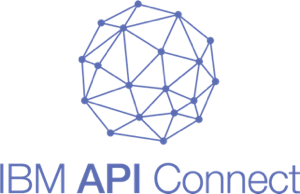
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic_banner-mini.png)

**Introduction**

In this lab, you’ll gain a high level understanding of the architecture, features, and development concepts related to the IBM API Connect (APIC) solution. Throughout the lab, you’ll get a chance to use the APIC command line interface for creating LoopBack applications, the intuitive Web-based user interface, and explore the various aspects associated with solution’s configuration of REST API as well as SOAP APIs.

**Note**: This lab is based on version 5.0.7.0 It will be updated as much as possible to follow the new versions of API Connect. Many new features have been announced at InterConnect 2017. [Statement of directions - 2017 March 14th](https://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/2/897/ENUS217-152/index.html&lang=en&request_locale=en#abstrx).

For any comments, please send an email to [arnauld\_desprets@fr.ibm.com](mailto:arnauld_desprets@fr.ibm.com) (Arnauld Desprets)

**Objective**

In the following lab, you will learn:

* Goals of API Connect, main use cases (Presentation)
* Basics on the architecture of the API Connect and terminology useful with API Connect (Presentation)
* Installation (Presentation)
* How to create and test a REST API definition (Lab)
* How to publish an API to Bluemix (Lab)
* How to subscribe to an API previously published and test in the portal (Lab)
* How to create a simple LoopBack application to implement Microservice architecture (Lab)
* How to version an API and deploy another version of an API (Lab)
* How to create and manage a SOAP API (Lab)
* Basics about the command line apic to script recurring operations (Lab)
* Transformation JSON to SOAP (Lab)
* API security (pres)

**Pre-Requisites**

* Sign in to [Bluemix](https://bluemix.net) and set up your IBM ID
* Install IBM Node SDK 6.x [Node.js](https://developer.ibm.com/node/sdk/v6/)
* Install (or Update) API Connect Toolkit [API Connect Developer Toolkit](https://www.npmjs.com/package/apiconnect) by issuing the following command:
* npm install -g apiconnect --no-optional --ignore-scripts

You must have version 5.0.7.0 or above. check the version with the following command :

$apic -v

API Connect: v5.0.7.0 (apiconnect: v2.6.2)

**Note**: To check all available versions of API Connect: npm view apiconnect version To check the local version of API Connect: apic -v

**Goals of API Connect, main use cases**

This chapter does not intend to describe all the possible use cases of API Connect, but instead provide some simple and concrete common usages of API Connect.

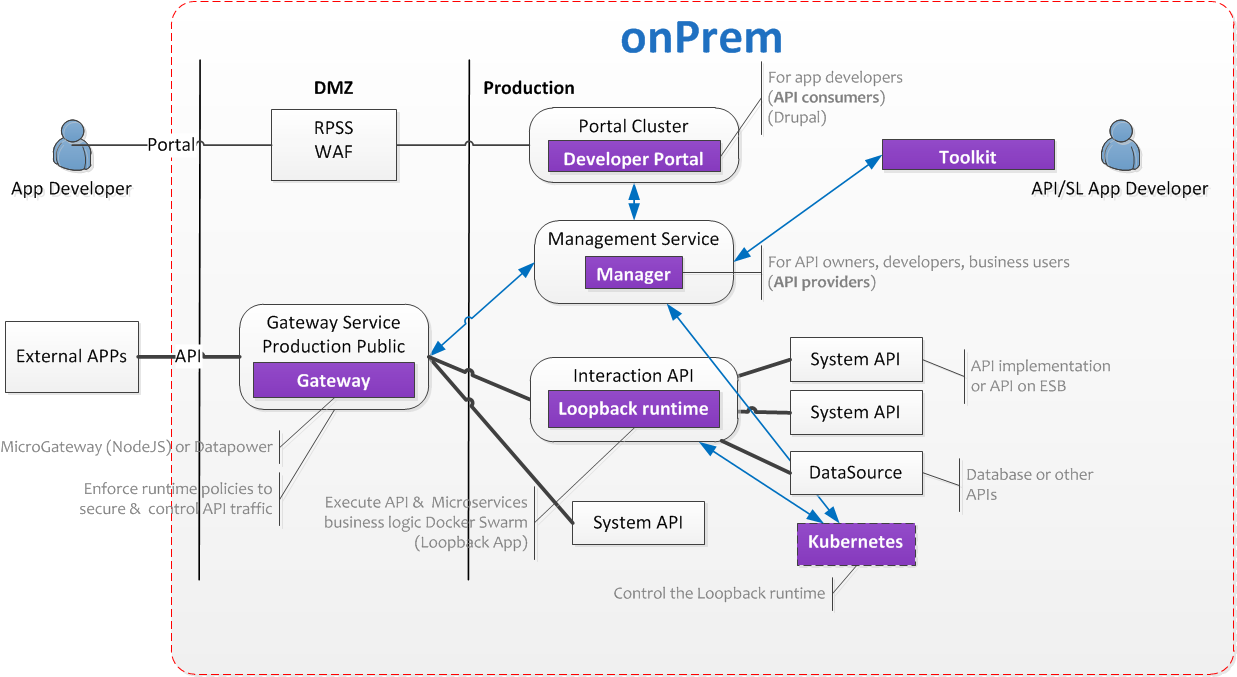
1. **Use case 1**: I have existing internal SOAP services and/or REST APIs. I want to expose and increase visibility internally and externally. I need to understand how my APIs/Services are used and apply quotas. I need to provide to secure the access.   
   Solution: Simple proxyfication, not complex policies, use OOTB portal, manager.
2. **Use case 2**: All the above + my APIs/services do not have the right granularity or the right format to be used by my Apps.   
   Solution: Use map policies to adapt the interfaces, and/or use JSON <-> XML policies with a powerful versioning management.
3. **Use case 3**: I organise a hackathon or I'm in context of co-creation with extended eco-system and I need to rapidly create APIs from data sources or from models.   
   Solution: Create Loopback Applications and expose them as APIs. Cloud workload are good candidates.
4. **Use case 4**: I need some kind of composition/aggregation and expose an API.   
   Solution: Create a Loopback Application and add remote hook

**Architecture and terminology**

**API Connect architecture onPremise**

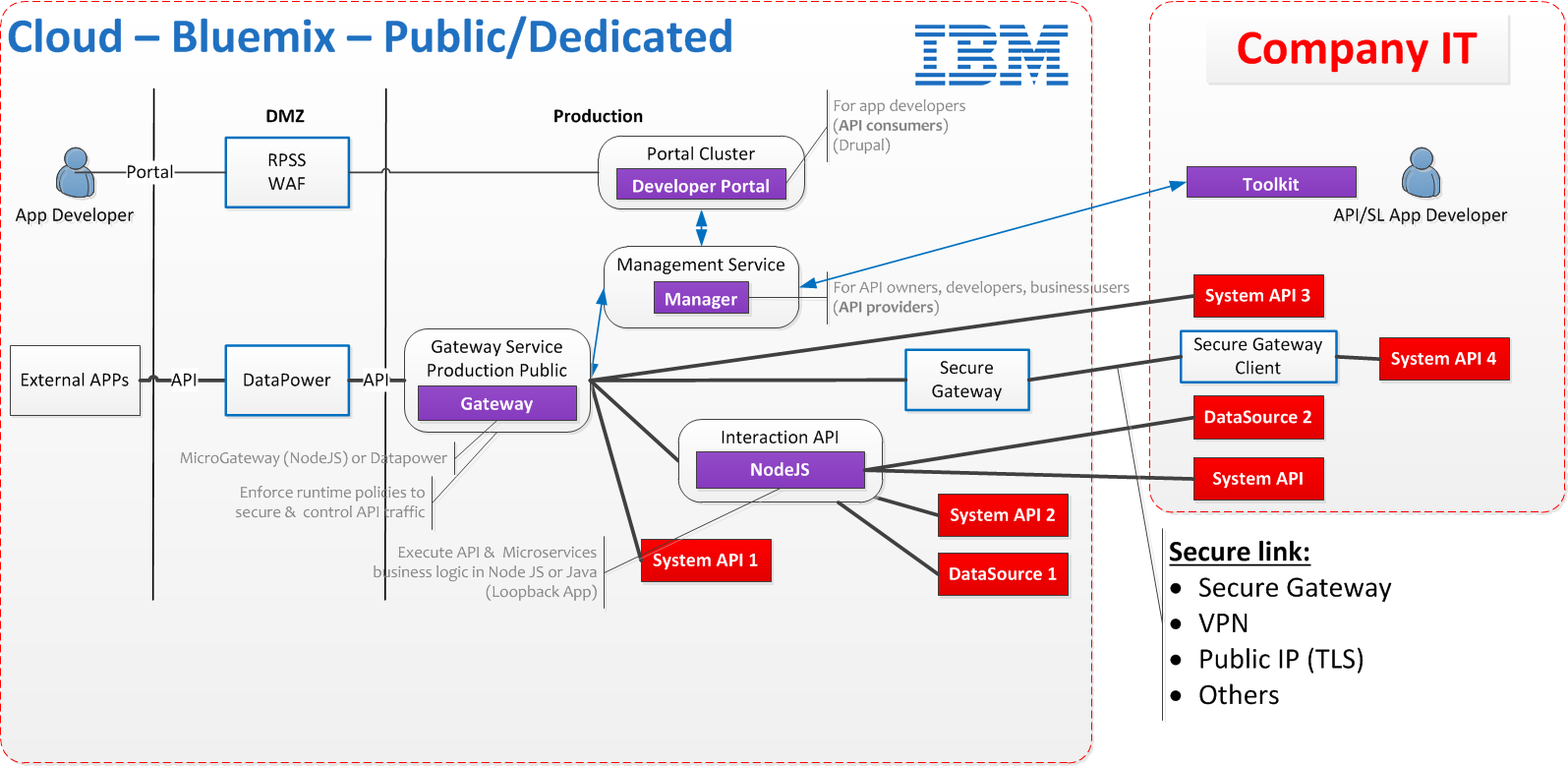
The main components composing API Connect are:

* The **Gateway** (either DataPower, either a NodeJS implementation called micro gateway in this case). The requests from apps are going through the gateway, policies are enforced and analytic are gathered.
* The **Manager** where the APIs are defined and governed. It also collects the analytics from the gateway. The manager can be used directly or more likely using the toolkit.
* The **Portal**, an open source Drupal CMS - Content Management System. For the API consumers (Apps developers), they create Apps and subscribe to API within the portal. Based on Drupal, it is highly customizable.
* The **Loopback runtime** or micro services runtime. This is where the loopback applications are running. This component is originally coming from StrongLoop acquisition. Loopback applications can be created in minutes to expose data from SQL or NoSQL database and also a good place to perform composition of APIs.
* Associated to the Loopback runtime is the **Kubernetes** that monitors the Loopback runtime and can provide advanced feature such as auto-scaling.
* The **Toolkit**, running on the API developer, it offers the same web experience as the manager to manage APIs. But this is also the only place where you can define Loopback applications. It also contains CLI to operate directly on the manager whether it is an onPremise version or Bluemix version of API Connect.

Below a sample of deployment of API Connect on premise. System API is a generic term to define an API implementation, for example running in WAS Liberty (JAX-RS) or an API exposed on another layer such as an ESB. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-archi-on-prem.png)

**API Connect architecture in Bluemix**

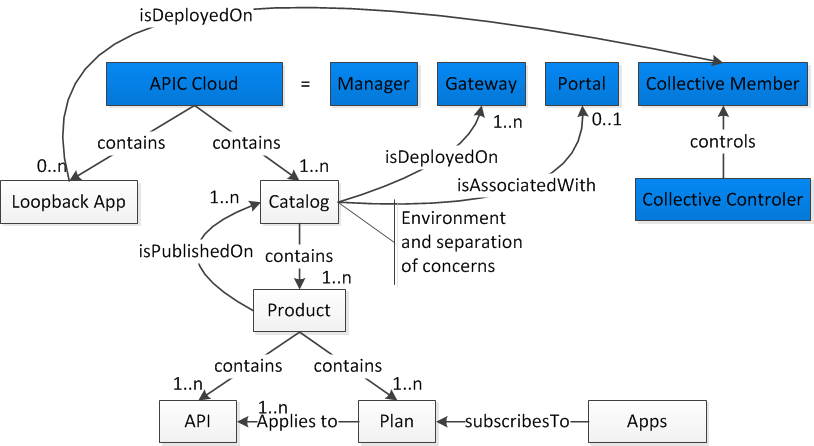
Well, API Connect in Bluemix, is the same product as onPremise (use of NodeJS for the loopback application). This gives us a competitive advantage, because we have no issues at all to move from onPremise to Bluemix and vice-versa and have hybrid implementations.

