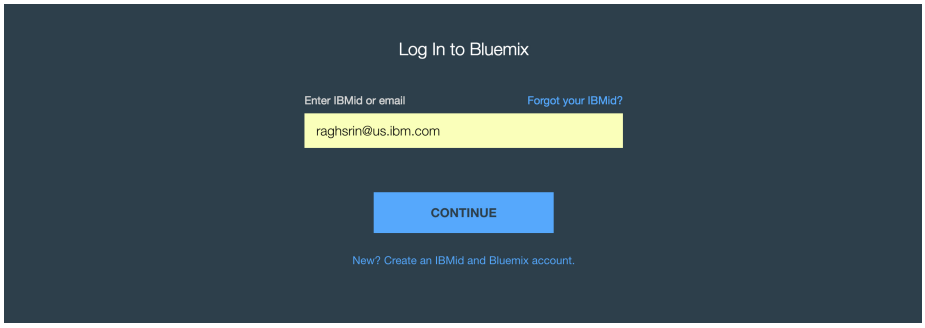
Overview of exercise

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, but on the cloud.

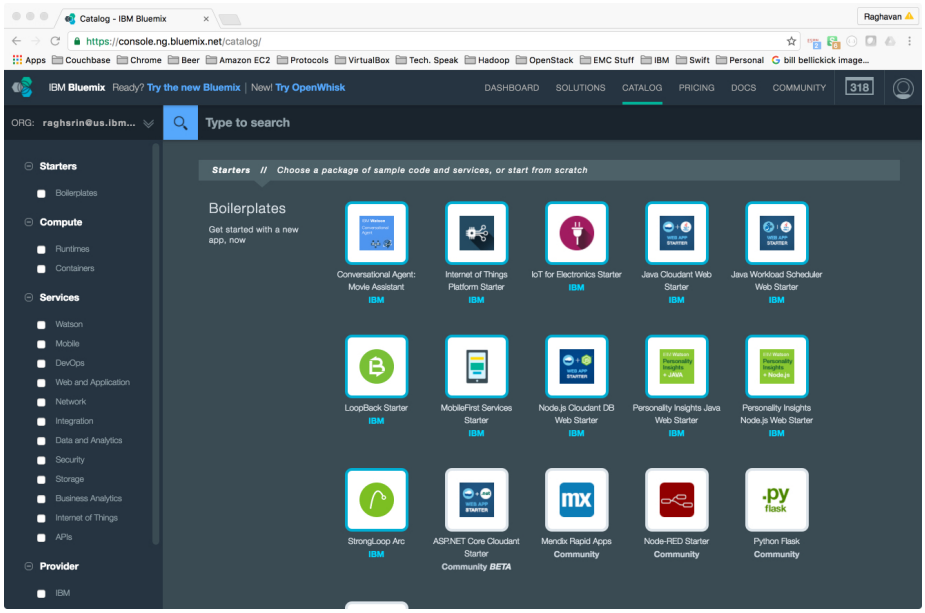
The service also enables publication of the APIs to a developer portal which will be covered in a subsequent exercise.

Login to Bluemix and instantiate the API Connect Service

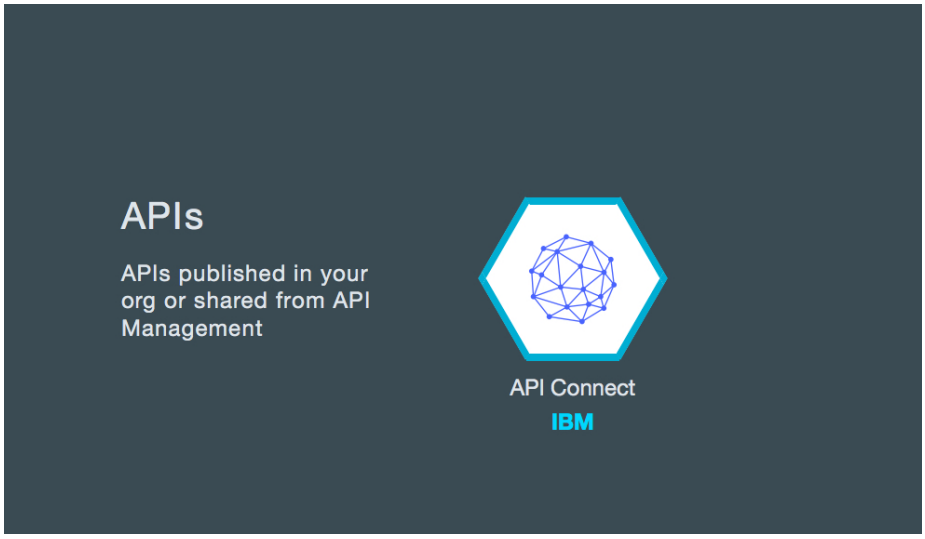
Login to Bluemix by providing the credentials that you used during the registration process.



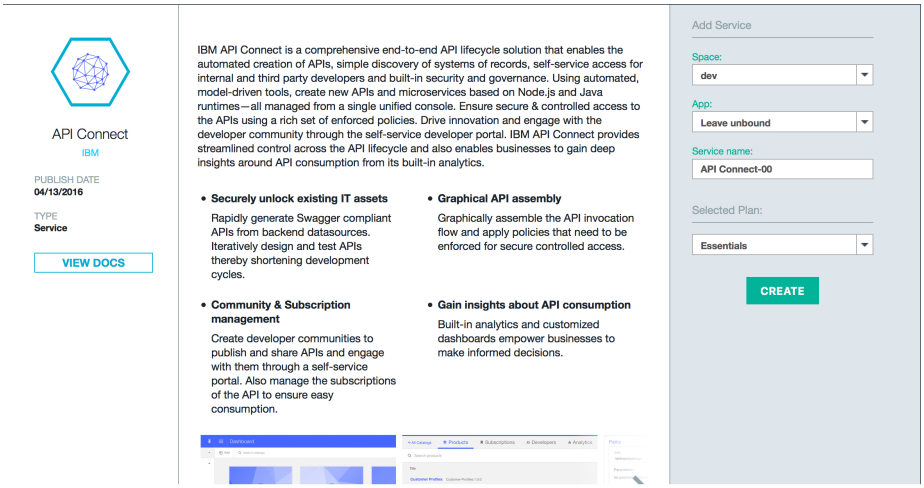
Once logged in, click on `Catalog` tab.



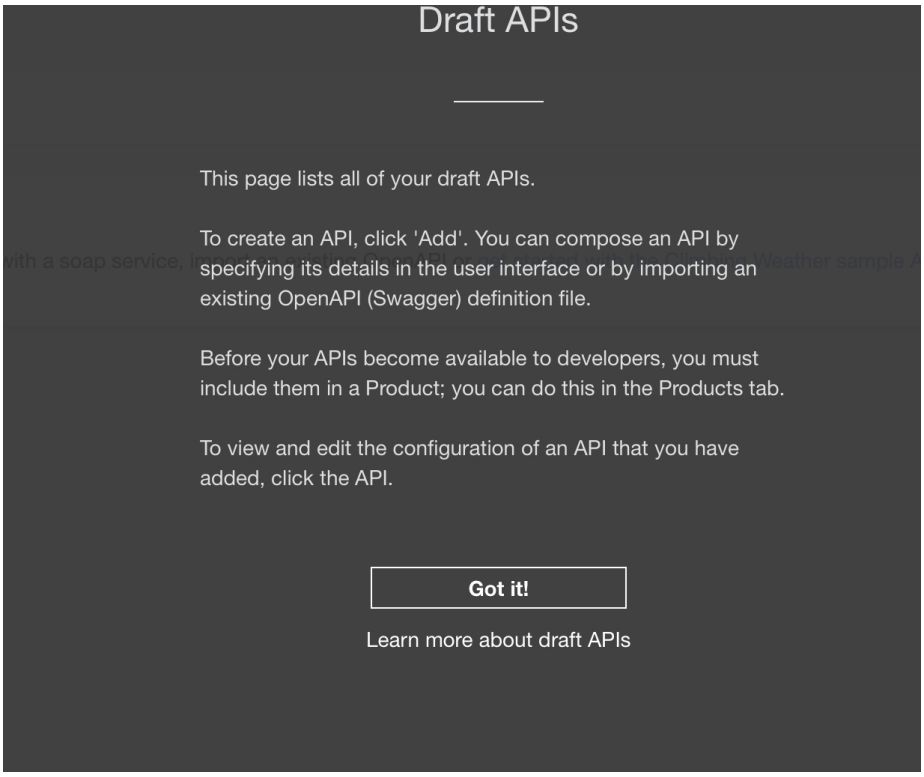
Pick the API Connect service tile



Pick the defaults for the service

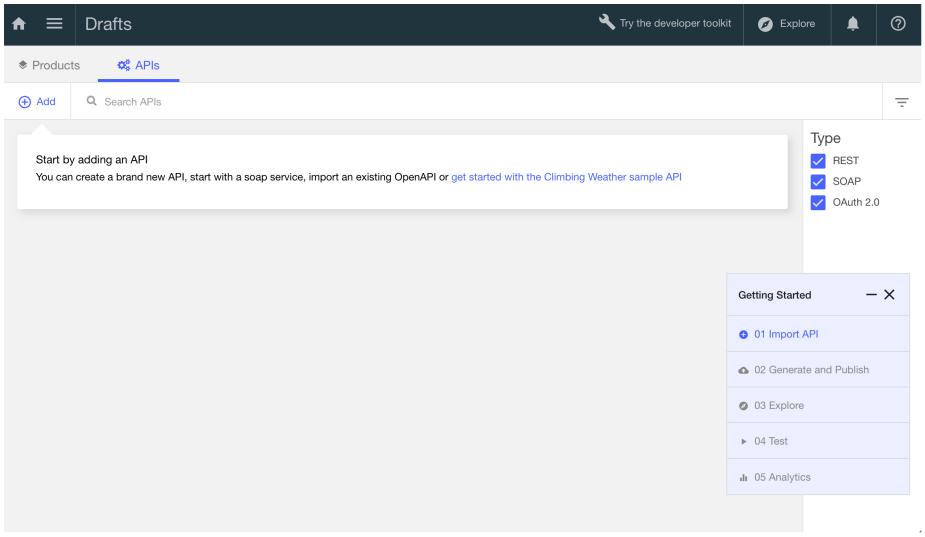


You will presented tith a Drafts API screen that looks like below.

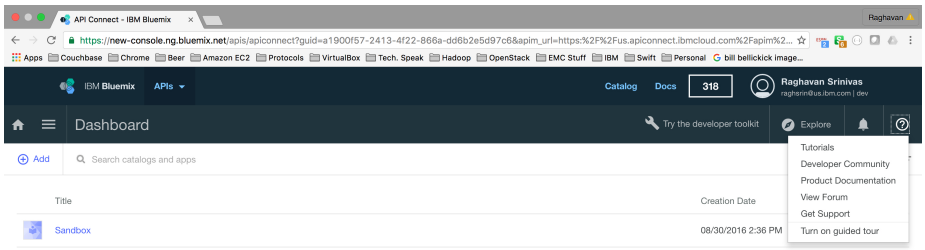


Click on “Got it” and proceed.

You will be presented with the Drafts screen that looks like below with the `Getting Started` window on the top right as shown below.



If you do not see the "Getting Started" window at any time, **click on ? and "Turn on Guided Tour"** as shown below.

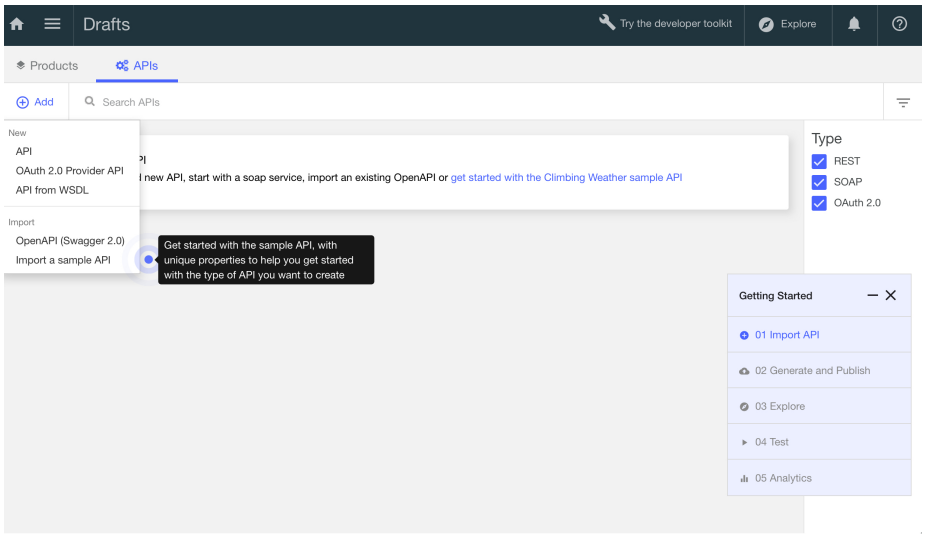


We will follow the "Getting Started" tour essentially from now on.

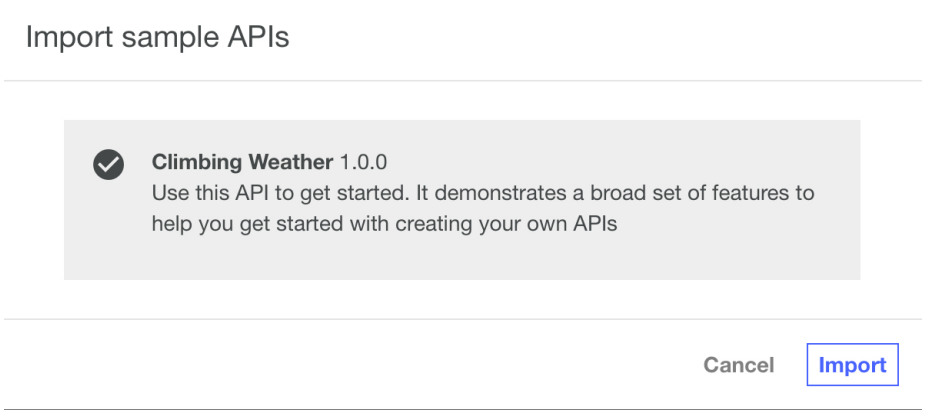
A tour of the API Connect Manager via the sample Weather API

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

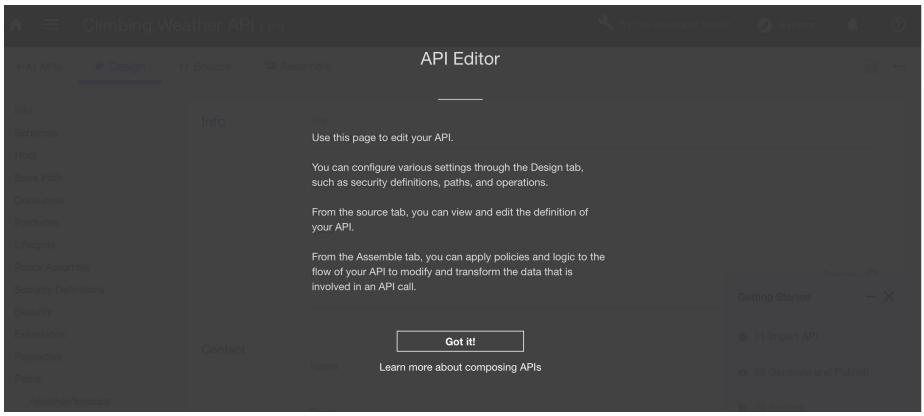
As indicated before, if you do not have the `Getting Started` window on top right, **click on ? and "Turn on Guided Tour"**



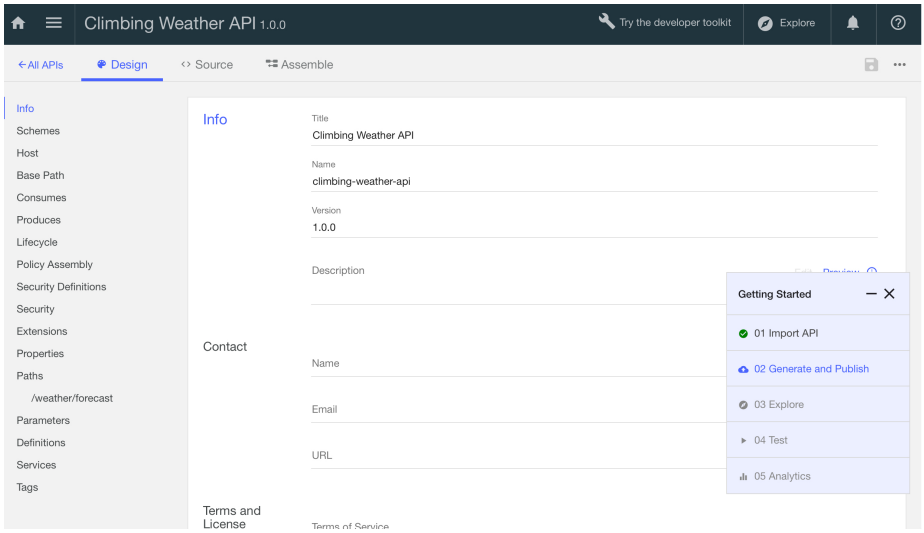
In the Dialog box shown below pick the "Climbing Weather" project and hit "Import" as shown below.



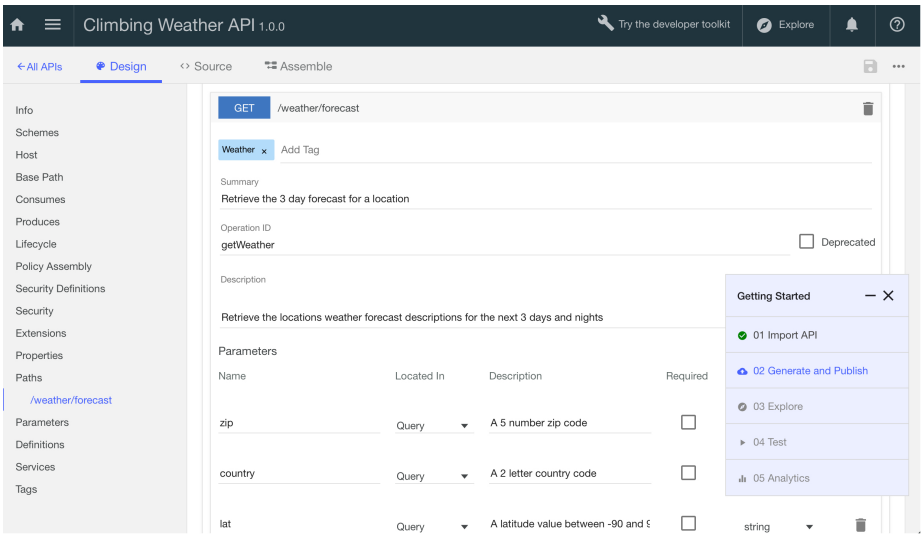
This will prompt you about the "API Editor" as shown below. Hit "Got it!".



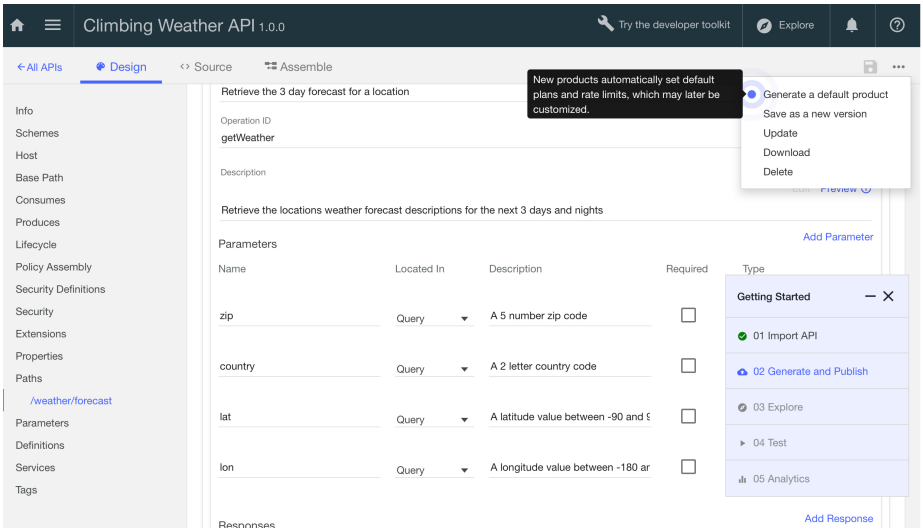
This will bring up the Design view similar to what you saw earlier, this time on Bluemix.



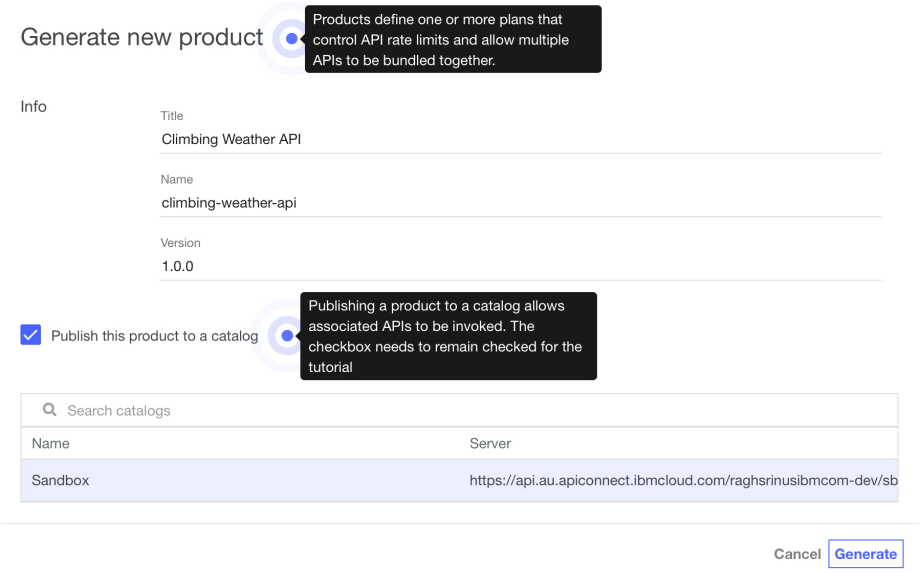
Scroll down to the `getWeather` method which looks like below.



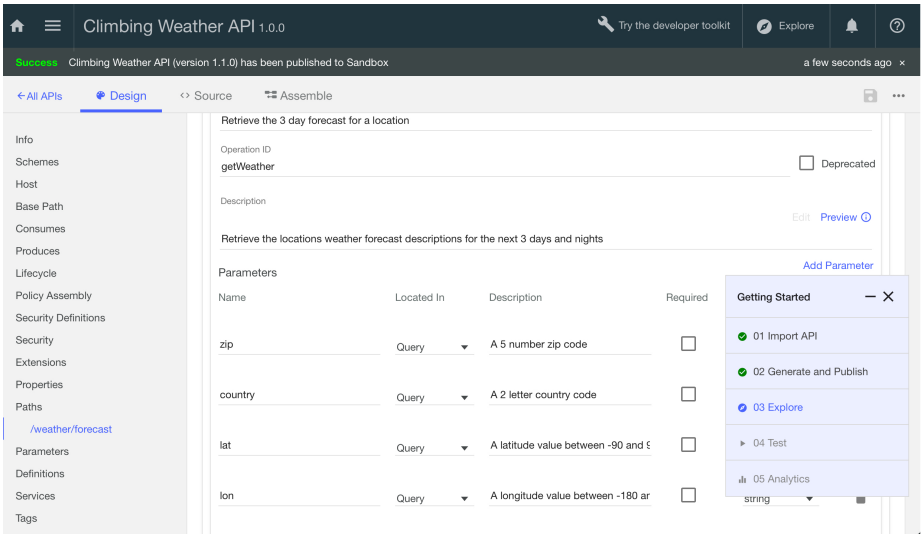
Click “Generate and Publish” in the “Getting Started” window picking “Generate a Default Product” as shown below.



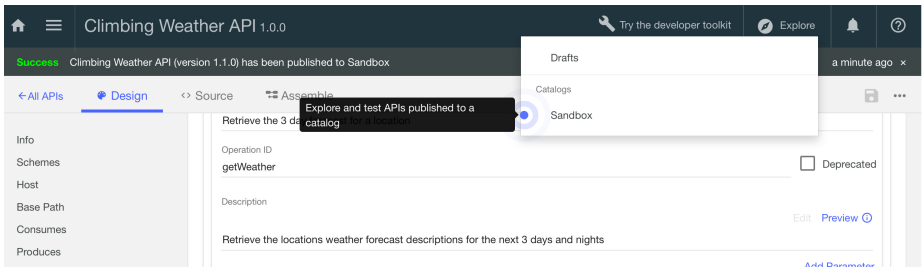
You will see that a default plan will be used and that you will publish the product to the Sandbox catalog. Click on “Generate” as shown below.



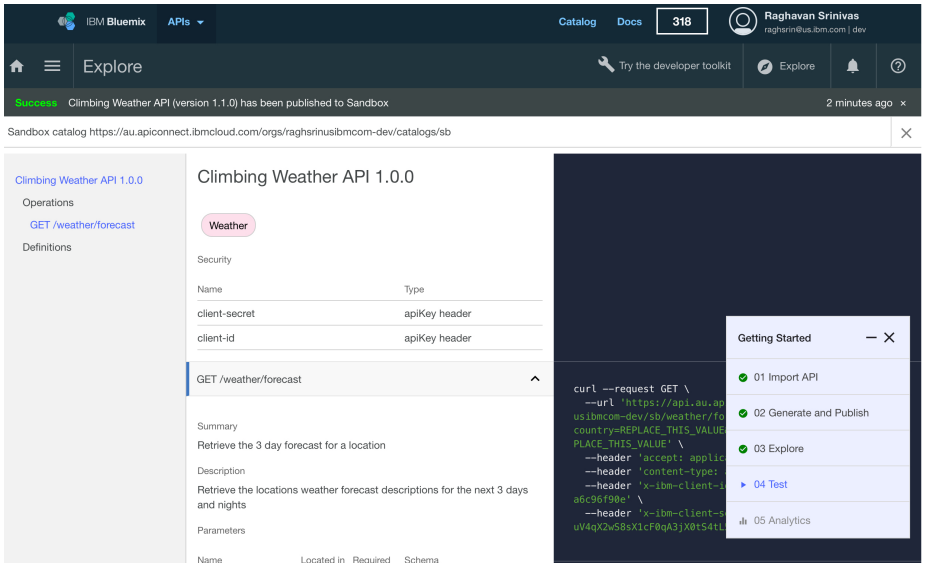
Next, click “Explore” in the “Getting Started” window as shown below.



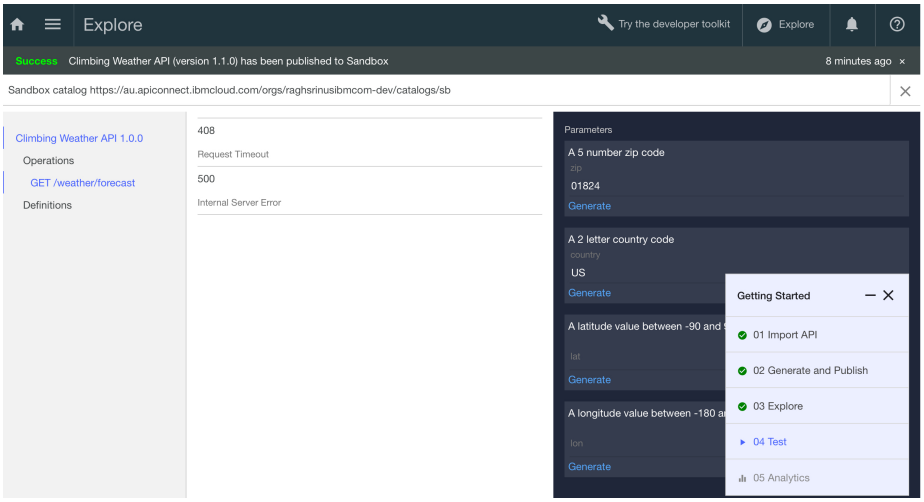
Pick “Sandbox” as prompted and as whown below.