Notice the **secure gateway** component that allows to connect simply IBM Bluemix and the company IT. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-archi-bmx.png)

**Terminology**

* An **API**: Can be SOAP or Representational State Transfer - REST API defined with an Open API definition (Swagger) as a YAML file. One API = one yaml file though WSDLs and Schema are separated in a zip file for a SOAP API.
* A **Plan**: this is where we specify the quotas and if an approval is needed to subscribe to a Product/API.
* A **Product**: this is an aggregation of APIs, and one or many plans associated to those APIs. This is what is published to a catalog. One Product = one yaml file.
* A **Catalog**: it's relates to a cluster of gateways and a portal. It sounds like an environment but it also contains a business dimension. For example, good names for a catalog are Sandbox, Dev, Production, CRM (for my CRM APIs exposed to a specific population), etc ...
* An API Connect **Cloud**: not to be confused with a cloud infrastructure/platform, it is a combination of gateways clusters, managers cluster, portal clusters and loopback applications runtimes. Usually a customer will have one, two, sometime three or more API Connect clouds, based on its organisation and needs to separate the infrastructures.
* **Assembly panel**: this is where we specify the policies to be executed in the gateway for each transactions.

**Concepts map**

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-ConceptsMap.png)

Below the available policies in API Connect 5.0.6

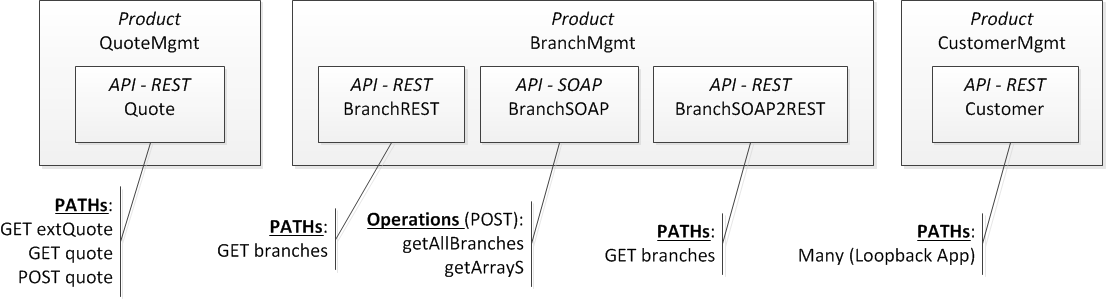
| **Category** | **Name** | **Description** | **-** | **Category** | **Name** | **Description** |
| --- | --- | --- | --- | --- | --- | --- |
| Logic | Switch | switch | - | Transform | Set variable | set |
| Logic | Operation Switch | on op | - | Transform | Validate | JSON schema |
| Logic | If | if | - | Transform | Gateway script | any code |
| Logic | Throw | exception handling | - | Transform | XML to JSON | transform |
| Transform | Redac | obfuscate | - | Transform | JSON to XML | transform |
| Transform | Map | mapping | - | Security | Validate JWT | JSON Web Token |
| Transform | XSLT | any code | - | Security | Validate UNT | UserName Token |
| Transform | Invoke | invoke | - | Security | Generate JWT | JSON Web Token |
| Transform | Proxy | proxie | - | Security | Generate LTPA | LTPA (for WAS) |
| Transform | Activity Log | audit | - | User Defined | Any | any logic coded in DP |

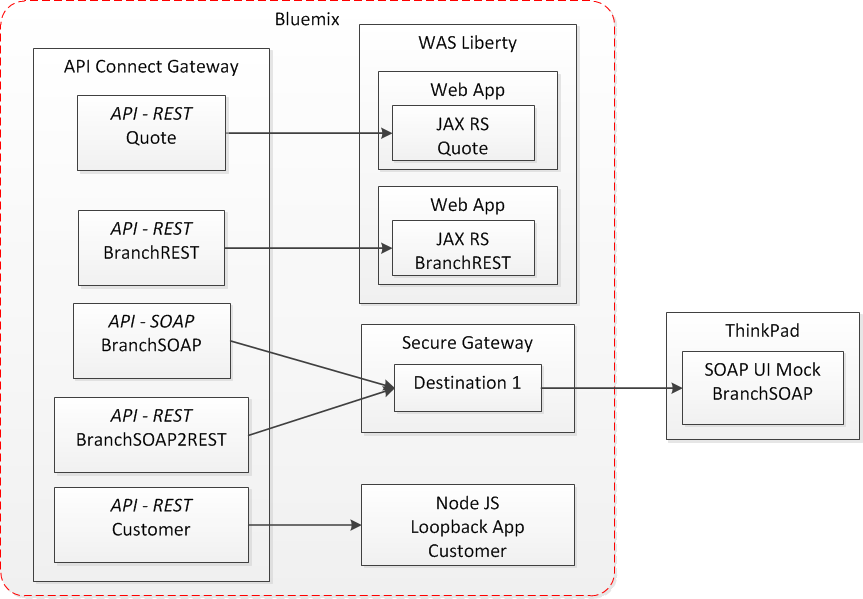
**Installation**

This chapter just illustrates one way to install the product in a very simple case. There are many more options to deploy install API Connect including full docker installation. The goal here is just to give a feeling on the installation process.

* Install the gateway   
  Start the OVA in VMWare, install firmware update if needed to DataPower, adjust network settings.
* Manager   
  Start the OVA in VMWare, adjust network settings, configure gateway access and configure portal.
* Portal   
  Start the OVA in VMWare, adjust network settings and exchange keys so manager can interact with portal.
* Toolkit   
  On Windows install api-connect toolkit using npm
* Loopback application   
  Install controller (WAS liberty) + IHS + member (nodeJS) and configure IHS Plugin

**Steps for the lab**

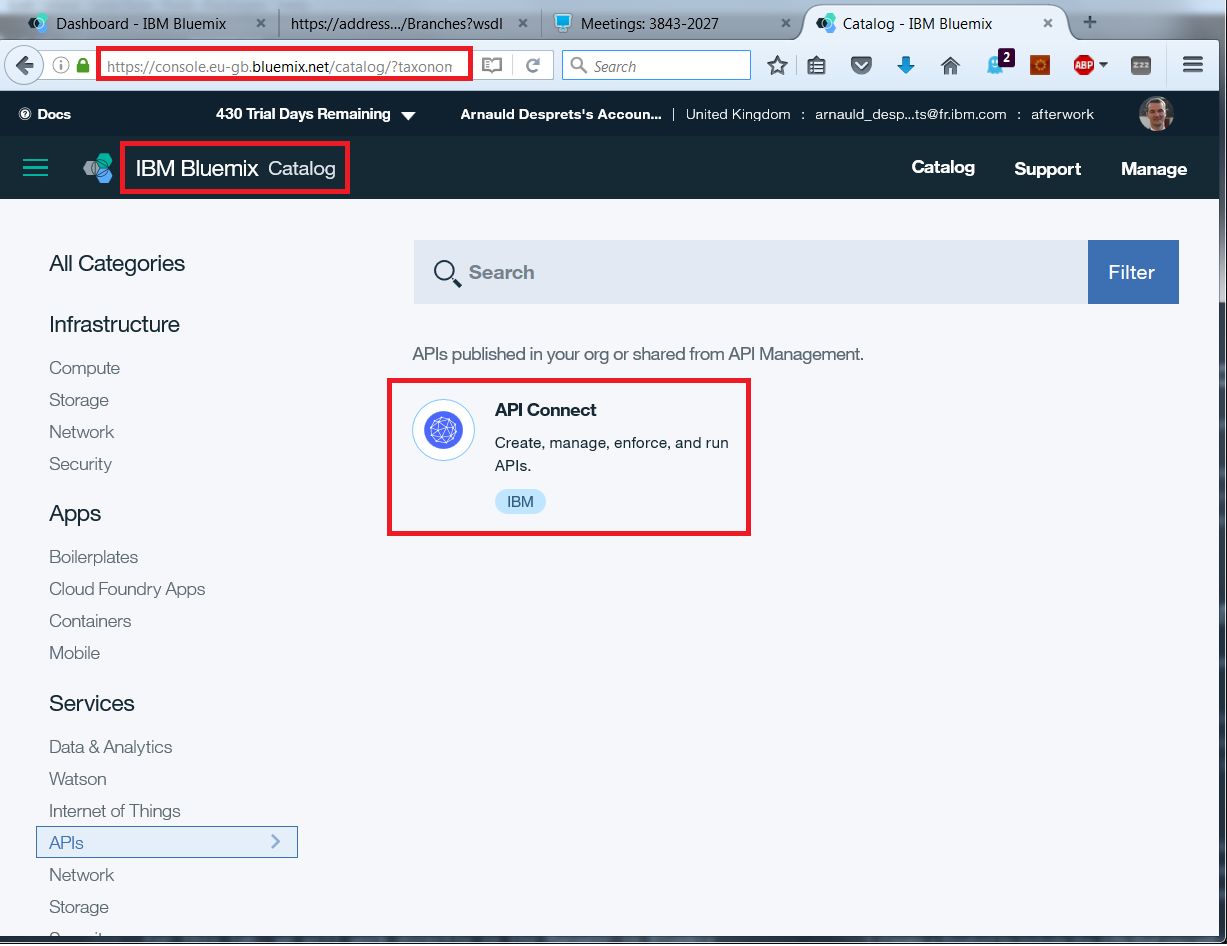
The goal of this lab is to define the following APIs: [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/ProductsAPIsToBuild.png)This provides a mix of REST and SOAP APIs, with or without mapping, using or not a Loopback Application.

The corresponding back end runtimes are as follow: [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/RuntimesAndBackEnd.png)This provides a mix of JAX RS, NodeJS (with Loopback Application) and Secure Gateway back end samples, demonstrating pure cloud APIs and hybrid APIs.

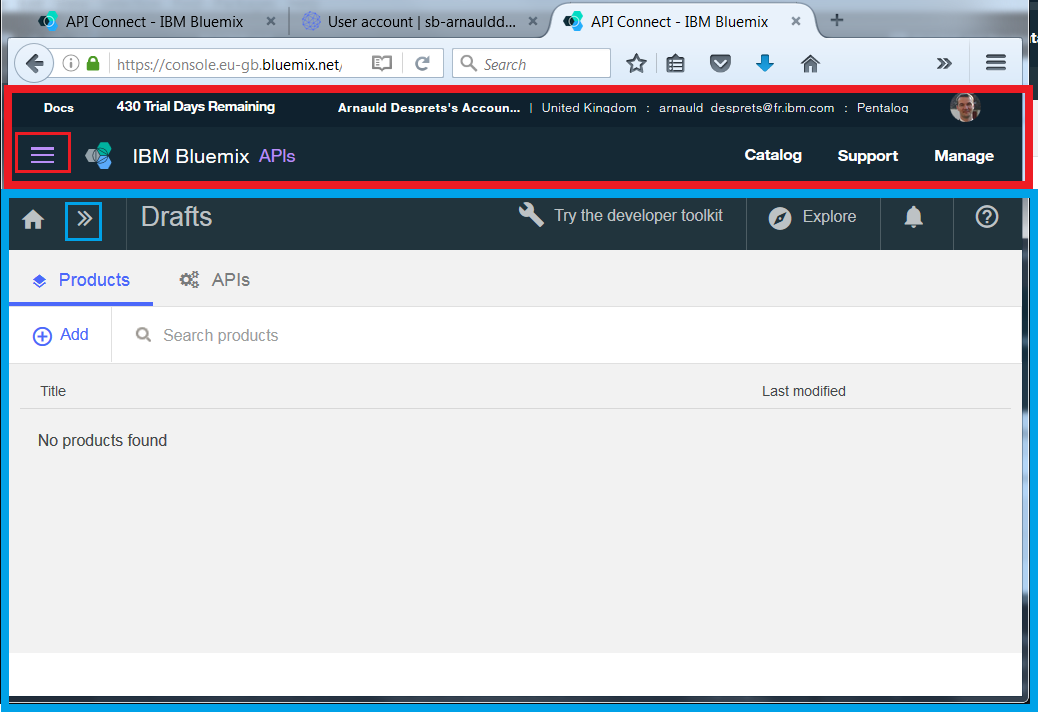
1. [Provision API Connect in Bluemix](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-1---provision-api-connect-in-bluemix)
2. [Expose an existing REST API](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-2---expose-an-existing-rest-api)
3. [Publish your API to the Sandbox catalog](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-3---publish-your-api-to-the-sandbox-catalog)
4. [Consumer Experience](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-4---consumer-experience)
5. [Invoke the API](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-5---invoke-the-api)
6. [Analytics](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-6---analytics)
7. [Create a SOAP API](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-7---create-a-soap-api)
8. [Create a SOAP to REST API](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-8---create-a-soap-to-rest-api)
9. [Create a Cloudant service](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-9---create-a-cloudant-service)
10. [Create a LoopBack application](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-10---create-a-loopback-application)
11. [Manage your API in API Designer](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-11---manage-your-api-in-api-designer)
12. [Test your API](https://github.com/ADesprets/bluemix-labs/tree/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect#step-12---test-your-api)