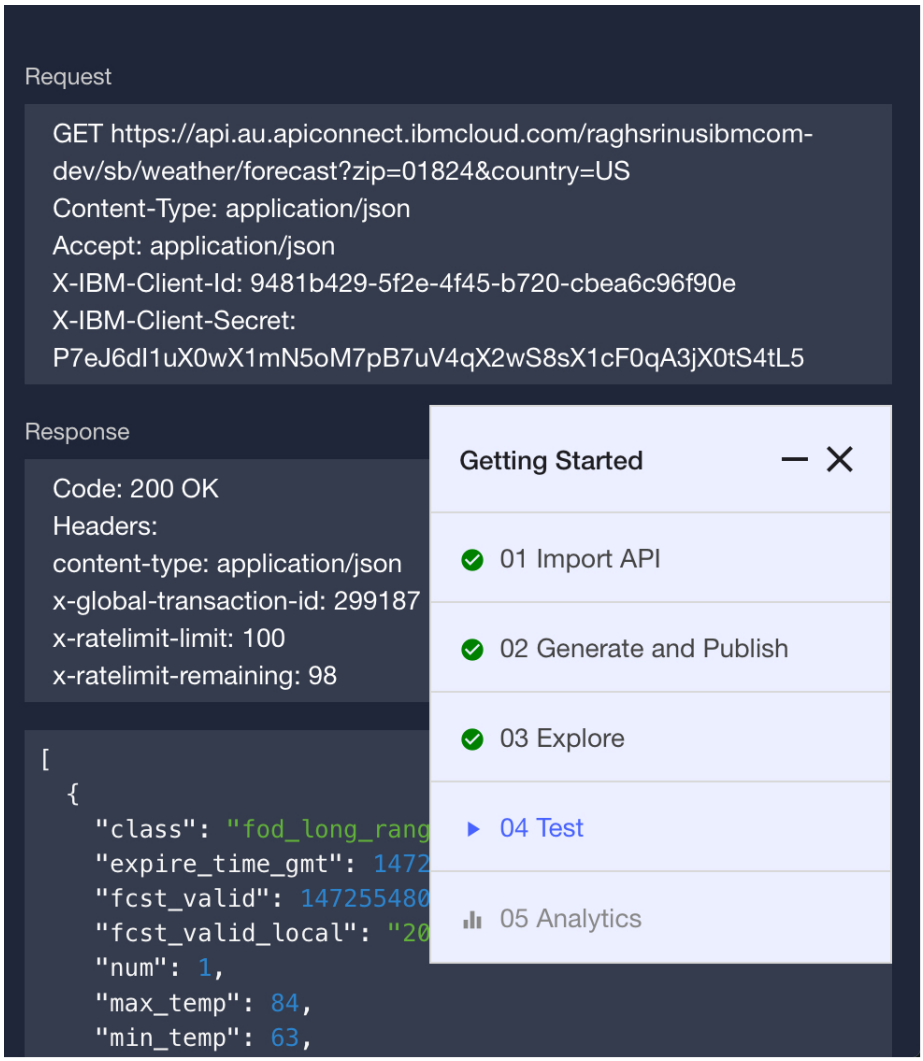
You will see the sample API as shown below.



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)

The following screen shots walks through how to ovveride this via a series of dialog boxes. This might be different based on the browser, the URL and the error but is illustrated below

Call operation

Request

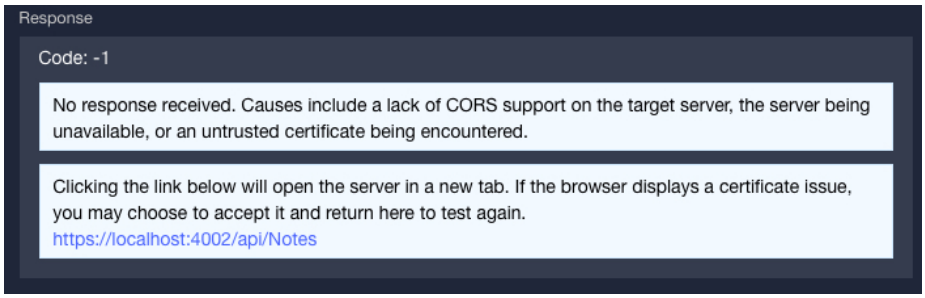
GET https://api.eu.apiconnect.ibmcloud.com/raghsrinusibmcom-dev/sb/weather/forecast?zip=01824&country=US
Content-Type: application/json
Accept: application/json
X-IBM-Client-Id: 591eee59-0f75-42e4-88b3-c6b445572748
X-IBM-Client-Secret: fA5jR1kK5dS5xN2nD6wY7wl6cW5yX8rE2lH4kR2yN3rG5jR7lE

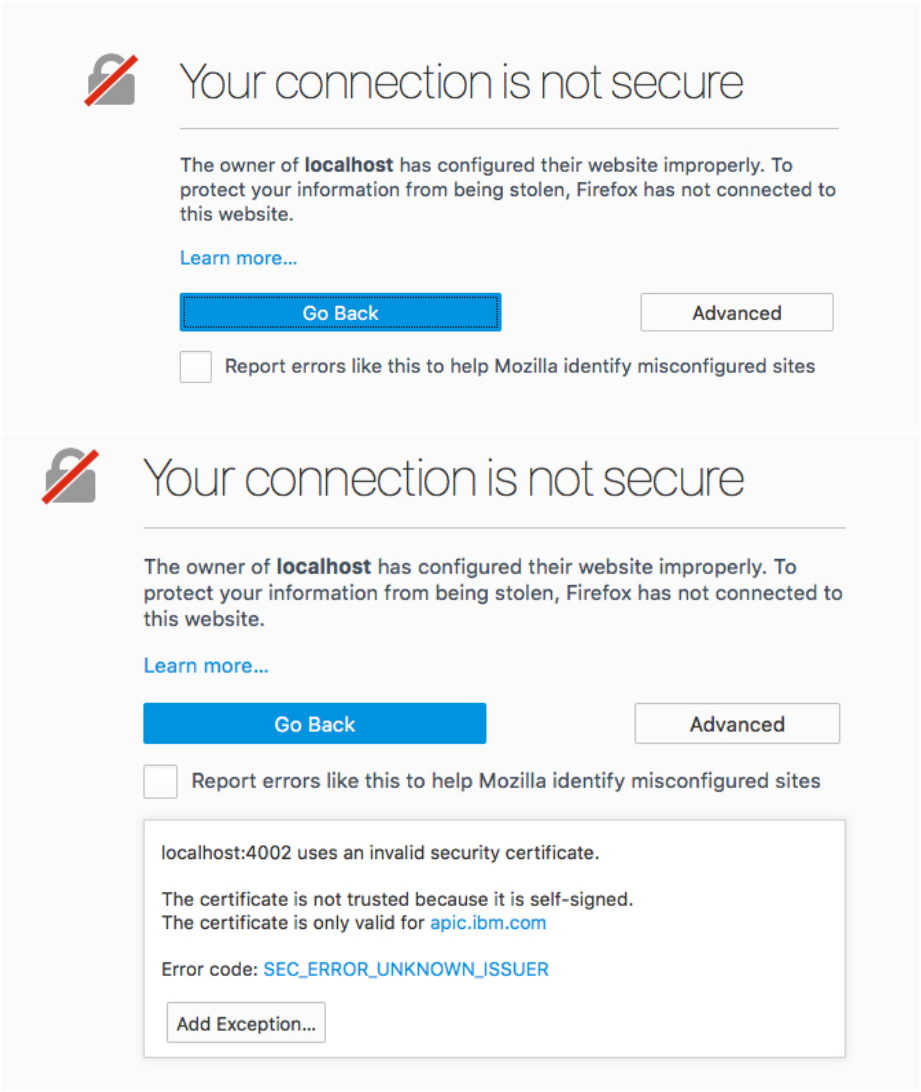
Response

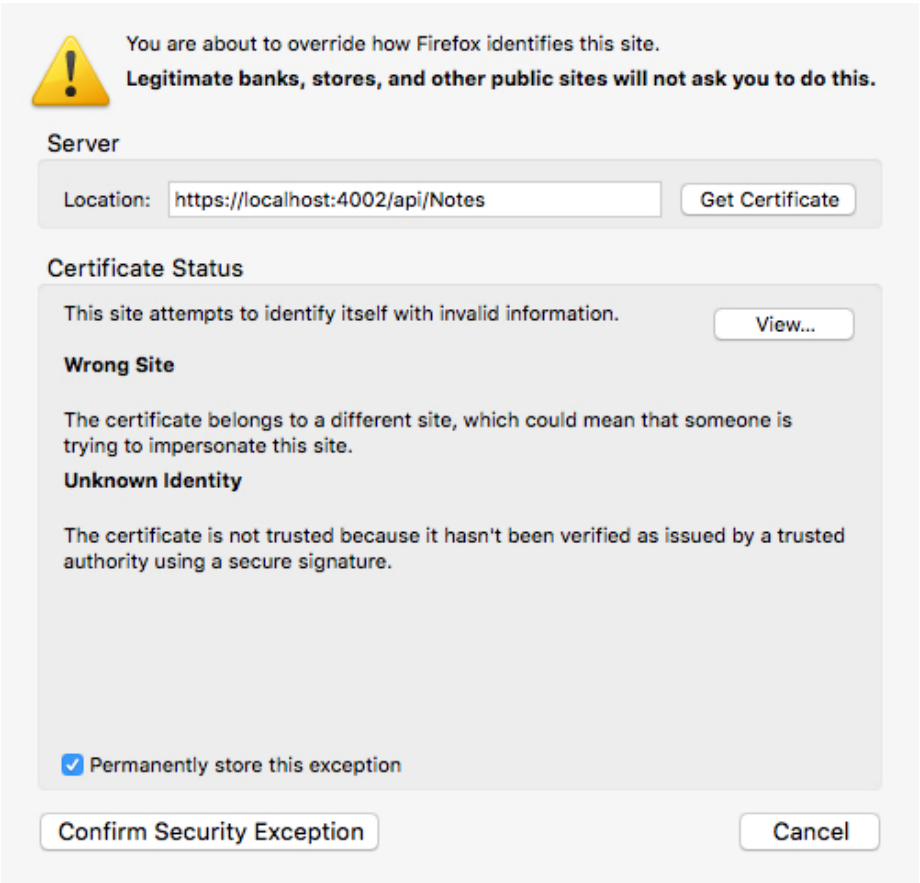
Code: -1

No response received. Causes include a lack of CORS support on the target server, the server being unavailable, or an untrusted certificate being encountered.

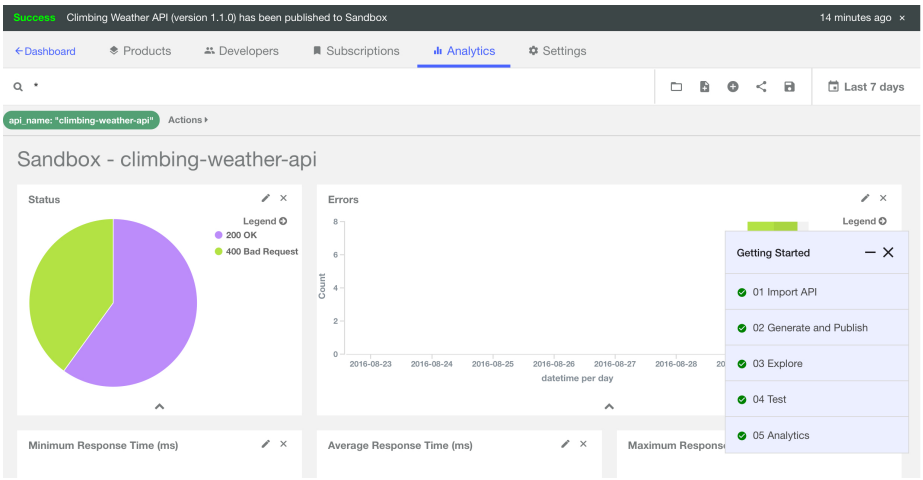
Clicking the link below will open the server in a new tab. If the browser displays a certificate issue, you may choose to accept it and return here to test again.
https://api.eu.apiconnect.ibmcloud.com/raghsrinusibmcom-dev/sb/weather/forecast?zip=01824&country=US




Your connection is not secure
The owner of localhost has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.
Learn more...
Go Back
Advanced
Report errors like this to help Mozilla identify misconfigured sites
Your connection is not secure
The owner of localhost has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.
Learn more...
Go Back
Advanced
Report errors like this to help Mozilla identify misconfigured sites
localhost:4002 uses an invalid security certificate.
The certificate is not trusted because it is self-signed.
The certificate is only valid for apic.ibm.com
Error code: SEC_ERROR_UNKNOWN_ISSUER
Add Exception...



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



Which should show the usage statistics for the product.

Summary of exercise and next steps

We started with the API connect product locally in an exercise 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.

In Exercise 3 we will look at how to use the OpenAPI specification (swagger) editor.

API Connect Hands-On Labs

Exercise 3: Design your OpenAPI Swagger specification

Prerequisites

Make sure you've met the following prerequisites.

Prerequisite 1: Installed the API Connect toolkit (Exercise 1).

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex3.

```
cd <path-to-hol-folder>/exercises/ex3
```

Overview of exercise

For this exercise, we'll:

1. Learn about the OpenAPI Specification (Swagger Specification) 2.0 and its components
2. Learn how to use Swagger Editor to design and modify an existing Swagger JSON specification of an API
3. Learn how to import an OpenAPI Specification into API Designer for further editing

OpenAPI (Current Version: 2.0)

The Open API Specification (formerly known as the Swagger specification) is the industry standard for defining REST APIs. The goal of the OpenAPI Specification is to define a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined via OpenAPI, a consumer can understand and interact with the remote service with a minimal amount of implementation logic.

OpenAPI specs remove the guesswork in calling a given service

OpenAPI Components

- FORMAT: The files describing the RESTful API in accordance with the Swagger specification are represented as **JSON objects** and conform to

the JSON standards. YAML, being a superset of JSON, can be used as well to represent a Swagger specification file.

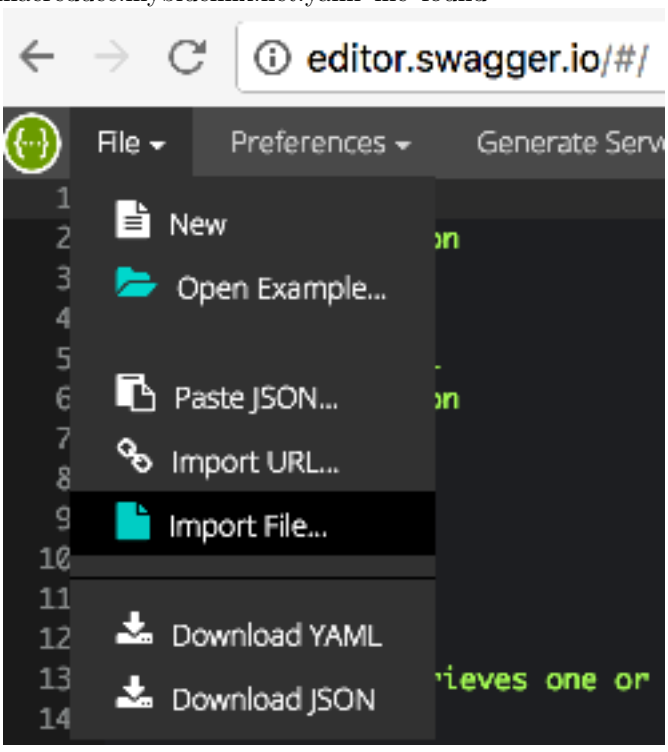
- BENEFITS: With a OpenAPI specifcation, you'll be able to generate client libraries in lots of runtime languages, generate server stubs, import these definitions into API management tools such as Bluemix APIConnect and use tools to verify conformance.

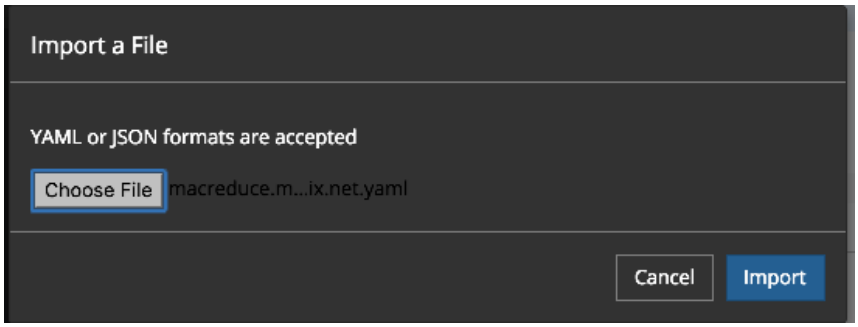
Swagger Editor

Swagger Editor at http://editor.swagger.io/#/ is a handy open source web application that lets you quickly edit OpenAPI Swagger specifications in YAML or JSON. You can import or create custom specifications within a browser. We'll use this hosted editor and a Swagger JSON specification **macreduce.mybluemix.net.json** located within the ex3 sub-directory to illustrate the use of the editor and how it can be used to design/modify a Swagger specification.

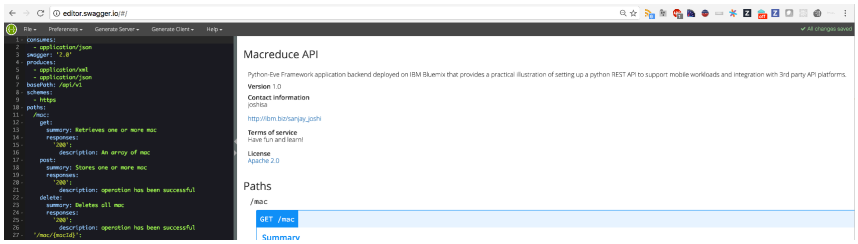
Exploring an OpenAPI (Swagger) 2.0 Specification

1. Browse to Swagger Editor and click on the File menu choice **Import File...** . You will select the macreduce.mybluemix.net.yaml file found within your ex3 sub-directory.





2. After import, you should see a split pane view with the raw text Swagger spec on the left and a rendered view on the right.



3. We will now make a modification to the existing swagger specification. Browse to the section of the spec that looks similiar to:

```
'/mac/{macId}':
get:
  summary: Retrieves a Mac document
  responses:
    '200':
      description: Mac document fetched successfully
  parameters:
    - in: path
      name: macId
[...]
```

We will add an expected response to document the API's behavior when attempting to fetch a non-existent macId resource. Add two lines just above the parameters section so that it now looks like this:

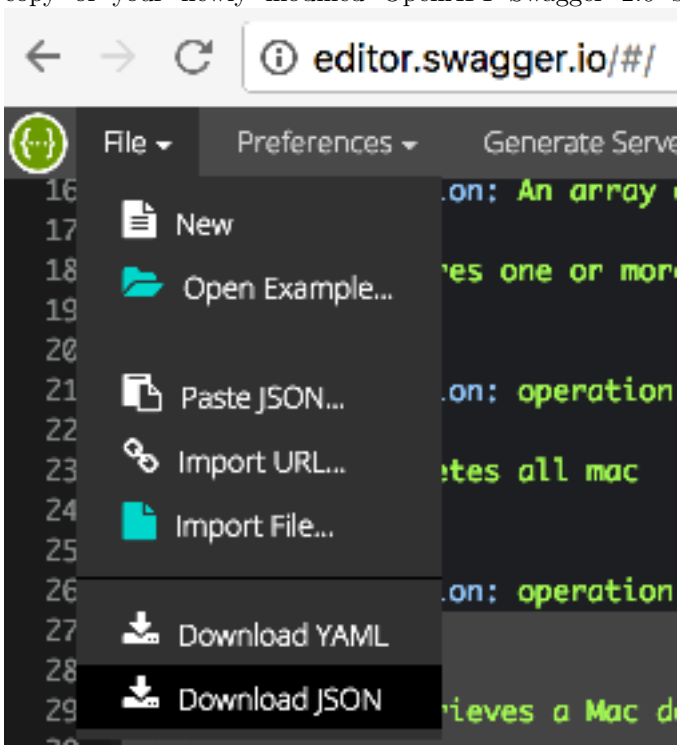
```
'/mac/{macId}':
    get:
      summary: Retrieves a Mac document
      responses:
        '200':
          description: Mac document fetched successfully
        '404':
          description: Mac document not found
```

```
        parameters:
          - in: path
            name: macId
[...]
```

The specification provides a wide variety of data elements to facilitate describing your API set. As you inspect your OpenAPI specification file, some important elements to consider include:

- method types (get, post, delete, …)
- responses (200, 400, 404, …)
- parameters
- paths
- content-types (application/json, application/xml, …)

4. Click on the File menu choice **Download JSON** to obtain a local copy of your newly modified OpenAPI Swagger 2.0 specification.



5. As you can see, Swagger Editor is a great tool for modifying existing OpenAPI specifications and/or creating brand new specifications.

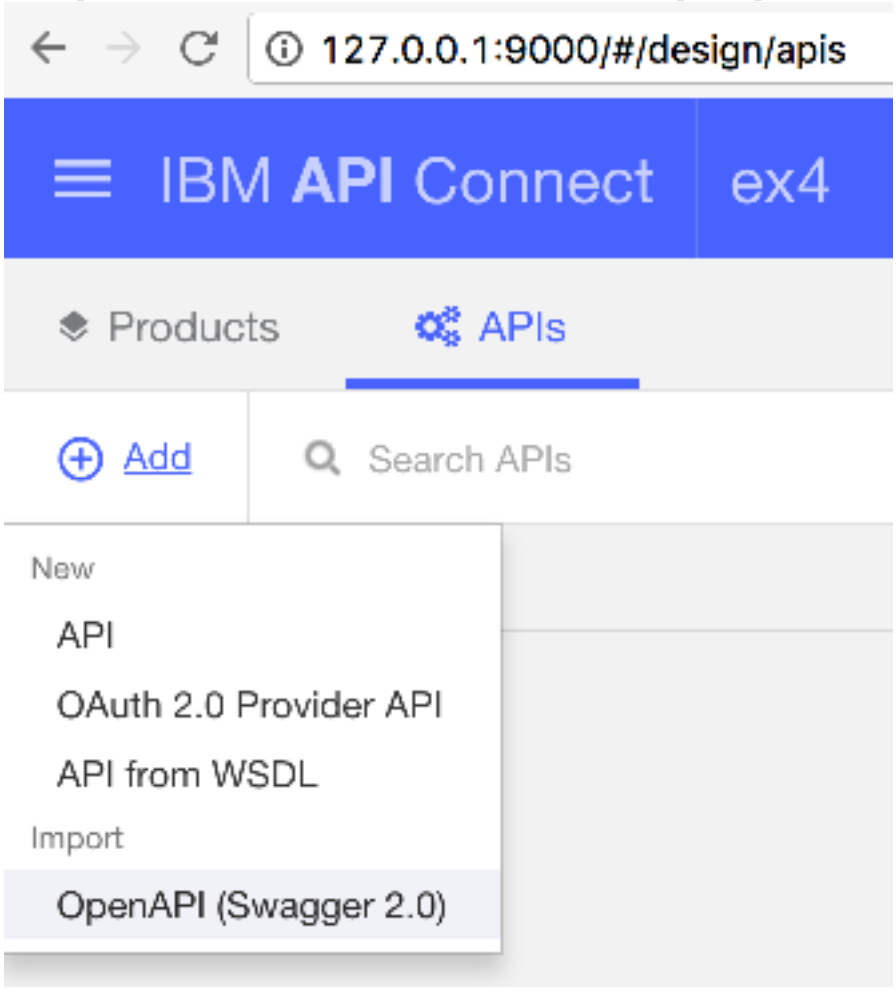
Open API Spec Explorer and Designer

To open the API Designer, on the command line enter:

```
SKIP_LOGIN=true apic edit
```

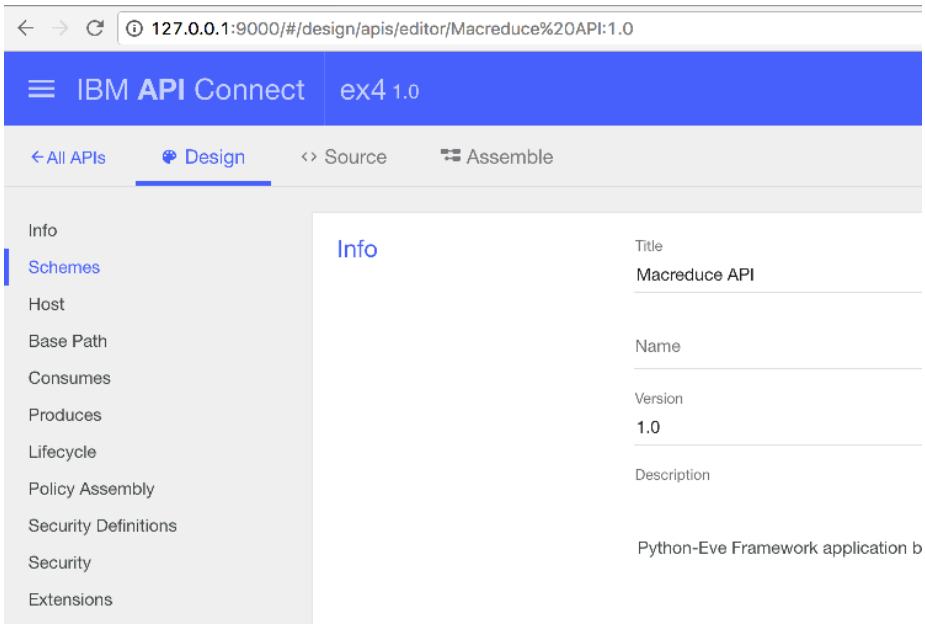
This should result in the API Designer opening within your default web browser.

Our next step is to import the OpenAPI specification into API Designer. To accomplish this, click on the + **Add** link and select the Import OpenAPI choice



You will want to browse to the location where you downloaded the swagger.json file created from Swagger editor and click on the **Import** button.

This should cause focus on the API Design tab with various fields populated via data found within the OpenAPI specification file. Feel free to explore the various section links on the left to get a feel for design options available to you.



It is also worth noting that if you already possess an existing backend API application, you can publish this Open API (Swagger) specification to Bluemix whereby the platform would then manage your existing APIs for you. Platforms such as Bluemix are great at providing analytics, gateway, security, authentication and user management facilities for your API needs.

Summary of exercise and next steps

We've now learned quite a bit. We know what an Open API (Swagger) specification is, how its used and what is its composition. We've explored a couple of tools that assist us with Open API design, composition and management.

In Exercise 4, we'll create a Loopback Application against these defined APIs. Having a backend application takes the conceptual descriptions within the spec and makes them concrete (e.g. Functional Create, Read, Update and Delete API endpoints).

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 Installed the API Connect toolkit (Exercise 1)

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex4.

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

Overview of exercise

In this exercise, we'll:

1. Learn about how to create a LoopBack application
2. Learn how to consume an OpenAPI specification and shape a backend LoopBack application's behavior
3. Learn how to implement basic behavior around the imported API design

LoopBack API

We'll first make an empty project directory to contain all of our work for exercise 4.

```
mkdir -p ./loopbackapp
```

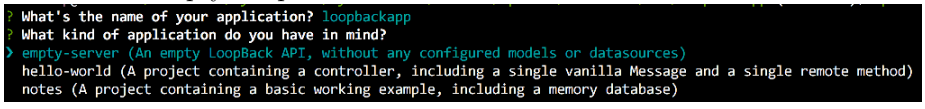
Next, navigate within the empty loopbackapp folder by typing the following:

```
cd loopbackapp
```

Next, we'll leverage the **apic** binary to create a Loopback Application project. 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.



This should result in a parade of files being generated within the folder that represents a skeleton LoopBack application. Next, we'll create an in-memory db datastore. This db datastore is required to help us test our new Loopback application – specifically for create operations which use HTTP POST and PUT method calls. Without a datastore, our application will have no default place to store data sent to it.

Execute the following command:

```
apic create --type datasource
```

You'll be prompted for:

1. A data-source name. Enter **db**.
2. A connector for the db: select the default selection of **In-memory db**.
3. A window.localStorage key for persistence. Accept the default of blank.
4. A full path to file for persistence. Accept the default of blank.

This will update your applications definition file with this new datasource.