**Step 1 - Provision API Connect in Bluemix**

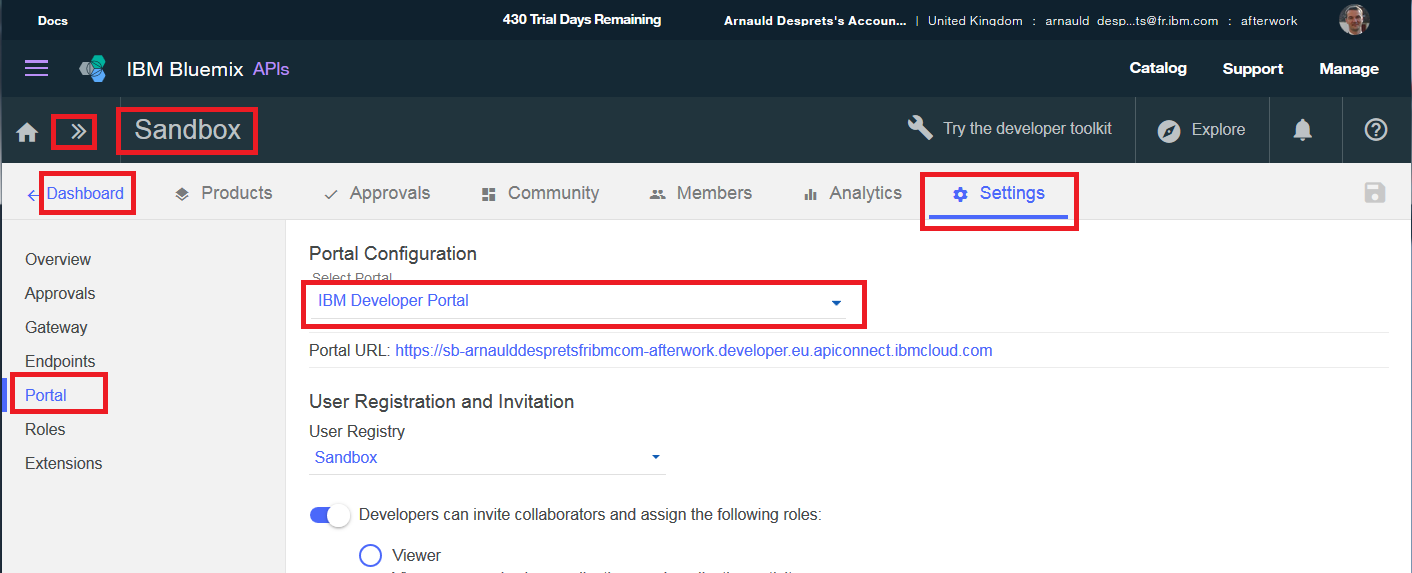
Login to the Bluemix [Catalog] [bmx\_catalog\_uk\_url](https://console.eu-gb.bluemix.net/catalog/), in the UK region, provision (create) an instance of the service **API Connect**.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/bluemix-createapic.png)

When the API connect instance is created in your space for the organization, you should get the following screen. Notice that you can create only one instance of API connect per bluemix space.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/bluemix-apiccreated.png)The bold red box belongs to Bluemix, the Hamburger (small red box) give menus specific to Bluemix.

The bold blue box belongs to API Connect, the chevrons (small blue box) gives menus specific to API connect.

We also instantiate the Portal associated to the Sandbox catalog. Click on the API Connect instance from your catalog, Go to *Dashboard* (within API Connect), click on the Sandbox catalog, then on the menu bar, click on *Settings*, then on the navigation panel, click on Portal, and select *IBM Developer Portal*, click the Save icon. This will instanciate an instance of a site within Drupal, when it is completed you will receive an email. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/bluemix-configportal.png)

**Step 2 - Expose an existing REST API**

For this lab, we will work directly in the Manager in Bluemix instead of using the toolkit.

**Note**: Using the toolkit (locally) or using manager directly (remote server) is a pretty important decision. Using the toolkit allows to use a Source Control Management system and perform micro versioning as well as backup of the various yaml (and wsdls). It also provides a local experience with a very low response time. Using the Manager simplifies sharing the API Drafts. In reality, there are ways to benefit of both approaches.

In this first step, we assume that a developer of an API is providing you the Swagger associated with that API. The developer is using WAS Liberty as the runtime and he also uses JAX-RS annotations along apidiscovery feature. This allows him to get a Swagger easily consumed by API Connect. Download the Swagger [here](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/materials/step2/QuoteManagementAPI_AW_S.yaml) on your file system, for example under C:\IBM\Afterwork

1. Create a Product to include the APIs that you will create in this lab. Within your instance of API Connect in your space, click on Draft > Products and the Add button, select New product. In the *Title* section, enter **QuoteMgmt**.
2. Now we can create the API that will be included later in the Product just created. Within your instance of API Connect in your space, click on **Import API from File or URL**. This is accessible from Drafts > APIs. Specify the location of the Swagger you just downloaded. Click OK.
3. We need to complete/review a few informations, that were not specified in the generated Swagger. The amount of information that need to be completed will depend greatly on the use of the annotations or the Swagger generator used.

* Select https for the scheme, in the *Schemes* section.
* Create the security definition, click on + sign close to the *Security Definitions* section, and select API Key. Rename it to **cliend-id**.
* Specify the security of the API, click on the + sign close to the *Security* section, and check the client-id API Key
* Create a property to define the **target-url** of the back end API. This allows us to create a variable that may take different values based on the catalog. Click on the + sign close to the *Properties* section. Set the Property name to target-url, and enter [**http://SampleJAXRS20-aw.eu-gb.mybluemix.net**](http://SampleJAXRS20-aw.eu-gb.mybluemix.net) in the value, close to *Default* value.
* Click on the Assemble menu (or Edit ssembly button), click on the Invocation policy, and set the URL property to **$(target-url)$(request.path)$(request.search)**
* Save the changes, by clicking on the *Save* icon at the top.

1. We are almost ready to test the API. We need to add the API to the Product. You do this within the API by clicking on the ... icon on the top, and select *Add to existing products*, then select the *QuoteMgmt* and click on the Add button.
2. To test the API from an API provider perspective, click on *Assemble*, in the left hand side panel, switch from **Micro Gateway policies** to **DataPower Gateway policies**. Save the change. click on the triangle icon (on the top left), check the catalog to run the API, select the QuoteManagement product, then select the operation to test, for example, get /extQuote, enter the parameters (amount 1000, rate 1.1, duration 36, delay 10, msg length 11) and click invoke. (If you are using the toolkit you also can do this, but in this case you need to start the micro gateway and the loopback application runtime).

You can also test the API using the explore facility. You get a view similar to the API consumer in the portal, but in this case you do not need to create an Apps and subscribe to the API. Click on the *Explore* link on the top right, you see the documentation and have the possibility to test the various operations.

**Note**: Documenting an API is very important. API Connect supports the MarkDown format, which is popular in the GitHub sphere. Below some samples of usefull MarkDown instructions.

* Sample inline code api.instagram.com : use single quote around
* Sample of code block

some information

: use triple ' and separate with line

* Sample of URL without displaying the URL: [IBM](https://www.ibm.com) : use [text] and then (URL) no spaces
* Sample of table

| **Header1** | **Header 2** |
| --- | --- |
| Value 1 | Value 2 |

: use | to delimit columns, a line with | - | for the headers and a line before and after the table

You can perform the same set of operations for the BranchREST API and using a new product called BranchMgmt. The Swagger for the Branch API is [here](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/materials/step2/BranchRESTAPI_AW_S.yaml)

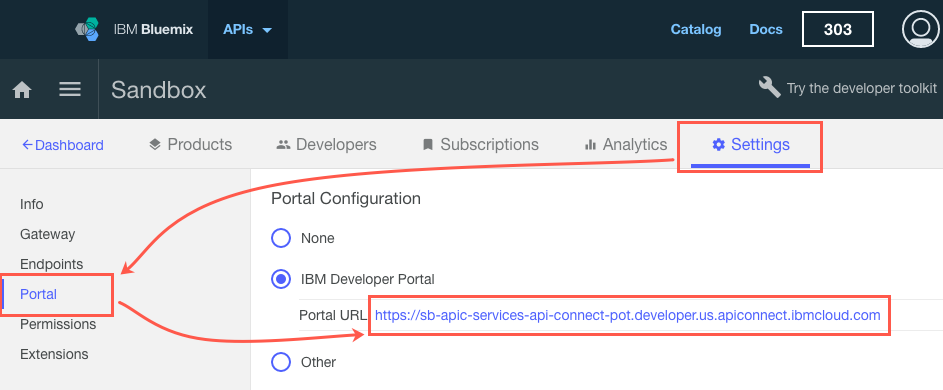
**Step 3 - Publish your API to the Sandbox catalog**

1. Within the Draft area, select the **QuoteMgmt** product, and click on **Publish icon** (cloud shape) in the top right corner, select Sandbox. This does effectively stage the product in the Sandbox catalog. The product is not yet published to the Portal.
2. Go to the Bluemix [Dashboard] [bmx\_dashboard\_url](https://console.eu-gb.bluemix.net/). Click on the Sandbox catalog, you should see the product just staged. Click on the ... link and select Publish.
3. Check the visibility and click on Publish button.
4. Your QuoteMgmt is now visible in Developer Portal

**Step 4 - Consumer Experience**

Summary: In this step, you will learn the consumer experience for APIs that have been exposed to your developer organization. You login as a developer to register your application and then subscribe to the product just published and then test the API included in the product.

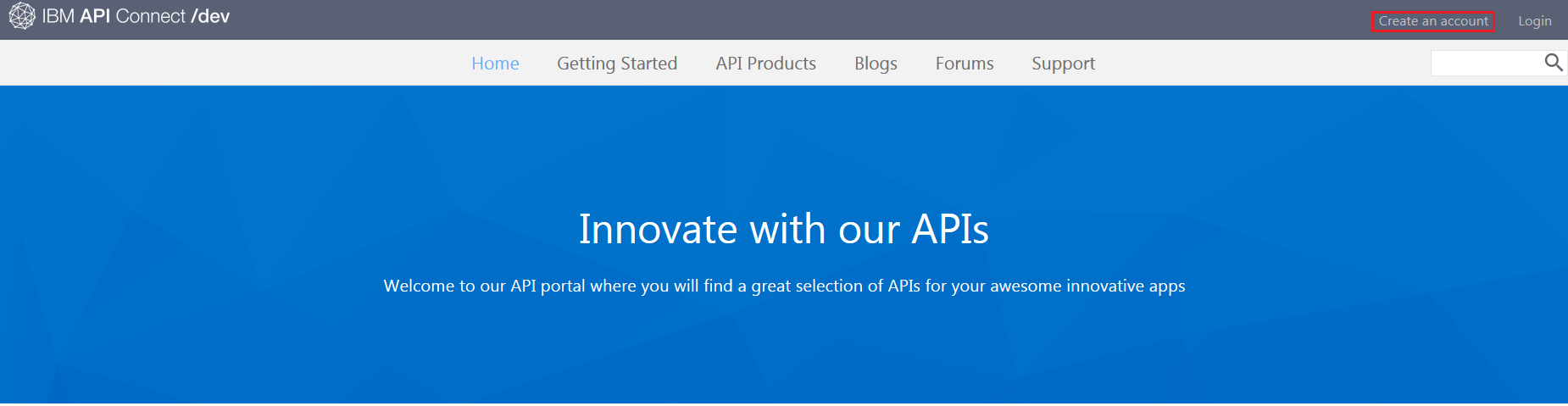
**Open the Portal login page**

1. Return to the Bluemix API Manager screen.
2. Navigate to the Dashboard section and click on the Sandbox catalog tile.
3. Choose the Settings tab, followed by the Portal option.
4. Click on the Portal URL link to launch the Developer Portal [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/launch-portal.png)
5. Click on **API Products** to explore the available APIs

**Register an Application as a developer**

Let's now subscribe to the Product. You will log into the portal as a user in the application developer role, then register an application that will be used to consume APIs.