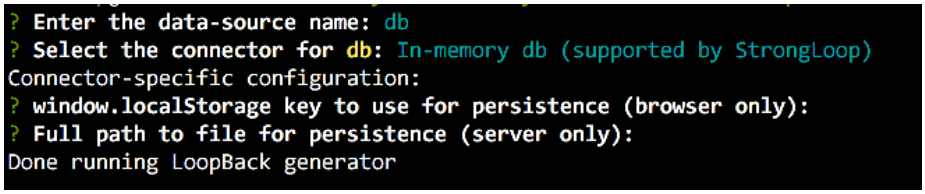


Figure 1: newdatasource

Next, we'll import our OpenAPI specification from exercise 3 to shape this application. To prepare for this, copy the downloaded OpenAPI spec json file obtained from Swagger Editor and place it within the ex4 directory. As an alternative, you can also copy an unmodified swagger spec file by executing the following command:

```
cp ../../ex3/macreduce.mybluemix.net.json ../swagger.json
```

Execute the following command:

```
apic loopback:swagger
```

A series of menu prompts will display:

1. Prompt: A local path or remote url to your OpenAPI specification file.
   Response: Enter a path to the json file that you copied in an earlier step (e.g. **../swagger.json**).
2. Prompt: Selection of the models to be generated.
   Response: You'll select the default of **swagger_api_v1**.
3. Prompt: Selection of the data-source to attach models to.
   Response: You'll select the default of **db (memory)**

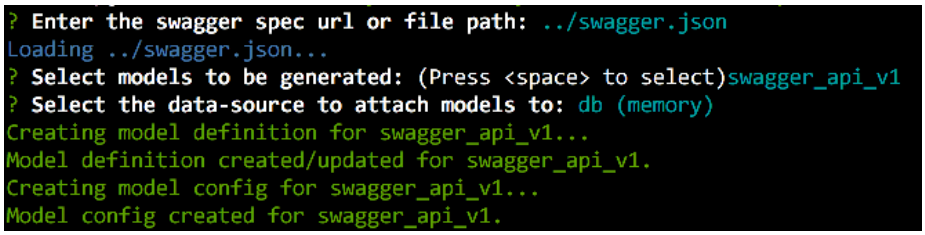


Figure 2: swaggershaping

Next, you'll confirm that our datasource is attached to our model using the API Design and Management User interface which exposes concepts such as the underlying API model and registered datasources. To do this, let's jump into the API Design and Management UI by executing the following command:

```
apic edit
```

This should result in the API Design and Management UI opening within your default web browser. You'll need to sign in with your Bluemix account and browse to the Models tab …

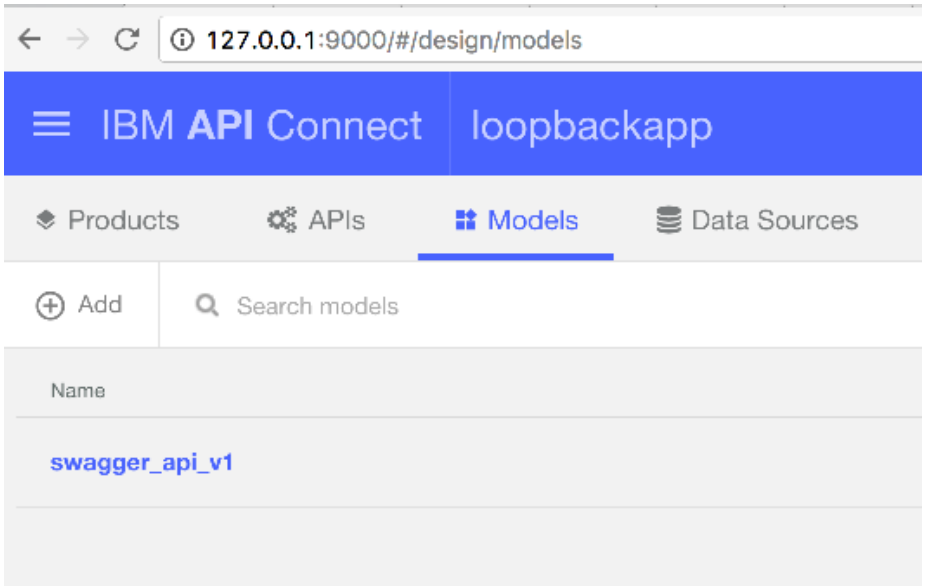


Figure 3: Models Tab

and click on our model named `swagger_api_v1`. This will open the properties view for the model. Click the dropdown arrow and select the datasource **db**. Hit the **save icon** in the upper right of the window and then close the browser tab. You'll also need to break out of the running `apic edit` process within the

terminal.

Figure 4: New Datasource

Sweet! We now have a node application with endpoints defined via our OpenAPI specification. To test, let's fire up the app:

```
node .
```

We'll observe that the application is listening on port 3000. Using a browser, let's navigate to the following url: http://127.0.0.1:3000/api/api/v1/mac .

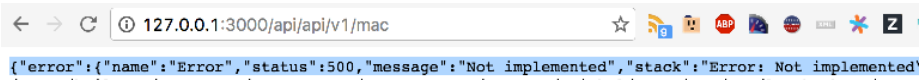


Figure 5: not implemented

Ughh ... a 500 response error.

Wait, that makes total sense. We've defined endpoints via a spec but have not implemented any logic around the behavior of the resources. Let's tackle that next.

Kill your running node process within the terminal and locate the LoopBack application's model controller file **swagger-api-v-1.js**

```
ls -al ./server/models/
```

Within this folder, you'll notice two files:

1. swagger-api-v-1.js : defines implementation logic for the generated stub APIs generated
2. swagger-api-v-1.json : defines meta properties about the generated API model

To expedite the logic implementation, a handy uncommented copy of a partially implemented controller file is provided within the ex4 parent folder named **swagger-api-v-1.js.uncommented**. While in the project folder loopbackapp folder, execute the following command to replace the existing controller with this partial implementation:

```
cp ../swagger-api-v-1.js.uncommented server/models/swagger-api-v-1.js
```

You'll be prompted to overwrite. Respond with **y** (yes).

This partial implementation enables four (4) of the OpenAPI specification entries for

- **GET** and **POST** on `/mac`
- **GET** and **DELETE** `/mac/{macId}`

If you're curious, we've also provided for comparison and inspection, a commented copy in the ex4 directory. (`../swagger-api-v-1.js.commented`).

Let's fire up our Loopback Application based on Swagger once again … and try to populate it with some data.

```
node .
```

In a separate terminal window, issue the following 2 commands:

```
curl -X POST -H "Content-Type: application/json" -d "{\"organization\":\"IBM Corporation\",\"h
curl -X POST -H "Content-Type: application/json" -d "{\"organization\":\"IBM Corporation\",\"h
```

If all went well, you should receive a response within your terminal from your node backend API server that looks similar to:

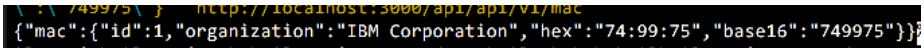


Figure 6: Successful Post

While it continues to run, let's now fetch a record associated with an ID=2. Navigate your browser to http://localhost:3000/api/api/v1/mac/2

The result should look similar to this:

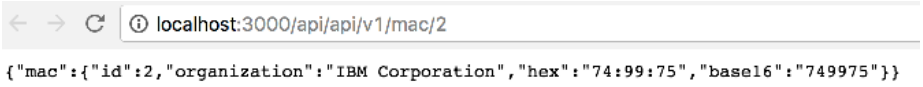


Figure 7: Successful Get

Awesome sauce! Go ahead and kill the running node process within your terminal and change directories to exercise 5.

Summary of exericse and next steps

You've now walked through the process of quickly creating a REST API Loopback Application shaped by an OpenAPI (swagger) specification. You also learned how to modify the generated backend Loopback application to partially implement support for a couple of HTTP method types (GET and POST).

In Exercise 5 we'll learn how to easily create and populate a MySQL database service instance as a building block towards our ultimate goal of establishing API operations that support Create, Read, Update and Delete (CRUD) methods.

API Connect Hands-On Labs

Exercise 5: Create a database service on Bluemix

Prerequisites

Make sure you've met the following prerequisites.

Prerequisite 1: Installed the API Connect toolkit and targeted the Bluemix instance (Exercise 1).

Ensure that you are in the right sub-directory

Ensure that you are in sub-directory ex5.

```
cd <path-to-hol-folder>/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
```

Overview of Exercise

For this exercise, we seek to learn how to rapidly instantiate a MySQL DB service and populate it with sample data, thus forming an asset for future exercises exploring the creation of database CRUD APIs.

We will use a simple Java application to populate the data that will be accessible via the APIs.

Although the API is based around a simple `employees` database, you can follow similar methods to provide APIs around your more complex and production enterprise Java/Node.js application.

Creating a database service

With a Platform as a Service (PaaS) such as IBM Bluemix, DB creation and interaction is super easy – as you'll see.

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 also offers a listing of the entire service marketplace registered within the PaaS platform. **Beware, this can be a lengthy query** (~2-5 mins) proportional to the size of the platform catalog. For this workshop, it neither recommended nor required. However, 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                                                    des
APIConnect                          Essentials, Professional*, Enterprise*, Professional 5M*,
AdvancedMobileAccess                   Gold*, Bronze*
Application Security on Cloud            free, standard*
Auto-Scaling                        free                                                     Au

[....]
```

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. 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                                                free or paid
spark       Great for getting started and developing your apps                  free
```

```
boost       Best for light production or staging your applications                    paid
shock       Designed for apps where you need real MySQL reliability, power and throughput  paid
amp         For apps with moderate data requirements                                  paid
```

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

At this point, it would seem like we have two options. Alas! there is another key detail that we absolutely must consider for our use case.

Whether a service provides a public vs. platform endpoint access of the DB service

The community provided MySQL service will generate connection urls that contain **private network IPs** only visibile to other platform applications (e.g. IBM Bluemix applications, containers, etc …). By contrast, the ClearDB 3rd party MySQL service generates **public internet visible connection urls**. Public accessibility is the characteristic that we must embrace for our Hands On Lab needs, given that we'd like to connect and populate this DB from our local VM image. Therefore, we will create our MySQL service instance by issuing the following command specific for ClearDB MySQL:

```
cf create-service cleardb spark 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. Fear not, we simply 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.

```
{
 "hostname": "us-cdbr-xxxx-xxxx-36.cleardb.net",
 "jdbcUrl": "jdbc:mysql://us-cdbr-xxxx-xxxx-36.cleardb.net/ad_2xxxxxxxxxxf?user=b2xxxxxxxxx
 "name": "ad_2xxxxxxxxxxf",
 "password": "1xxxxxx7",
 "port": "3306",
 "uri": "mysql://b2xxxxxxxxx7: 1xxxxxx7@us-cdbr-xxxx-xxxx-36.cleardb.net:3306/ad_2xxxxxxxxx
 "username": "b2xxxxxxxxx7"
}
```

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

Let's leverage this newly minted power to populate it with some Employee Data. If you're curious about the data that we'll insert, check out the `populate.sql` file found within this folder. You'll see a very long set of VALUES representing basic personnel info like names, birthdate, etc …:

```
INSERT INTO `employees`
(emp_no, birth_date, first_name, last_name, gender, hire_date)
VALUES      (10001,
             '1953-09-02',
             'Georgi',
             'Facello',
             'M',
             '1986-06-26'),
[....]
```

To make things smoother, we've provided a convenient set of scripts and a CLI Java file that illustrates:

- Using Java Code with the MySQL Connector jar to connect to an IBM Bluemix db service
- Using Bash/Shell scripting techniques to harvest db service credentials using the cf CLI

You're encouraged to inspect the **ex5.java**, **setupMySQL.sh** and **svcinspect.sh** files found within this directory that help put these concepts into action. Assuming that your ClearDB MySQL service is named **workshopmysql**, execute the following command. Otherwise, substitute your alternate ClearDB MySQL service instance name as appropriate:

```
./setupMySQL.sh workshopmysql
```

The utility execution is broken up into 3 main phases:

- Service Inspection + Credential Gathering
- Java Compilation
- SQL Updates and Queries

Your output should look similar to:

```
  Welcome to the MySQL Setup Helper Script
  Brought to you courtesy of IBM
  setupMySQL.sh invoked

  Welcome to the Service Inspector ...
  Target Instance: workshopmysql
  Interrogating Service Instance for Credentials ...
    Service Key Found: connectioncreds
    Parsing Service Key ...
    Service Instance Type: ClearDB
      hostname: us-cdbr-iron-east-04.cleardb.net

[....]

  Fetching commons-cli-1.3.1.jar from http://.../commons-cli-1.3.1-bin.tar.gz
2016-09-15 23:22:22 URL:http://.../commons-cli-1.3.1-bin.tar.gz [305600/305600] -> "commons-c
  Compiling Exercise 5 Java Setup, Populate and Query Utility
  Executing utility to see the arguments available ...
usage: ex5 [-d <dbname>] [-h <host>] [-n <port>] [-p <password>] [--query]
       [-s <sql>] [-u <username>] [-w <wanthelp>]
       Options, flags and arguments may be in any order

[....]

Updating target MySQL DB ...
0
  Employees Table Created

[....]

Updating target MySQL DB ...
1000
  Employees Table Populated

[....]

Querying target MySQL DB ...
10001  1953-09-02  Georgi  Facello  M  1986-06-26
10002  1964-06-02  Bezalel  Simmel  F  1985-11-21
10003  1959-12-03  Parto  Bamford  M  1986-08-28
10004  1954-05-01  Chirstian  Koblick  M  1986-12-01

[....]
```

For the curious, the utility intentionally generates and persists a resultSet of

the records in a tab delimited (*.tsv) file within this directory for your perusal.

Sweet! That's it! We can all agree, that was pretty simple to standup a MySQL DB and get it populated.

Summary of exercise and next steps

We started with a goal of instantiating a MySQL DB and getting it populated with some sample data. We learned about available IBM Bluemix DB services, the required details for creating a MySQL DB instance and the use of a simple Java program that consumes the generated credentials to facilitate connecting and populating the new MySQL DB instance.

In Exercise 6, we will dive into creation of database CRUD APIs that leverage this just populated DB.

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 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).

Overview of Exercise

In this exercise we will create simple CRUD-based APIs around the `employees` database.

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-hol-folder>/apichol/exercises/ex4/loopbackapp
```

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.

If the previous step did not work, continue with this step. Otherwise, proceed below.

Work-Around: If you saw an error, switch back to your terminal. Use `Ctrl+C` to end the `apic edit` instance. This shuts down the API Designer Toolkit. Then, type the command `npm i --save loopback-connector-mysql` to manually install the LoopBack connector for MySQL databases. Then, run `apic edit` again, switch to the `Databases` tab, open your `mysql-db` entry, and continue below.

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

Let's create a model so that you're able to perform CRUD (Create/Read/Update/Delete) operations against your MySQL database. Go to the Models tab and add a new model by clicking the add button. Name it `employees`.

Choose the `mysql-db` Data Source, and enter the following properties:

We are about to create a LoopBack model representation of the MySQL database. The name of the model `employees` refers to the table name in the MySQL database. The various property names refer to the MySQL "fields". The types translate to MySQL fields – for example, `String` to `VARCHAR`. `Key` in LoopBack is similar to `Key` in MySQL.

Hit the `Save` button on the top right to create the model. That's it! Once a model is created, the APIs to represent that model are automatically generated for you.

Summary and next steps

We just created APIs around the `employees` database.

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).

Overview of Exercise

In this exercise we will test the APIs created in the previous exercise, explore the swagger-based API UI, and deploy your APIs to Bluemix.

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.

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.

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.

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.

GET /Employees

Let's test retrieving the list of 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.

You might get a CORS error. Please override the CORS error as suggested in exercise 2. Then, retry the `Call Operation`.

You should see a `200 OK` response, along with a large list of employees in the database!

POST /Employees

Let's test adding an employee to the database.

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

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

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.

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

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

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

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.

It may take a few minutes for the application to get started. You can track the status of your application by using the `CF` client that you setup in Exercise 1. Simply run the `cf app EmployeeAPI` command:

```
$ cf app employeeapi
Showing health and status for app employeeapi in org raghsrin@us.ibm.com / space dev as raghsrin
OK

requested state: started
instances: 0/1
usage: 256M x 1 instances
urls: apiconnect-be783035-d8e9-4ceb-b2d7-f16e9340c1d1.raghsrinusibmcom-dev.apic.mybluemix.ne
last uploaded: Tue Sep 13 19:06:32 UTC 2016
stack: cflinuxfs2
buildpack: unknown
```

Once the `requested state` is `started`, the application is started and your APIs are being managed by API Connect!

Summary and next steps

In this exercise we explored, tested and deployed the application to Bluemix.

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).

Overview of exercise

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.

Accessing your deployed APIs

Navigate to the Bluemix console at (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 previously created API Manager.

Let's explore the APIs you just pushed. Hit the `Explore` button on the top right, and choose the `Sandbox` catalog. This takes us to a view similar to the `Explore` view on the API Designer that we used in Exercise 7. The main difference is that this time, we are testing these APIs hosted by an application on Bluemix.

On the left, click on the app you just pushed, which is the name of your LoopBack application created earlier.

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.

This should return a list of the employees in the database.

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.

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.

Summary and next steps

We explored the app that was pushed in the previous exercise to Bluemix.

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 (Exercise 2)

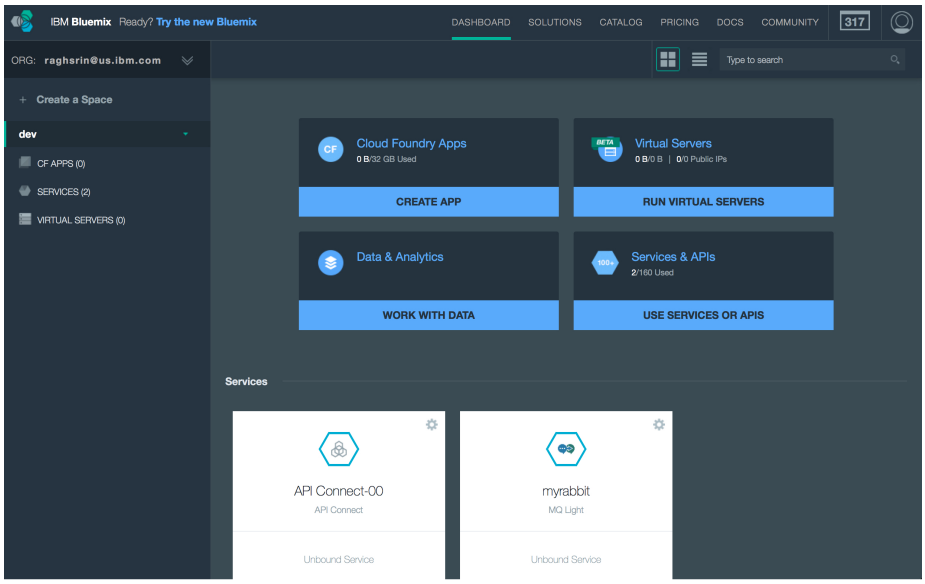
Overview of exercise

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.

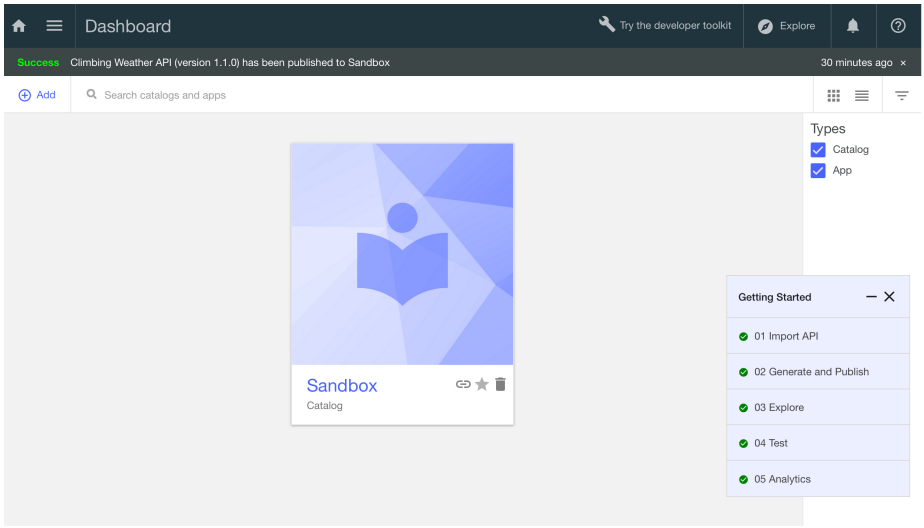
In this exercise, we will publish the APIs, with an associated plan and so on to a self-service developer portal which is an adjunct to the Bluemix API Connect service. These APIs can be invoked by any REST-based client.

Publishing the Sample API to the Developer Portal

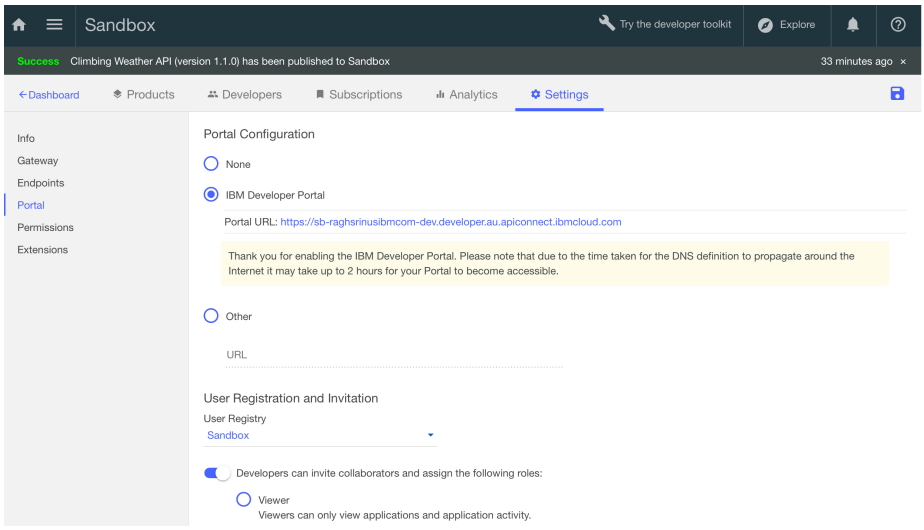
Let's head back to the API Connect Manager on IBM Bluemix and to the sandbox that we finished off in exercise 2 as we see below. You can click on the Dashboard tab on the Bluemix console and then on the API connect service as shown below.



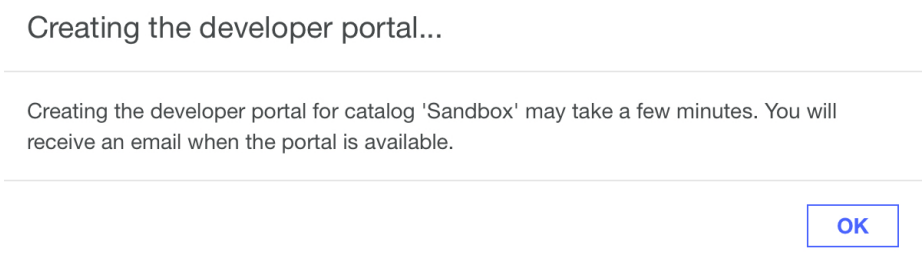
This will bring up the Dashboard for the API Connect manager as shown below.



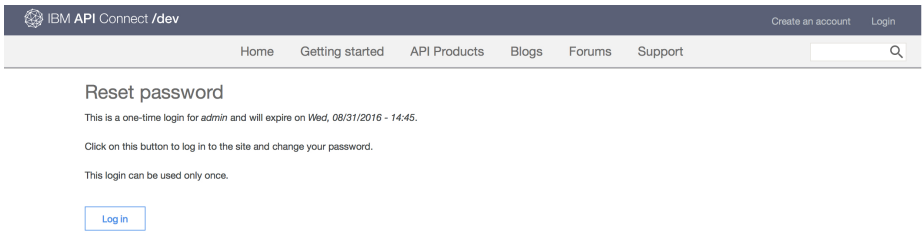
Select the “Settings” tab, pick “Portal” and pick the “IBM Developer Portal”. Finally click Save icon on the top right as shown below.



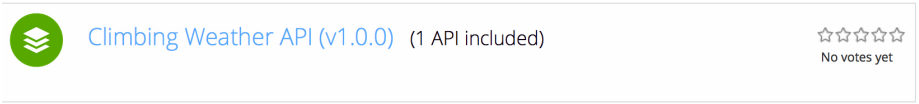
Wait for the email to arrive that the developer portal is ready as the dialog box below indicates. **This might take a few minutes.**



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



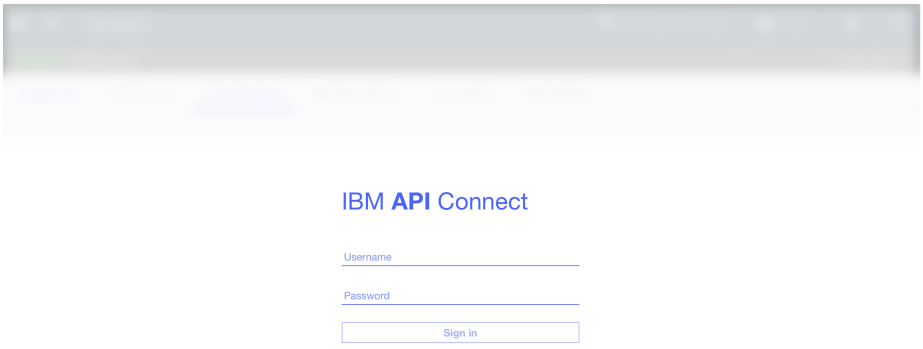
You will see the “Climbing Weather API” as seen below.



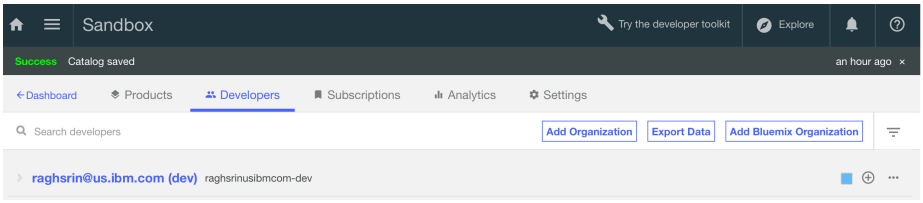
We will next publish this API to Bluemix.

Publishing the Sample API to Bluemix

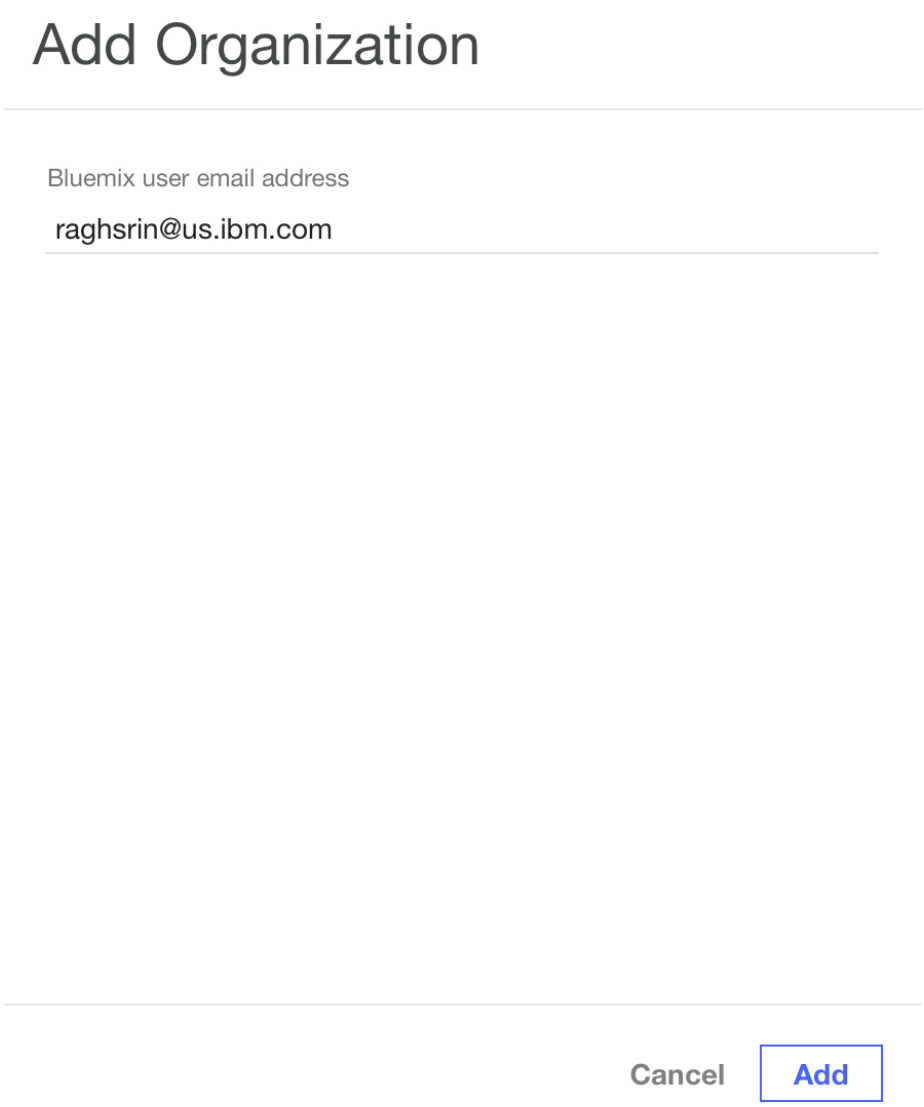
We will go back to the API Connect Manager. If required, re-login to the API Connect Manager as shown below.