If you have not created a developer account, you will need to use the **Create an account** link to do so now.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-portal.png)

1. Enter in your account information for the developer account. This must be a different email address than your Bluemix account. Click **Create New Account** once all the requisite data in the form has been filled out.
2. A validation email will be sent out to the email address used at sign up. Click on the validation link and then you will have completed the sign up process and will be authenticated into the page.
3. Login into the developer portal as an application developer using your developer credentials.
4. Click the Apps link, then click on the **Create new App** link.
5. Enter a title and description for the application and click the Submit button.

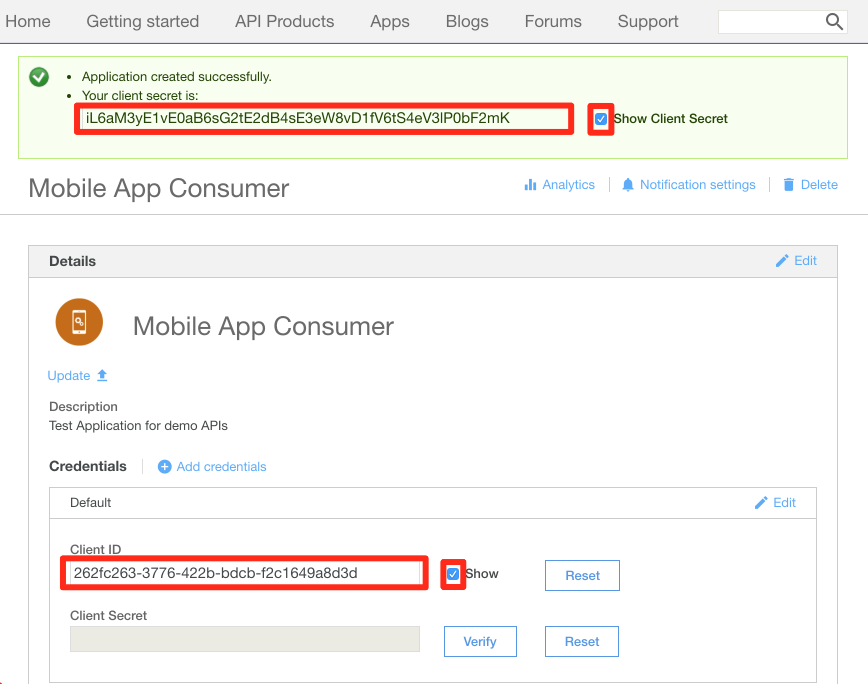
Title: Mobile App Consumer

Description: Test Application for the various API products

OAuth Redirect URI: < leave blank >

We need to capture the Client Secret and Client ID in a text editor for later use by our test application.

1. Select the Show Client Secret checkbox next to Client Secret at the top of the page and the Show checkbox next to Client ID.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-registerapp.png)

1. **Copy Client Secret and Client ID in a text editor**

**Subscribe to a Plan for the "QuoteMgmt" product**

In this section, we will subscribe to a plan for the "QuoteMgmt" using the Mobile App Consumer application.

1. Click the API Products link.
2. Click the QuoteMgmt (v1.0.0) API product link.

You will be directed to the Product page which lists the available plans for subscription.

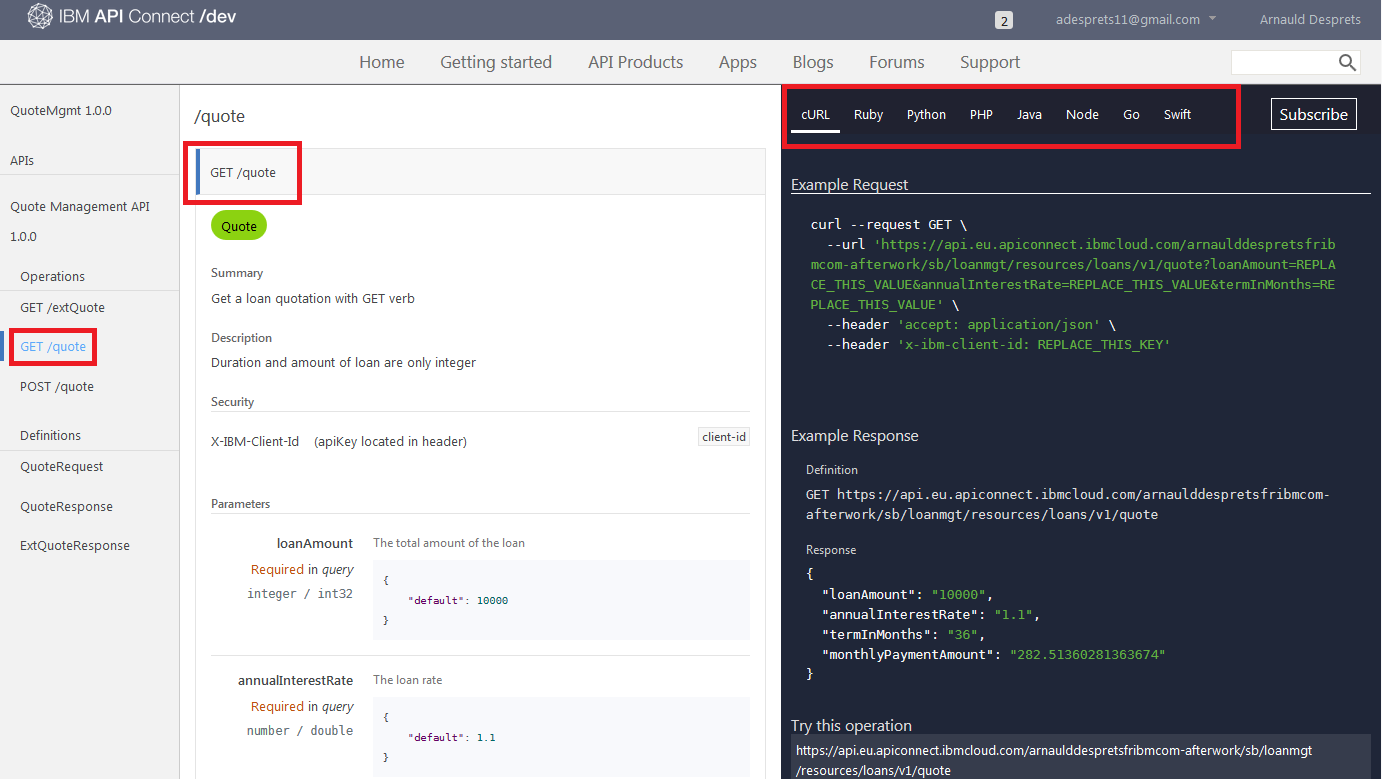
1. Click on the **Subscribe** button under the ***Default plan***.
2. Select the **Mobile App Consumer** toggle and click the **Subscribe** button.

The MobileApp Consumer application is now subscribed to the **Default plan** for the QuoteMgmt product.

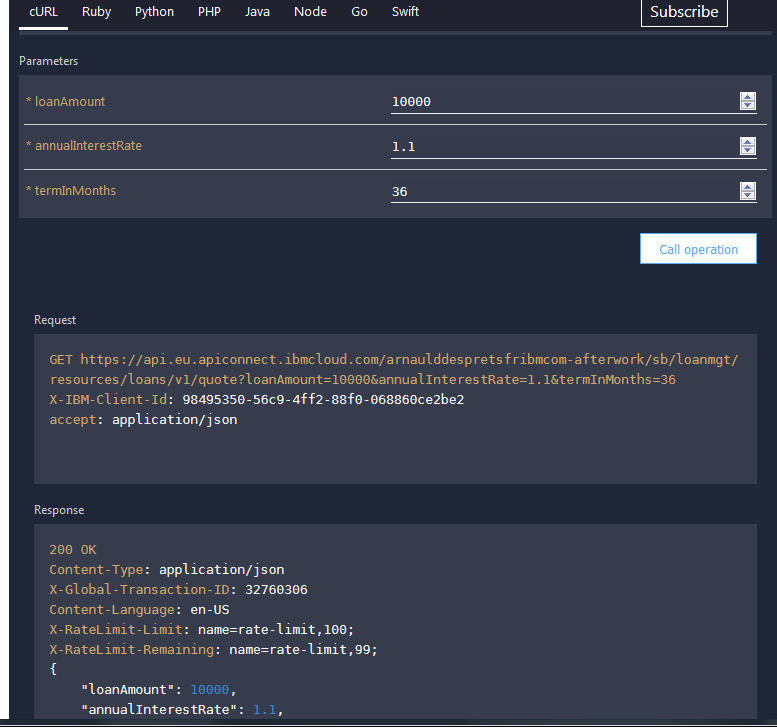
**Test QuoteMgmt APIs from the Developer Portal**

In this section, we will use the developer portal to test Quote Management API REST API. This is useful for application developers to try out the APIs before their application is fully developed or to simply see the expected response based on inputs they provide the API. We will test the **Quote Management API REST** API from the developer portal.

1. Click the **Quote** link on the left-hand navigation menu and then expand the GET /quote path by clicking on the twisty next to the path.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-testapidevportal.png)

1. Scroll down to the Try this operation section for the GET /quote path. Enter your Client ID and your Client secret and click the Call Operation button
2. Scroll down below the Call operation button. You should see a 200 OK and a response body as shown below.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-testapidevportalcall.png)

**Step 5 - Invoke the API**

Now that you have browsed the API Portal and registered / tested the API’s that **Quote** is providing, it’s time to test them out from a real application.

Sample code (snippets) are provided from developer portal for different language (cUrl, Ruby, Python, PHP, JavaScript, Java, Go, Swift).

1. Login into the developer portal as an application developer using your developer credentials.
2. Click the **API Products link**
3. Click the **Quote (v1.0.0)** API product link.
4. Click on **Quote** API in the left panel
5. Now, you can discover all operations with their properties and on the right hand side sample code.
6. On the right hand side you'll see the \***cuRL** expression
7. Copy it into your text editor window replacing **REPLACE\_WITH\_CLIENT\_ID** and **REPLACE\_WITH\_CLIENT\_SECRET** with your client id and your client secret saved from the prior step
8. Remove ***"filter"*** parameter in the url the result is like this :

curl --request GET \

--url 'https://api.eu.apiconnect.ibmcloud.com/BLUEMIXID-BLUEMIXSPACE/sb/loanmgt/resources/loans/v1/quote?loanAmount=10000&annualInterestRate=1.1&termInMonths=36' \

--header 'accept: application/json' \

--header 'content-type: application/json' \

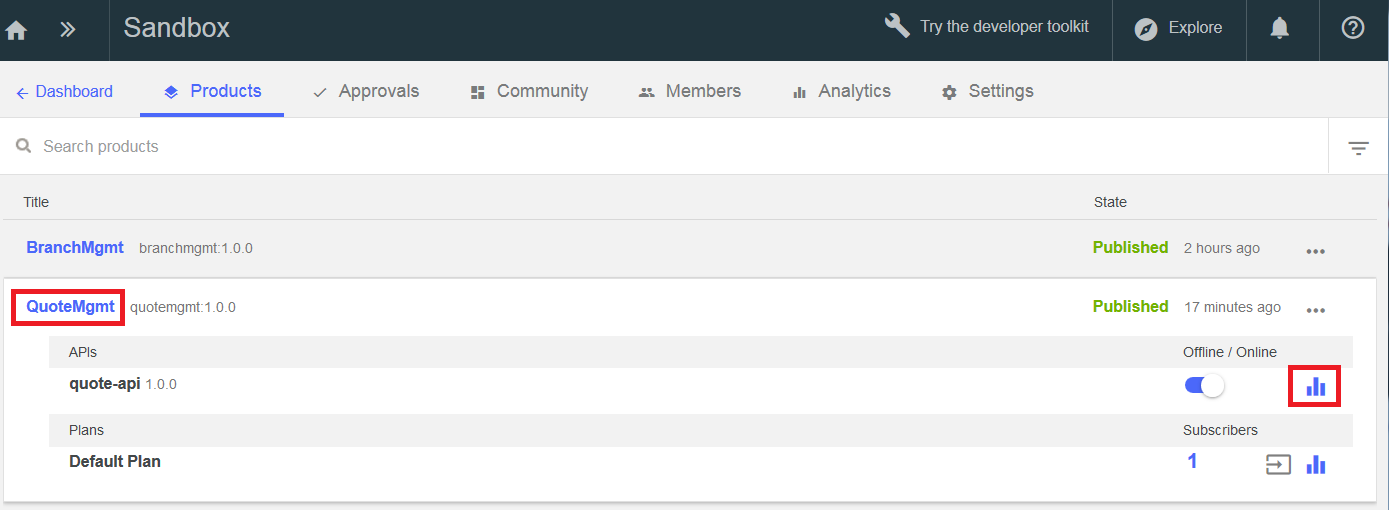
--header 'x-ibm-client-id: REPLACE\_WITH\_CLIENT\_ID'

1. Copy and try it into your terminal windows

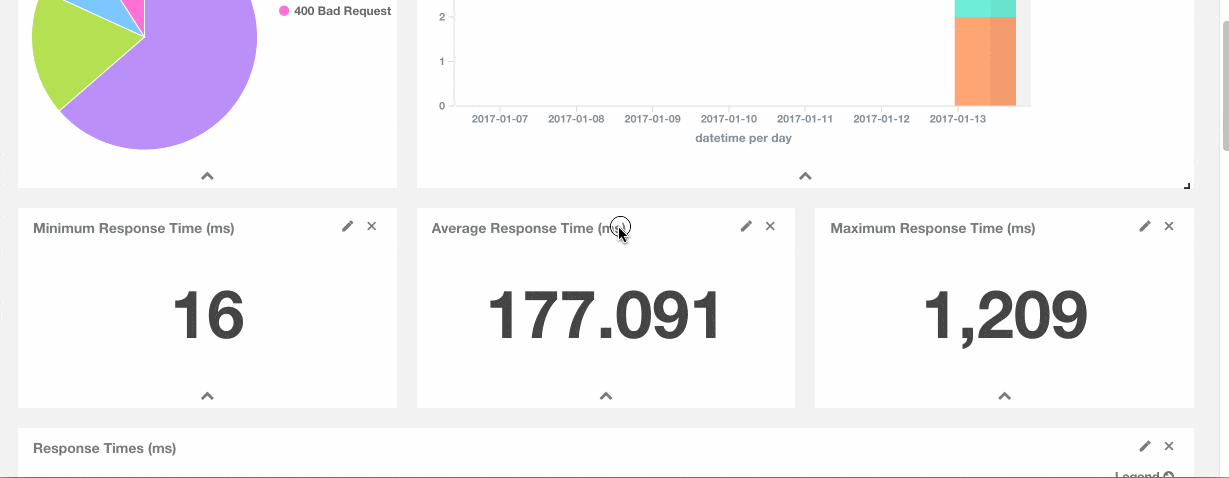
If all is OK, you should see the result of the quote in JSON format.

**Step 6 - Analytics**

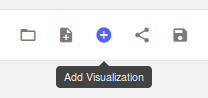
1. Return to the Bluemix API Manager screen.
2. Navigate to the Dashboard section and click on the **Sandbox** catalog tile.
3. Click on your **Quote** API and you can see number of subscriptions to you API.

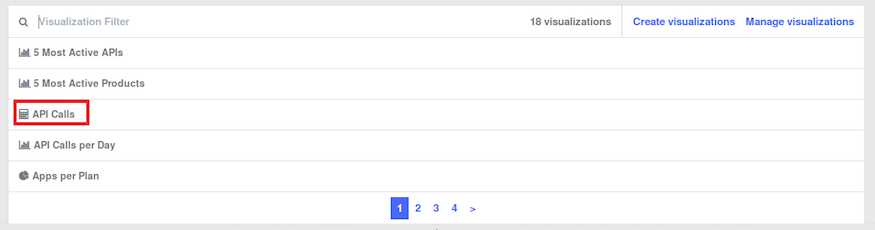
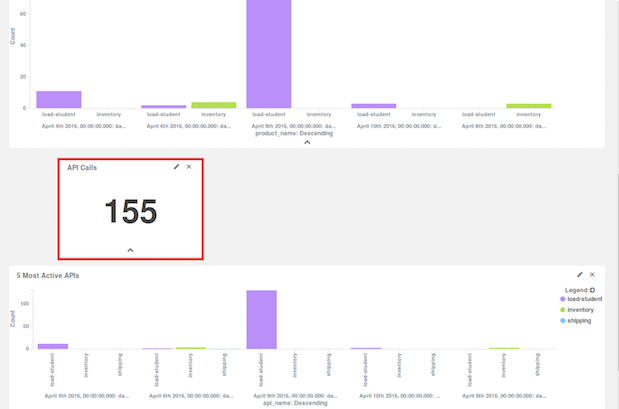
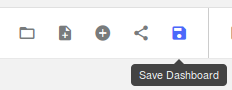
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-analytics.png)

1. To see analytic information, click on the analytic icon. [Anaytic icon app](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-analyticicon.png)
2. Now you can navigate to the Analytic dashboard to show analytic information for your API
3. You can show

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-analyticdashboard.gif)

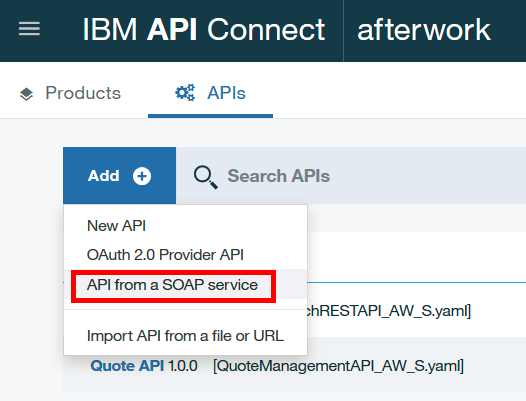
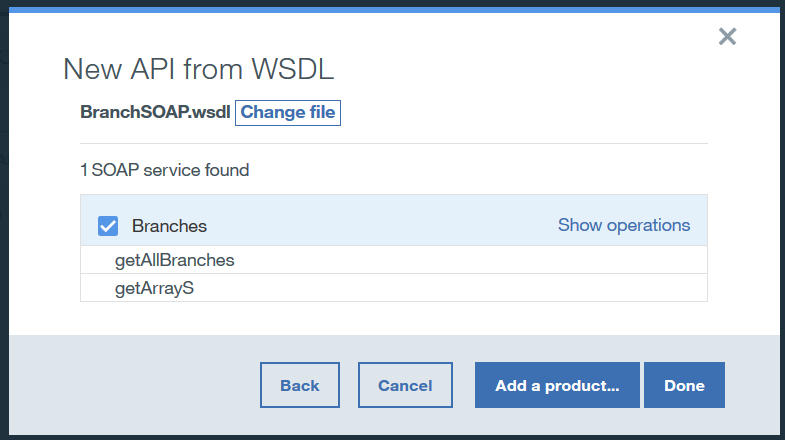
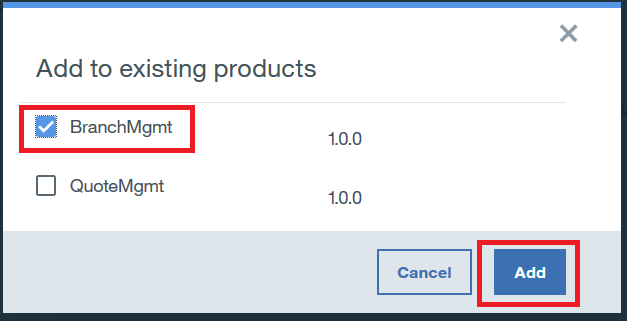
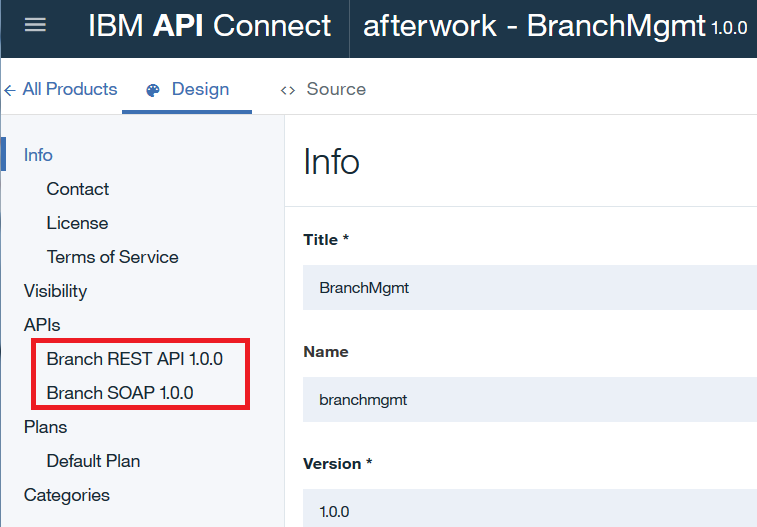
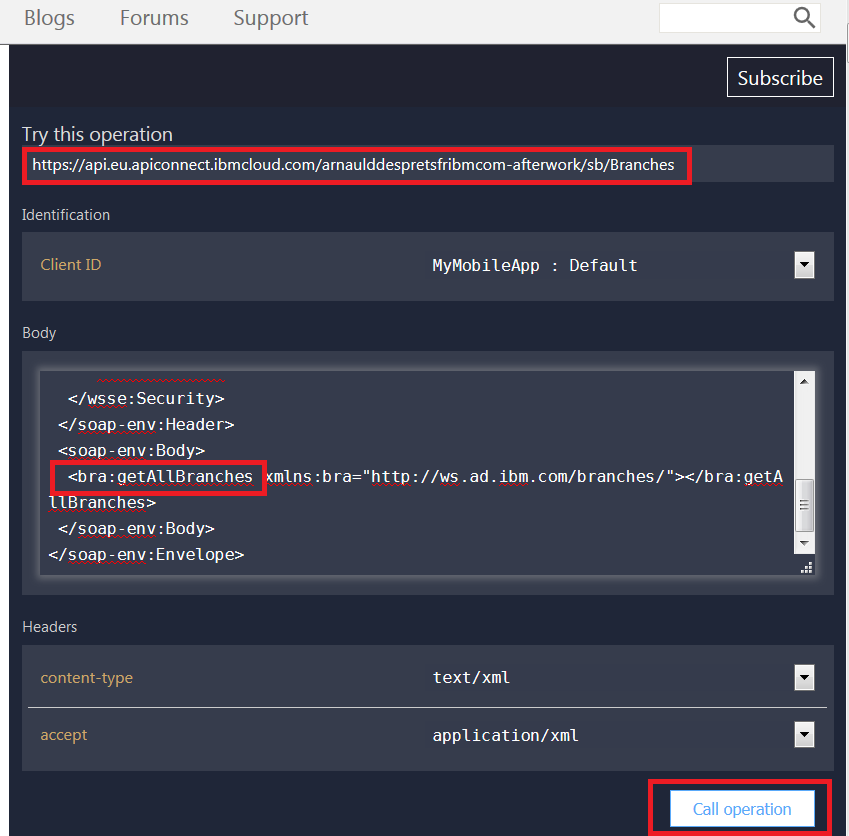
1. Click on the Sandbox catalog tile
2. From the Sandbox catalog configuration screen, click on the Analytics tab. [Anaytic screen](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/analytics-tab.png)
3. The default dashboard gives some general information like the 5 most active Products and 5 most active APIs. This information is interesting, but we can see much more information by customizing the dashboard. Add a new visualization by clicking on the + Add Visualization icon.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/analytics-add-visualization.png)

1. This will bring a list of some of the standard visualizations. You can then type in a string to filter through visualizations or use the arrows to page through the list.
2. Add the API Calls visualization to the dashboard by simply clicking on it. The new visualization will be added to the bottom of our dashboard. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/analytics-add-visualization-item.png)
3. Scroll down to find the new visualization. You can adjust the size by clicking and dragging the border from the lower right. Additionally, you can adjust its position by clicking and dragging the box to where you want it. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/analytics-add-visualization-new.png)
4. Feel free to play around with the other visualizations by adding them to the Dashboard. You can also save the dashboard by clicking on the Save Dashboard button: [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/analytics-save-dashboard.png)

**Step 7 - Create a SOAP API**

This is very similar to the creation of a REST API. The big difference is that we use a WSDL. The explanations will be short. We assume that we are still using the Manager (remote server in Bluemix), this would be the same user experience with the toolkit.