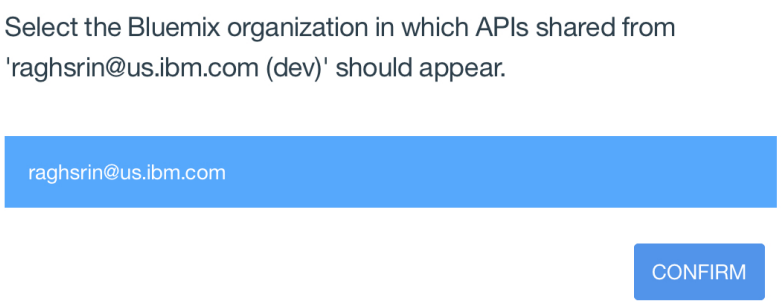
In the API Connect Manager on Bluemix click on “Developers” in the navigation pane and “Add Bluemix organization” as shown below.



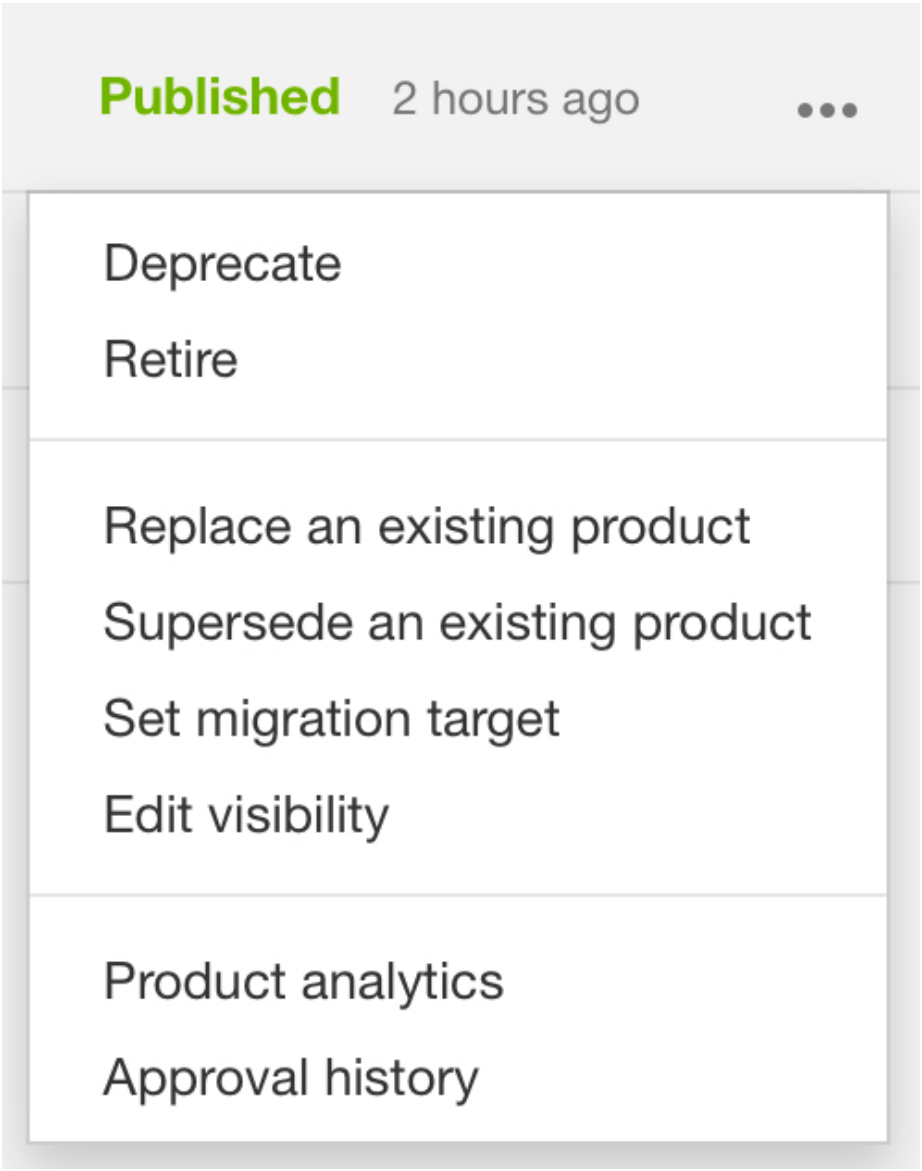
Provide the email address substituting the address with your Bluemix email address as shown below.



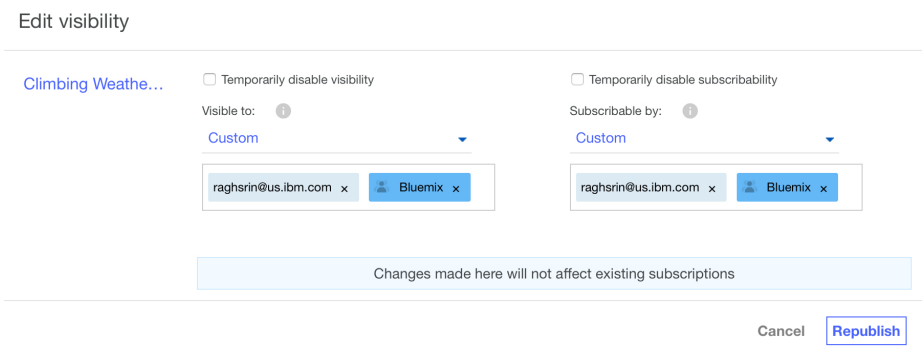
Select the Bluemix org and hit “Confirm” as shown below.



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”.



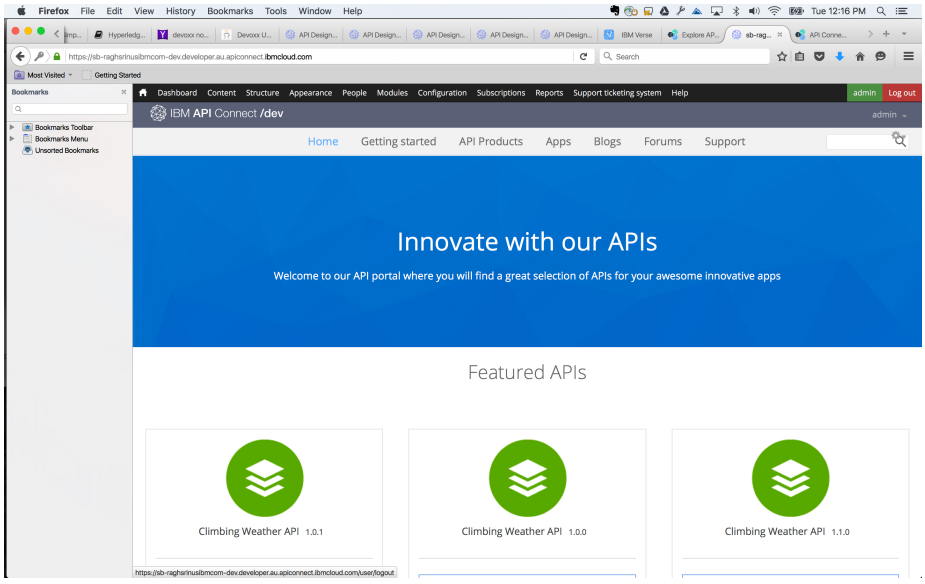
Change the visibility to “Custom” and add the organization (substituting your own organization) and Bluemix. Finally hit “Republish” as shown below.



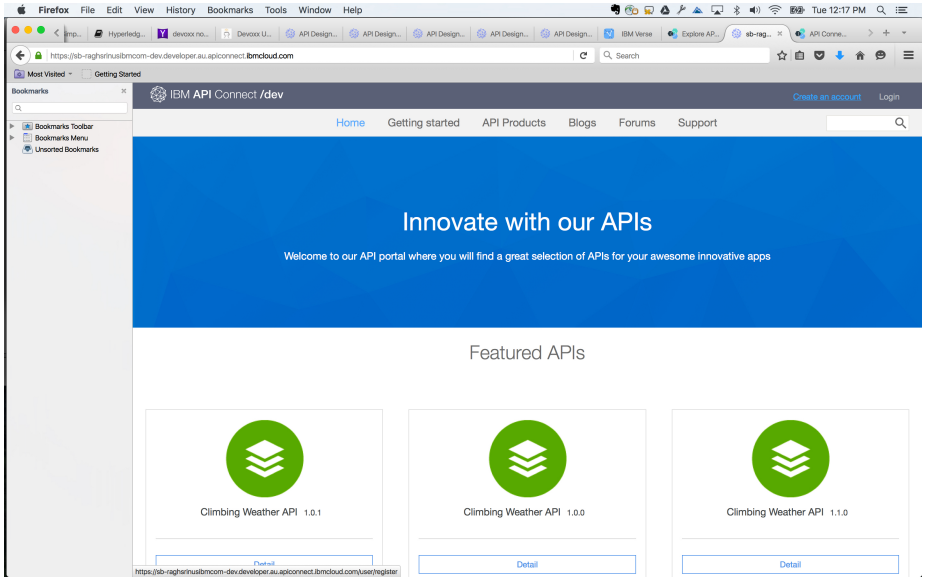
The Climbing Weather API product is displayed in the Explore APIs tab of the API Connect Dashboard. If you click, Climbing Weather App 1.0.0 it will take you to the API in the Developer Portal.

Registering an app for the sample API in the Developer Portal

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

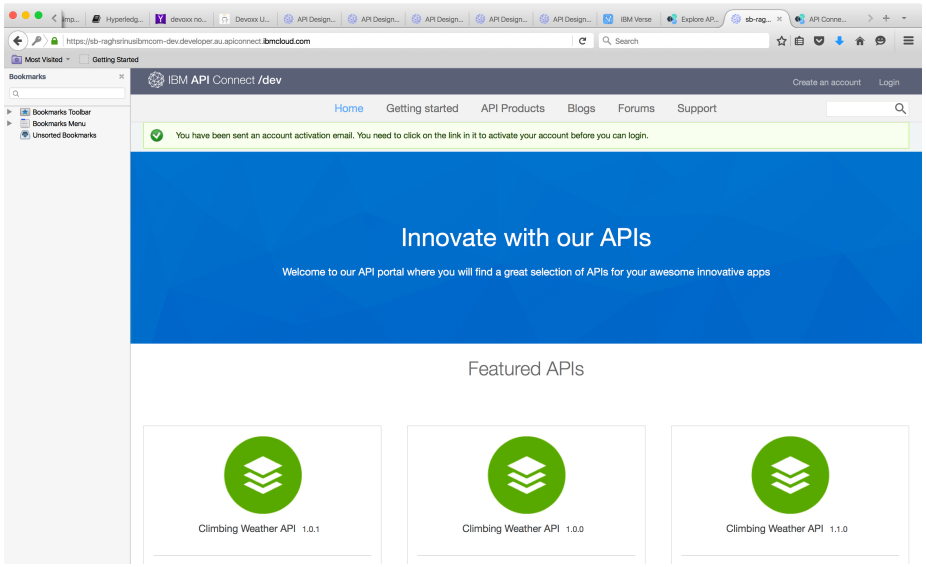


Click on “Create an account” as shown below.

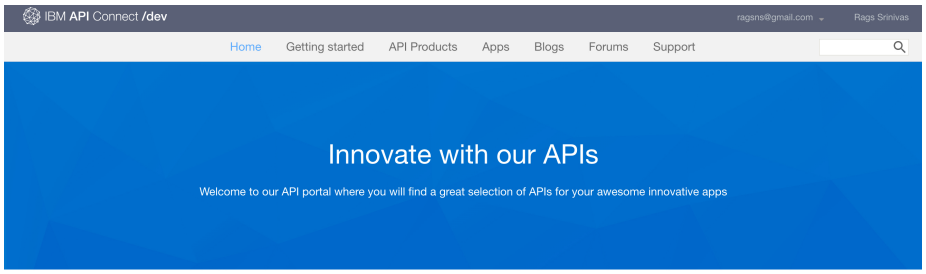


Complete the form and ensure that you use a different email address.

You should see an account ativation complete as shown below.

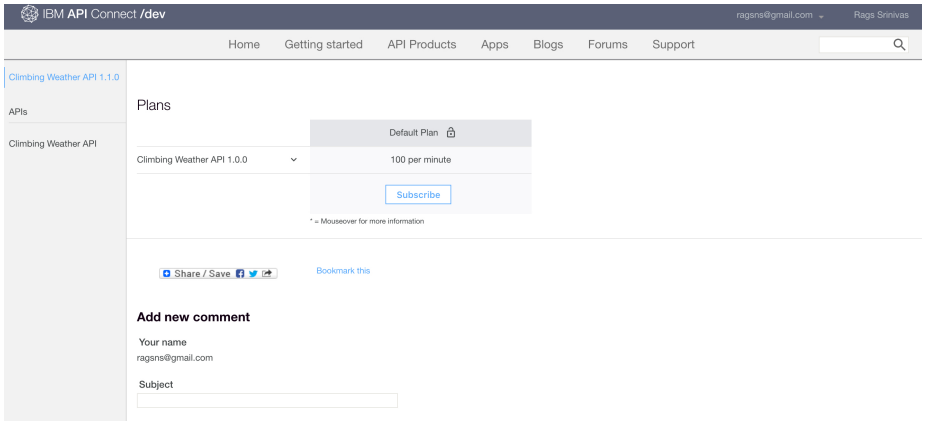


After the email arrives at this second login with the account details, click on the link which will bring up the portal as below.