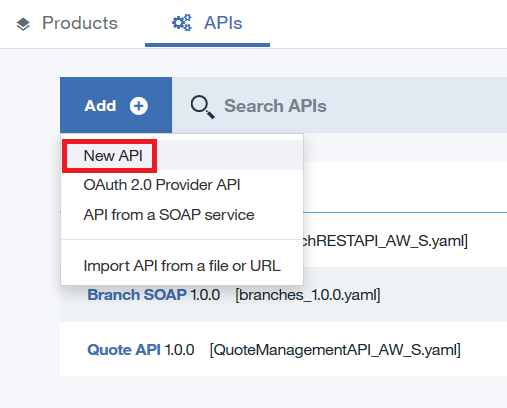
1. Download the WSDL for the Branch SOAP Service, you can find it [here](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/materials/step7/BranchSOAP.wsdl)
2. In the Draft area, in the APIs menu, click on Add and select API from a SOAP service [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-create.png)
3. Select the BranchSOAP.wsdl file to load [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-create-load.png)and click Done.
4. Rename the API to **Branch SOAP** by changing the title.
5. Add this new SOAP API to the existing BranchMgmt product [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-create-addtoproduct.png)Select The BranchMgmt product. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-addtoBranchProduct.png)
6. The BranchMgmt product now contains two APIs [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-product-with-two-APIs.png)
7. Publish the Product into the Sandbox catalog and test the getAllBranches operation in the Portal [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-test.png)

**Note:** We did not use a Properties and did not change the endpoint for the Proxy policy in the assembly panel, because the WSDL does have the correct endpoint on the Secure Gateway in Bluemix. In reality, you would probably want to create a properties that will point to the right endpoint depending on the environment.

**Step 8 - Create a SOAP to REST API**

In this step, we are going to create a New API (blank), specify the REST interface (paths, verbs and parameters) and add the Service definition using the Branch.wsdl file from the previous step, and then in the assembly will perform the mapping. You will see that API Connect supports array of objects or arrays of arrays very easily. Notice that SOAP to REST in this context means that we have a SOAP back end and we expose it in REST JSON, hence SOAP to REST. We could have had a another view and call it REST to SOAP :-)

Click on Add, and select New API

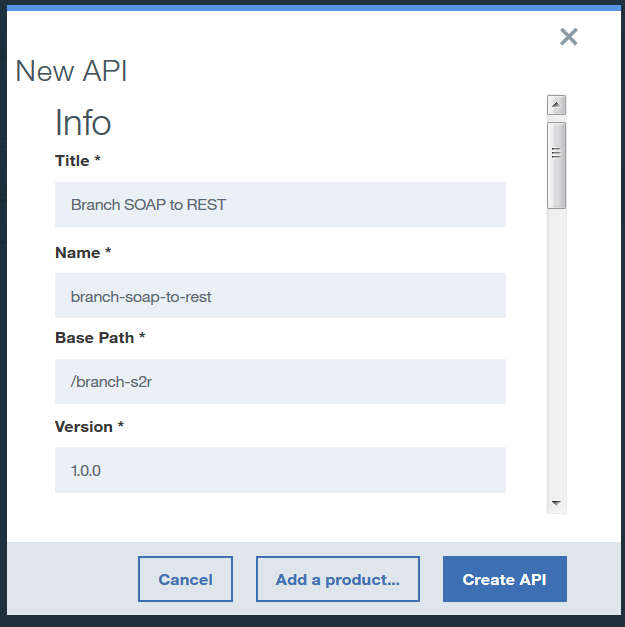
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-create.png)

Specify the following

Title: Branch SOAP to REST

Base path: /branch-s2r/v1

Then click on Create API button.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-new.png)

Add it to the existing BranchMgmt product.

Since we do not have a Swagger in this case, we need to define the various definitions for the interface manually and in particular for the response. The SOAP response is as follow:

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<ns2:getAllBranchesResponse xmlns="http://ws.ad.ibm.com/branches/getBranchById" xmlns:ns2="http://ws.ad.ibm.com/branches/">

<ns2:branch>

<ns2:id>123</ns2:id>

<ns2:type>atm</ns2:type>

<ns2:phones>

<ns2:type>personal</ns2:type>

<ns2:number>978-899-3445</ns2:number>

</ns2:phones>

<ns2:phones>

<ns2:type>professional</ns2:type>

<ns2:number>978-899-3446</ns2:number>

</ns2:phones>

<ns2:address>

<ns2:street1>600 Anton Blvd.</ns2:street1>

<ns2:street2>Appt 1</ns2:street2>

<ns2:zip-code>92626</ns2:zip-code>

<ns2:city>Costa Mesa</ns2:city>

<ns2:state>CA</ns2:state>

<ns2:country>USA</ns2:country>

</ns2:address>

<ns2:options>color screen</ns2:options>

<ns2:options>large screen</ns2:options>

<ns2:onlineStatus>true</ns2:onlineStatus>

</ns2:branch>

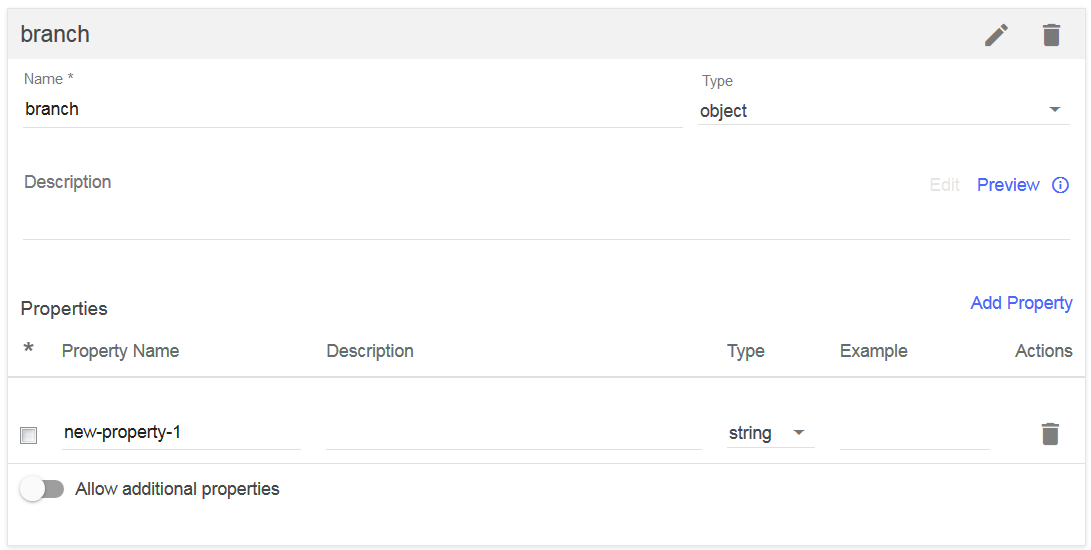
</ns2:getAllBranchesResponse>

</soap:Body>

</soap:Envelope>

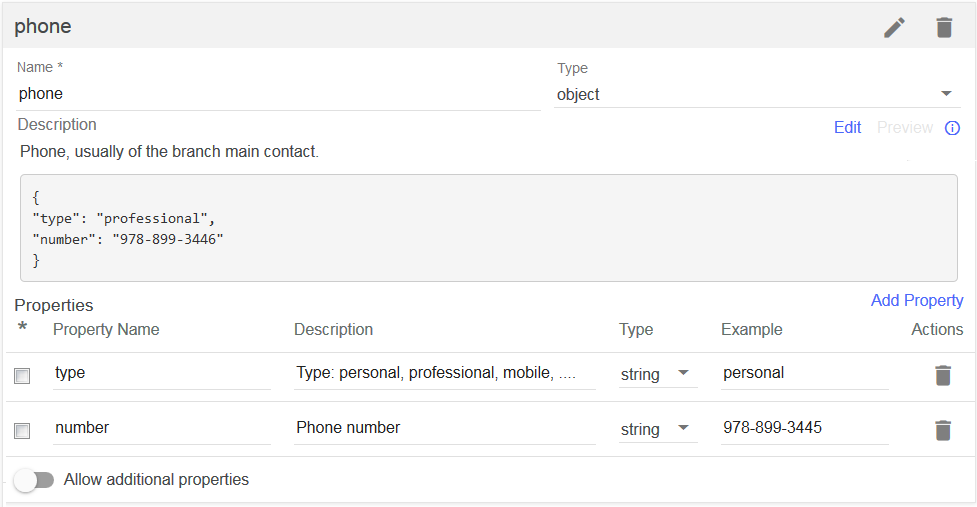
In this step, we only will implement the getAllBranches operation.

We start by creating the definitions. Looking a the SOAP message, we need to create: a branch, an address and a phone for the basic types and the following objects to implement the arrays: a phoneArray, an optionArray and finaly a branchArray. From the soap message above it is pretty obvious to determine the properties associated to those definitions. We use string for all properties except the onlineStatus property which is a boolean. Go to the definitions section, click on the + sign on the right. Rename the new definition to branch, keep the type to object

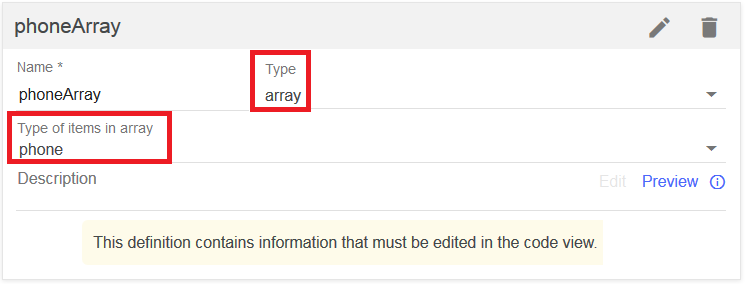
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/branchdef1.png)

Change the new property-1 to id, specify the description, keep the type to string and specify a sample value.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/branchdef2.png)

Repeat the same for the other properties. For example, here is the phone definition: [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/branchdef3.png)

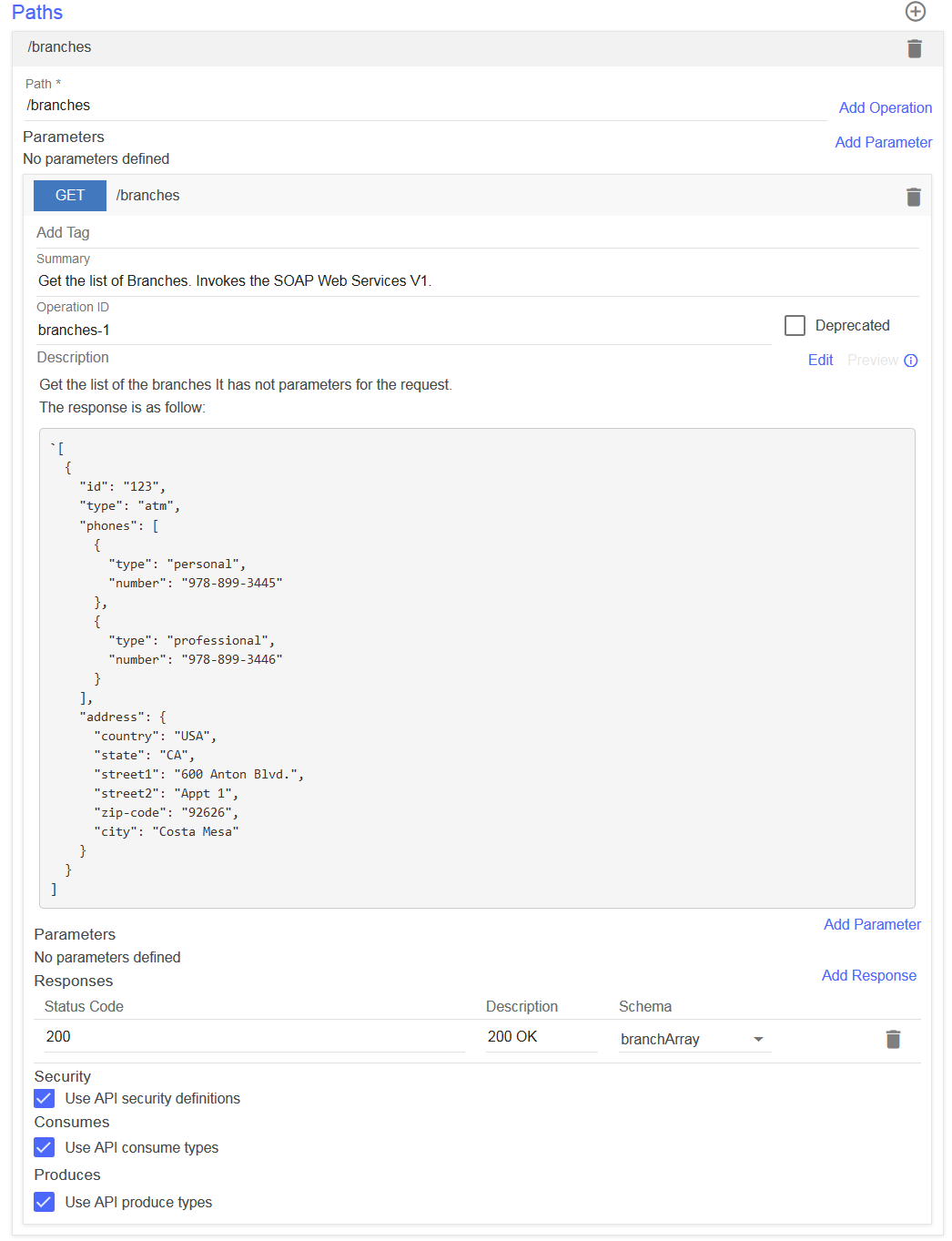
For the arrays, this is the same principle, except that we create an object of type array, and then select the type of items in the array. Below the phoneArray object

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/branchdef4.png)

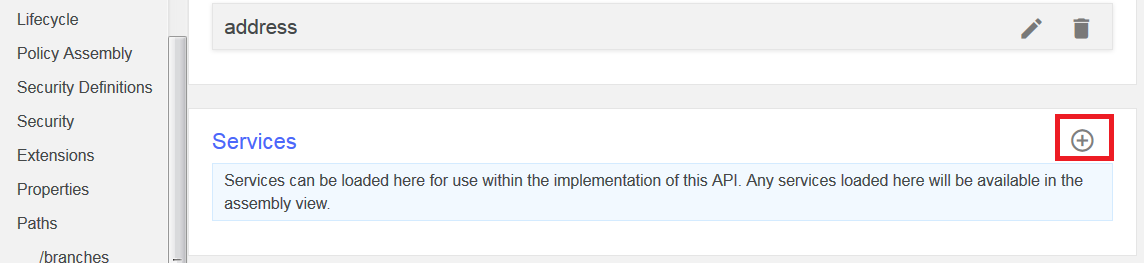
When you have created all the definitions, you should have something similar to this.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/branchdef5.png)

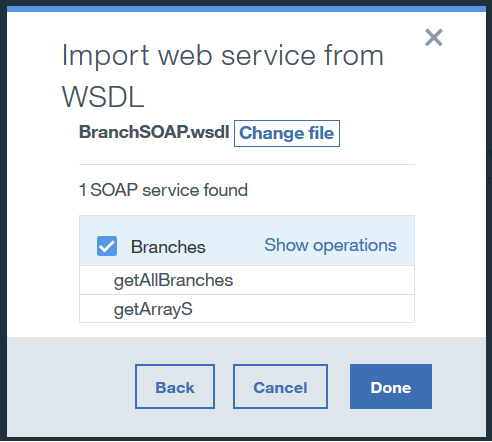
We can now create the path. In the Paths section, click on + sign on the right. Rename the path to /branches., expand the GET operation, provide an Operation ID, provide information on the path in the Description section. For the response, in the 200 status code section, specify the schema of the response, in our case, select branchArray.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/branchdef6.png)

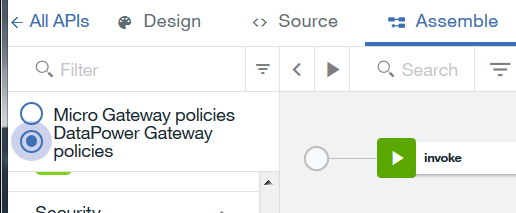
Add the Service definition and upload the BranchSOAP.wsdl file, by clicking on the + icon close to the Services definition

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-service-def.png)

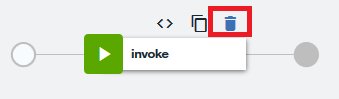
You can see the details of the Service, click on Done.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-service-def2.png)

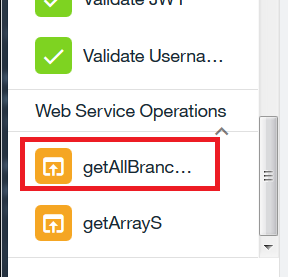
Now click on the Assemble menu. Make sure that you have the DataPower Gateway policies selected

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-assembly-wdp.png)

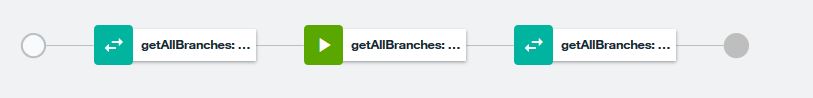
Remove the Invoke policy, by clicking on the Trash icon when you hoover your mouse on the Invoke policy.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-remove-policy.png)

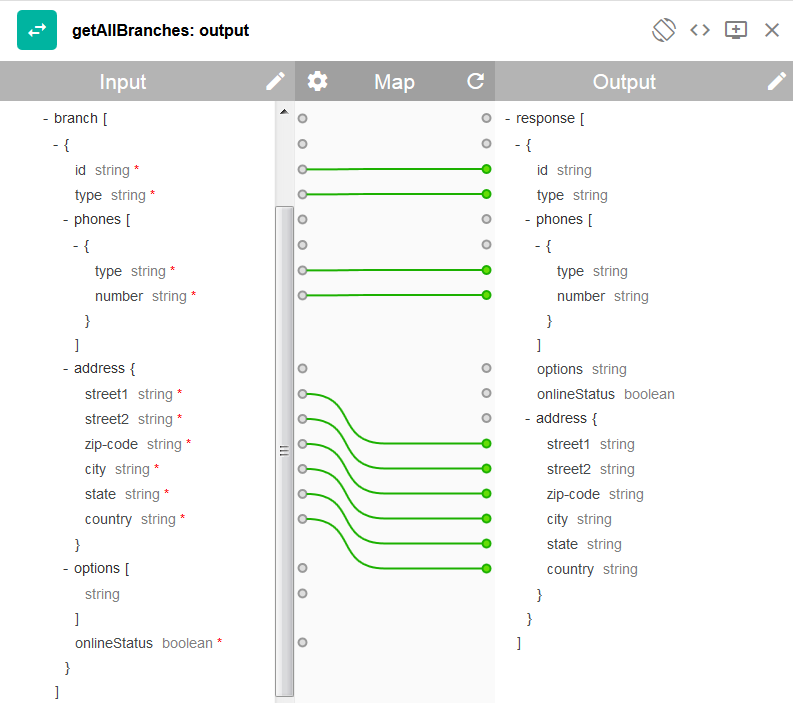
Drag and drop the getAllBranches Web Service Operations Policy from the palette.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-service-policy.png)

This automatically generates all you need to do the mapping.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-soap-rest-assembly-generated.png)

We just need to complete the mapping for both request and response. The first Map policy will convert the JSON request to a SOAP request, the second will convert the SOAP response into a JSON response. With the getAllBranches operation, there is a particularity. The request does not take any parameter. So there is not explicit mapping to perform. We are just going to open the first Map policy to understand what is happening. Click on the first Map policy, this opens the mapping editor. You can see on the right, the SOAP message, and nothing on the left. We do not need to specify an input in this case. The SOAP message contains everything needed, i.e. The SOAP headers, body, operation and setting the two specific HHTP Header SOAP action and the .

Mapping the response [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/restTOsoapMapOutput.png)

**Step 9 - Create a Cloudant service**

In order to store our data used by our API, we will need a persistent storage. To do so, we will use a Cloudant NoSQL database, a JSON document oriented store, compatible with CouchDB.

You can use a existing Cloudant service or create an instance of the service Cloudant DB.

1. Go to the Bluemix Catalog, create an instance of the service Cloudant NoSQL DB.
2. Search for **Cloudant** in the catalog
3. Select the free **Lite** plan
4. Give it a name such as **cloudant-db**.
5. Launch the Cloudand Dashboard. A new tab should open automatically with the list of databases.
6. Create a new database with the button on top right corner. Call this dabase : **test**. Make sure to use this name as this is expected by the persistence layer of API Connect.
7. Go back to Bluemix console and click the tab Service Credentials.

{

"credentials": {

"username": "XXXXXX",

"password": "XXXXXX",

"host": "f9246334-58d1-4a97-8bde-34c30121f063-bluemix.cloudant.com",

"port": 443,

"url": "https://USERNAME:PASSWORD@f9246334-58d1-4a97-8bde-34c30121f063-bluemix.cloudant.com"

}

}

1. Copy the url, username and password from the credentials into the your preferred editor. we will use these values later.

**Step 10 - Create a LoopBack application**

API Connect comes with a developer toolkit. This toolkit provides an offline graphical user interace named API Designer for creating APIs, the LoopBack framework for developing REST applications, a local unit test environment that includes a Micro Gateway for testing APIs, and a set of command line tools for augmenting the development toolset and assisting devops engineers with continuous integration and delivery.

1. Get help on the **apic** command set:

apic -h

The developer toolkit provides an integrated development environment for developing APIs and applications that use the LoopBack framework.

To create a new LoopBack project, use the command apic loopback; then use the apic edit command to edit the project in the API Designer.

**Note**: When working with the toolkit always be careful of where you are located on your file system. The working directory from where the apic command are started will be considered as the root of the loopback projects and products/APIs you are working at some point. Cautious must be taken on how you organise the directories. It also must take in considerations that at some point you will want to source control some of the generated files (such as the yaml files for example) in a Source Control Management system such as github.

1. Create an API Connect LoopBack application.

$ mkdir -p <your-favorite-working-dir>/apic/myfirstproject

$ cd <your-favorite-working-dir>/apic/myfirstproject

$ apic loopback

Next you will be asked to supply the name of the directory where the application will be created. Enter **Customer**

What's the name of your application? Quote

1. LoopBack will default the project directory name to the name of the application.
2. Press the ***Enter*** or ***Return*** key to accept the default value of inventory.
3. Next you will be asked to select the type of application. Use the arrow keys to select the **empty-server** option and press the ***Enter*** or ***Return*** key.

❯ empty-server (An empty LoopBack API, without any configured models or datasources)

1. At this point, the project builder will install the core dependencies for our Node.js application.

? Please review the license for API Connect available in /usr/local/lib/node\_modules/apiconnect/LICENSE.txt and select yes to accept. yese arrow keys)

? What's the name of your application? Customer

? Enter name of the directory to contain the project: Customer

? What kind of application do you have in mind? empty-server (An empty LoopBack API, without any configured models or datasources)

1. Change directory to your application directory

cd Customer

**Create a Datasource Connector to Cloudant**

The datasource is what allows the API to communicate with the backend data repository. In this case we will be using Cloudant to store the data item information.

There are two parts to this. First is the definition of how to connect to the backend system. The second is downloading the actual loopback connector for Cloudant.