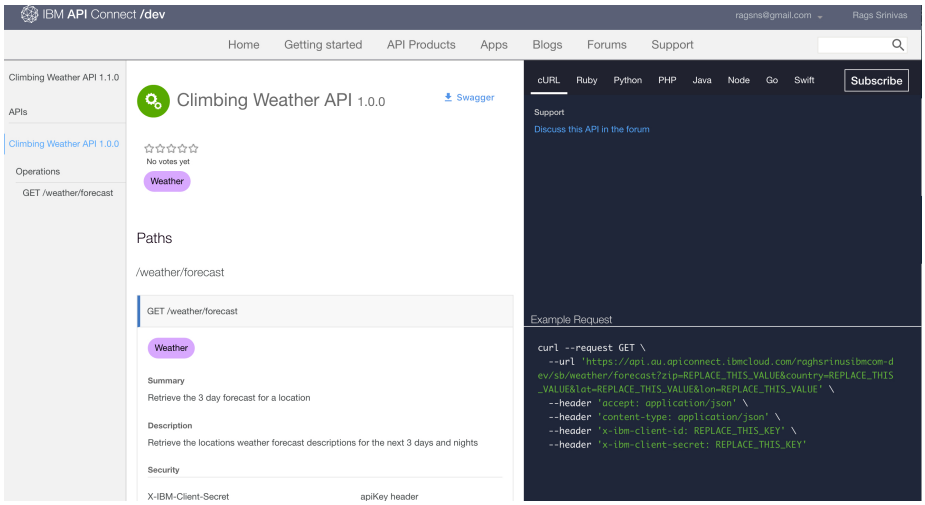


Featured APIs

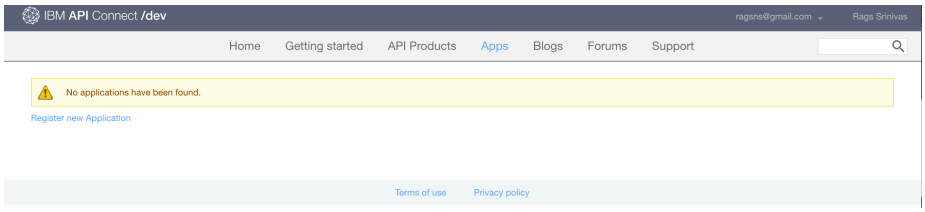
Click the API Products and Climbing Weather API. You can see that Climbing Weather API belongs to a default Plan with a rate limit of 100 calls per minute as shown below.



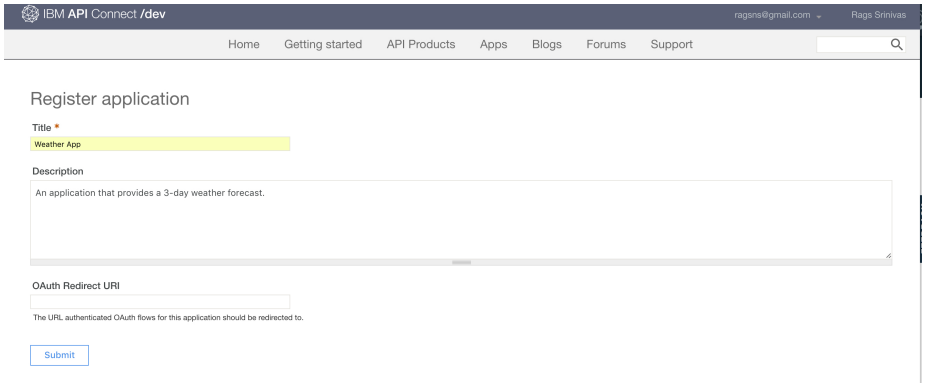
Click on the “Climbing Weather API 1.0.0” tab as shown below.



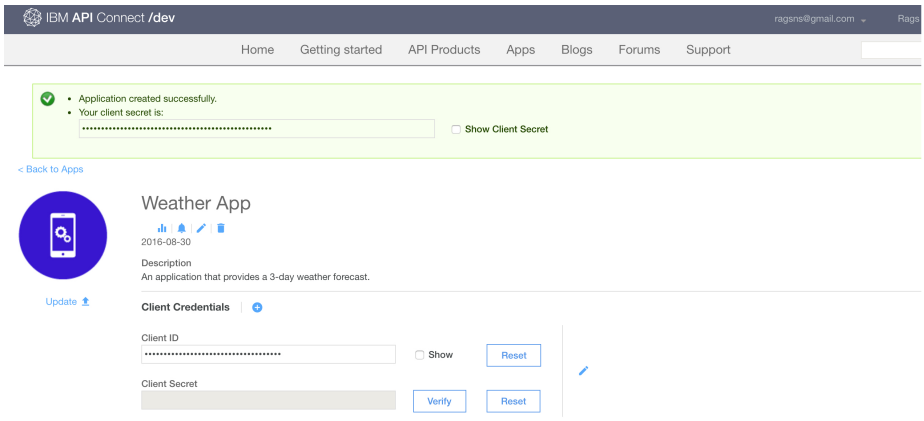
Click on the “Apps”navigation pane of the Developer Portal, click “Register a new Application” as shown below.



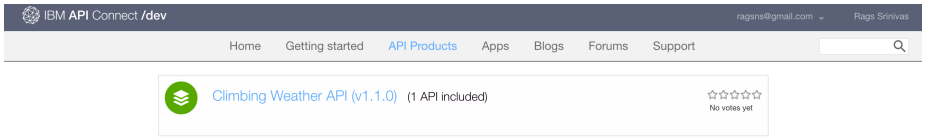
Fill in the details and hit “Submit”.



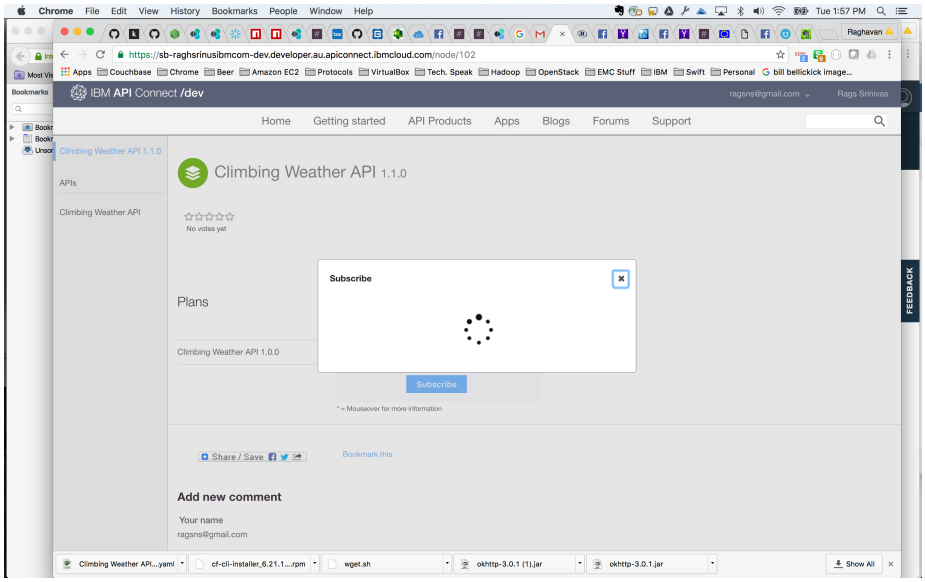
Click Show to display the “**Client secret**” that we will use subsequently as shown below.



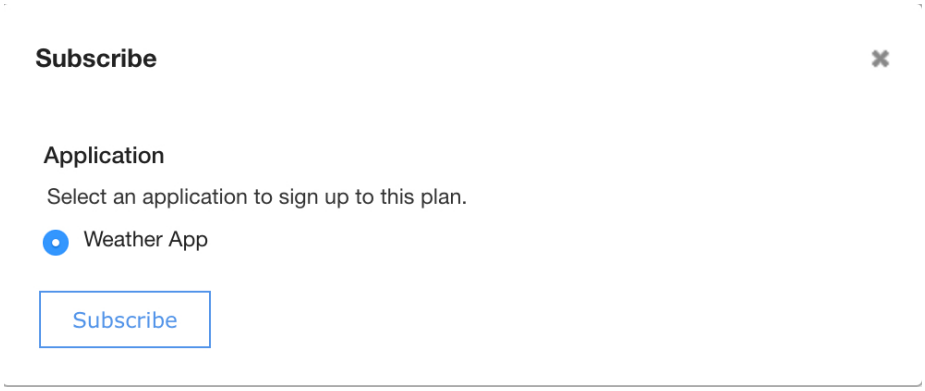
Click “API Products” and the “Climbing Weather API” as shown below.



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



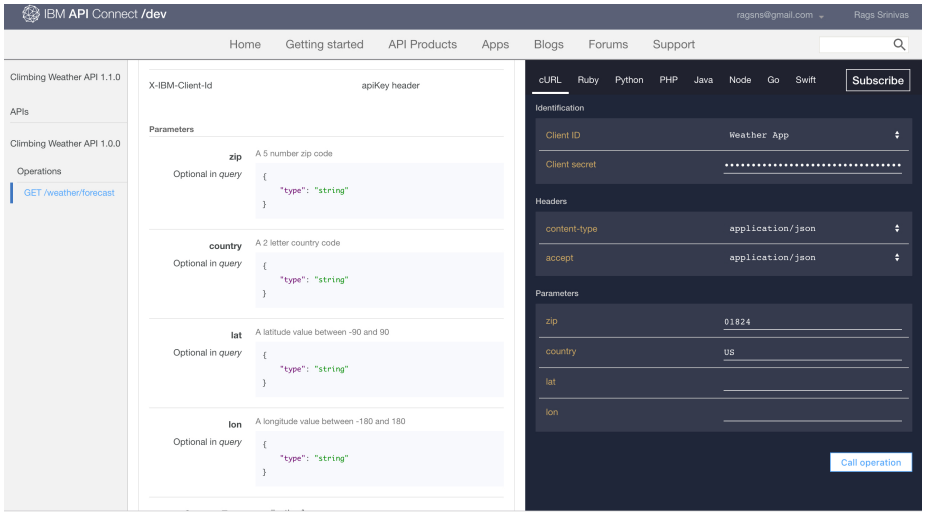
This will bring up a dialog box. Click on “Subscribe” as shown below.



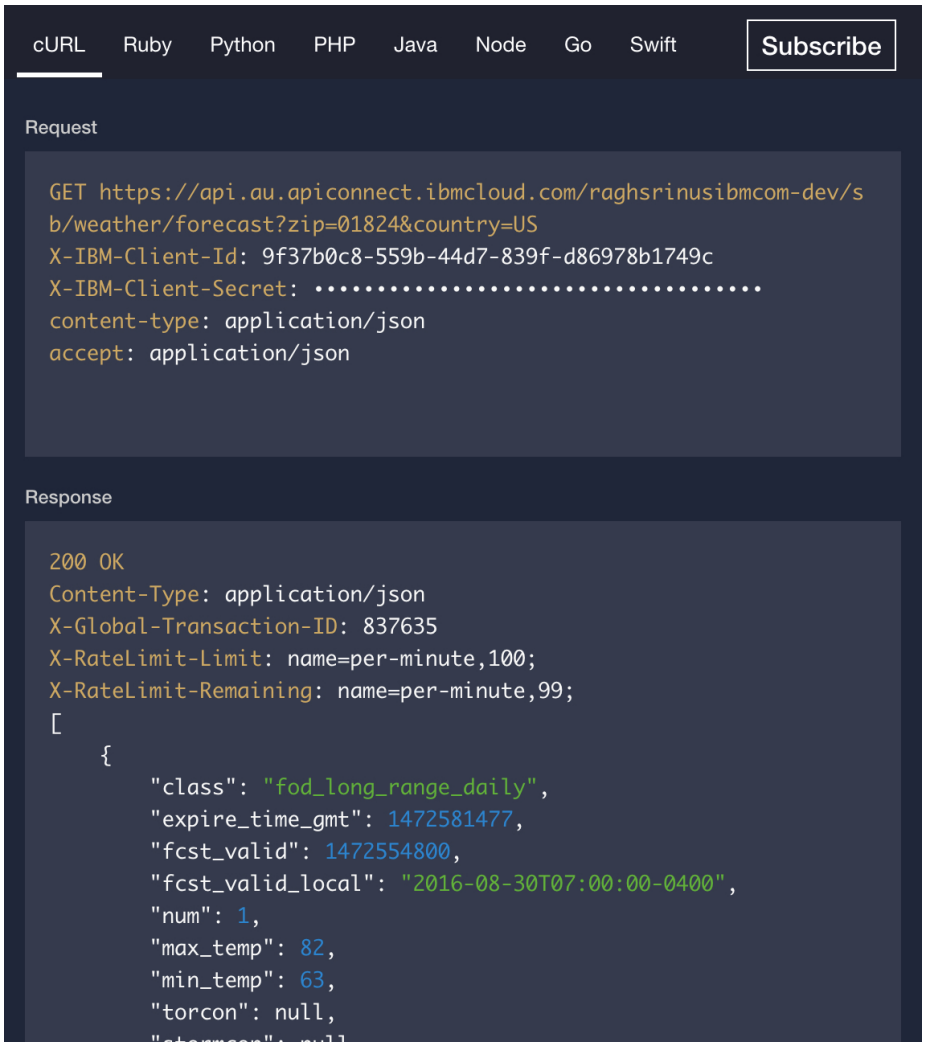
You have registered to the Climbing Weather App application and subscribed it to a Plan.

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".



That should provide a response which is shown below.



Summary of exercise and next steps

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.

You could follow a similar journey to publish your own APIs aroud your own user-generated data. You can use the variety of connectors availble via API Connect.

Final Summary

In conclusion, we covered the usage of the API Connect Manager locally and on Bluemix, how to create models locally, publish them on Bluemix and explore the APIs.

Finally we published the APIs on a self-service developer portal which provides API-based services including providing service keys, throttling, etc. without having to write any code.

Congratulations! You are done!!