In your terminal ensure that you are in the **Customer** directory.

cd Customer

In your terminal, type:

apic create --type datasource

The terminal will bring up the configuration wizard for our new datasource for the item database. The configuration wizard will prompt you with a series of questions. Some questions require text input, others offer a selectable menu of pre-defined choices.

Answer the questions with the following data:

**Note**: For **Connection String url** paste the previous value you copied about Cloudant credential in Step 1

| **Option name** | **Values** |
| --- | --- |
| ? Enter the data-source name : | **db** |
| ? Select the connector for db : | **IBM Cloudant DB** |
| ? Connection String url to override other settings | **YOUR Connection URL** [**https://username:password@host**](https://username:password@host) |
| ? database : | **test** |
| ? username : |  |
| ? password : |  |
| ? modelIndex : |  |
| ? Install loopback-connector-cloudant@^1.0.4 | **Y** |

Example :

? Enter the data-source name: db

? Select the connector for db: IBM Cloudant DB (supported by StrongLoop)

Connector-specific configuration:

? Connection String url to override other settings (eg: https://username:password@host): https:

//a836946d-92b5-41cc-b730-442b4235aae8-bluemix:7911bb5592e65f126903c59f6fa3d7f3b5bd4a1141951e31

938b6c6cb2efa852@a836946d-92b5-41cc-b730-442b4235aae8-bluemix.cloudant.com

? database: test

? username:

? password:

? modelIndex:

? Install loopback-connector-cloudant@^1.0.4 Yes

**Note**: By typing Y (Yes) to the question Install loopback-connector-cloudant, the Cloudant Connector will be downloaded and saved to your project automatically.

This will create a connection profile in the ~/Customer/server/datasources.json file. It is effectively the same as running the following to install the connector:

npm install loopback-connector-cloudant --save

For more information on the LoopBack Connector for Cloudant, see: <https://www.npmjs.com/package/loopback-connector-cloudant>

Note : You can create an api directly from a existing web service from the wsdl. Create a SOAP API definition from a WSDL definition file, or a .zip file that contains the WSDL definition files for a service with the following command: apic create --type api --wsdl filename

Note: You can create an API or Product from an OpenAPI (Swagger 2.0) template file by using the '--template template-name' option.

**Step 11 - Manage your API in API Designer**

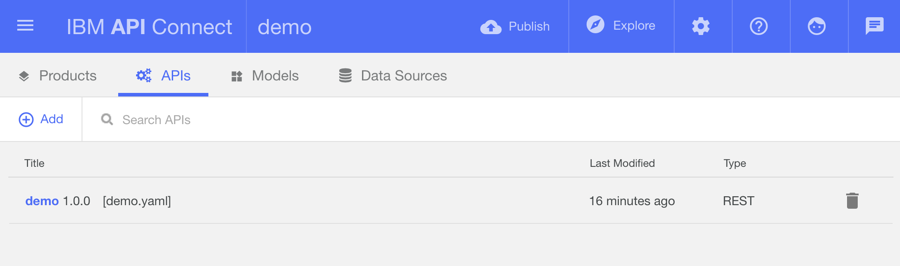
1. Launch API Connect Designer

apic edit

If the designer started correctly, a webpage will automatically opens and the terminal will show a message similar to this one:

Express server listening on http://127.0.0.1:9000

1. Click **Sign in with Bluemix**. If you're already sign in with Bluemix, you'll be automatically signed into the designer.
2. The designer opens into the APIs section showing the API definition we created from the command line.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-firstscreen.png)

###Create a Model for the **Customer** items

In this section, you will define the item data model for our **Customer** API and attach it to the Cloudant datasource. LoopBack is a data model driven framework. The properties of the data model will become the JSON elements of the API request and response payloads.

1. Click the **Models** tab.
2. Click the + Add button.
3. In the New LoopBack Model dialog, enter **Customer** as the model name, then click the New button.
4. When the Model edit page for the item model displays, select the **db** Data Source:

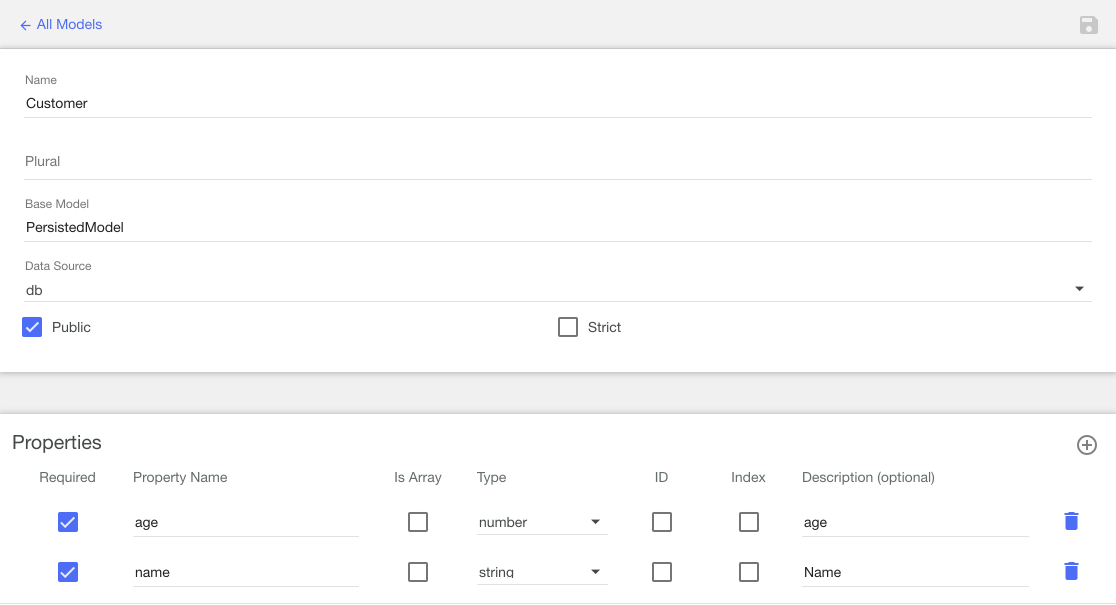
####Create Properties for the Customer Model

The Customer table in the database has 6 columns that will need to mapped as well. To start creating properties for the item model:

1. Click the + button in the Properties section.
2. The Customer data model consists of six properties. Use the data below to add each of the properties:

| **Required** | **Property Name** | **Is Array** | **Type** | **ID** | **Index** | **Description** |
| --- | --- | --- | --- | --- | --- | --- |
| yes | name | no | String | no | no | Name |
| yes | age | no | number | no | no | Age |

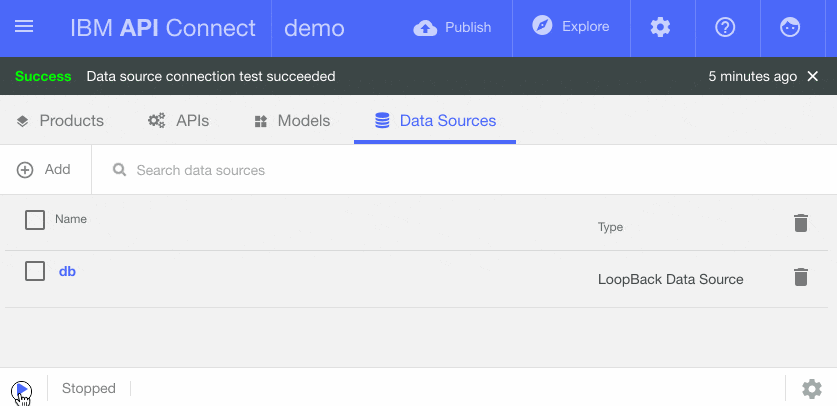
1. Scroll to the top of the page and click the **Save button** to save the data model.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/allmodel.png)

1. Click the All Models link to return to the main API Designer page.

**Step 12 - Test your API**

1. Let's test the API in the Designer. First, start the server by clicking the play button in bottom left corner. Once the server is started, you should see the endpoint of the Local Micro Gateway.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-server.gif)

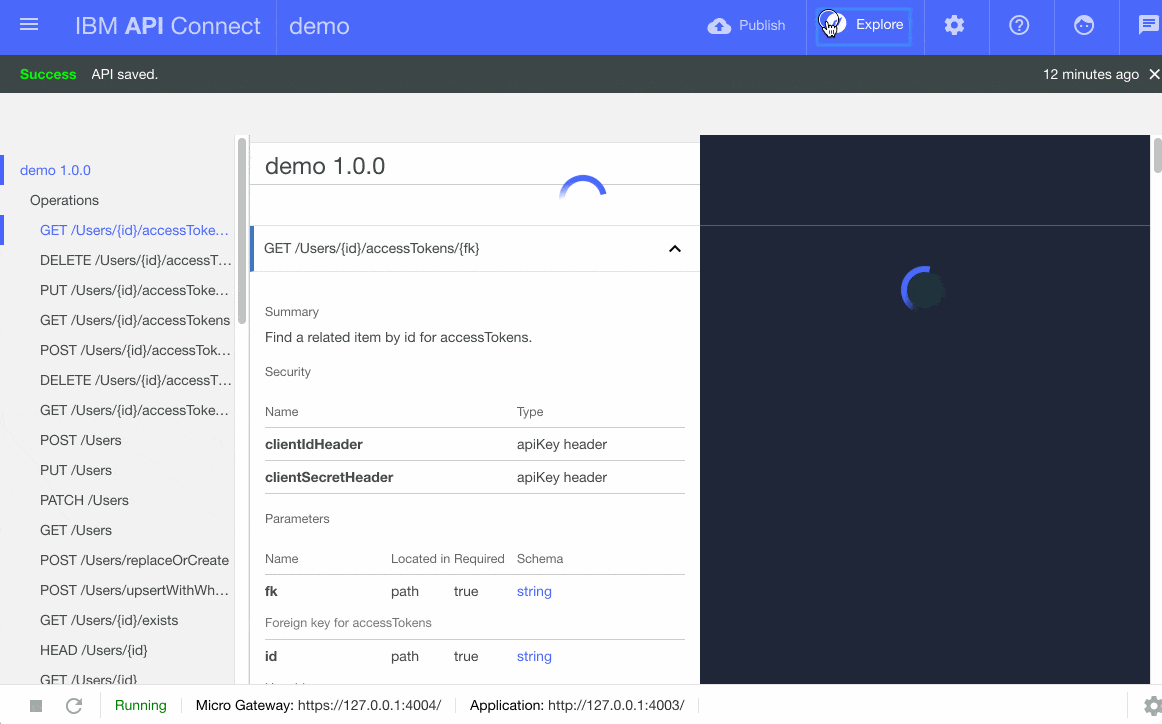
1. On the server is started, Click on **Explore** in the top right corner.
2. Select the operation POST /Customers. Click on *Generate* hyperlink before the button **Call operation** in the right panel.

{

"name": "YOUR NAME",

"id": -30239449.275000483

}

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-test.gif)

1. The first time you will most likely get a CORS error as follows:

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://$(catalog.host)/api/Customers

1. Open the url below in a new tab of your browser:

<https://localhost:4002/api/Customers>

1. Click on Advanced. Accept the exception.
2. Go back to the Explore view in API Designer and click **Call operation** again. You should get a successful response code 200 OK.
3. If you have kept the Cloudant DB dahsboard open, you can select the database **test** and view the newly created record.
4. Congratulations you successfullly tested your API.

Congratulations. You have completed this Lab!

**Additional Resources**

For additional resources pay close attention to the following:

* [API Connect v5 Getting Started: Toolkit Command Line Interface](https://github.com/ibm-apiconnect/cli)
* [API Connect v5 Getting Started: API Products](https://github.com/ibm-apiconnect/product)
* [API Connect Developer Center](https://developer.ibm.com/apiconnect)
* [API Connect v5 Knowledge Center](http://www.ibm.com/support/knowledgecenter/SSMNED_5.0.0/mapfiles/ic_home.html)
* [Follow us @ibmapiconnect](https://twitter.com/ibmapiconnect)
* [POT API Connect](https://ibm-apiconnect.github.io/pot/)
* [POT API Connect Customization](https://ibm-apiconnect.github.io/faststart/)
* [PSA Sample of customized portal](https://developer.psa-peugeot-citroen.com/)