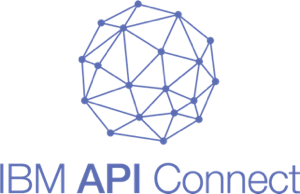
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic_banner-mini.png)

**Introduction**

This article is the updated version for V2018. The original version was addressing V5 and can be found [here](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/ReadMe.V5.md). This article is made of two parts. In the first section, we explain what is the purpose of API connect and the concepts behind IBM API Connect. Then in the second section, we will practice labs in order to get hands-on with IBM API Connect. 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 is a Kubernetes installation with IBM API Connect 2018.4.1.10. It will be updated as much as possible to follow the new versions of API Connect. The version 2018.x is out since 30th March 2018 and the LTS has been released on 15th November 2018. Due to the significant changes brought by APIC V2018 and also because IBM Cloud (former Bluemix) is using API Connect v5 as of today (19th March 2020), this lab will use the on-premise version and an updated version of the lab will be made when the IBM Cloud infrastructure is updated. The SaaS version of API connect should be released very soon.

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 v2018 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 (Lab)

**Pre-Requisites**

It is possible in IBM API Connect to either develop locally on a developer machine or develop directly in the Manager. The user experience is very similar in both cases. Developing locally provide the benefits of being able to directly use a source control management system such as Git.

* Download the Designer
* Have an API Connect instance available (in our case the ICP4I instance)
* In order to perform some basic local testing we will also use the LTE - Local Test Environment which allows to perform simple testing locally. The LTE is as of today in beta.

There is no need to install the designer since this is a simple executable. For the Local Test environment, I'm using a Ubuntu desktop because it is simpler to use Docker on it rather than on Windows where there are some incompatibilities between Docker and VMWare. The installation of the LTE is explained here.

**Hint:** to find the various executable for a specific version, you can find all the link in the article called IBM API Connect is available. for example, you will find an articled called *IBM API Connect V2018.4.1.8-iFix1.0 is available* [here](https://www.ibm.com/support/pages/ibm-api-connect-v2018418-ifix10-available). In this article, you will find the content of the fix, the link to the various images usually in Fix Central and optionally the link to the LTE.

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

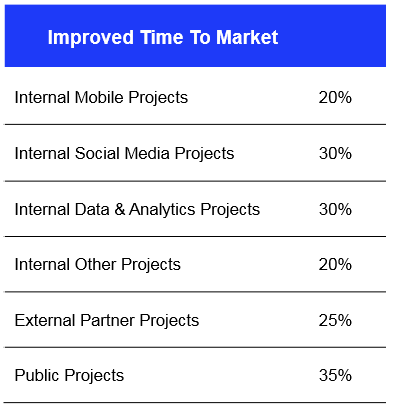
This chapter does not intend to describe all the possible use cases of API Connect, but instead provide some simple and concrete common usages of API Connect.

**Main use cases**

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

**Where APIs are used in companies?**

Here is the result of Forrester study, performed in February 2017.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/ForresterStudy.png)  
*Fig. 1: Forrester use cases study*

As you can see a lot of projects are using API internally and the very visible part, the public projects represents 35%.

Read the full study: The Total Economic Impact™ of an API Management Solution <http://ibm.biz/APICTEIstudy>

**Architecture and terminology**

**API Connect architecture on-premise**

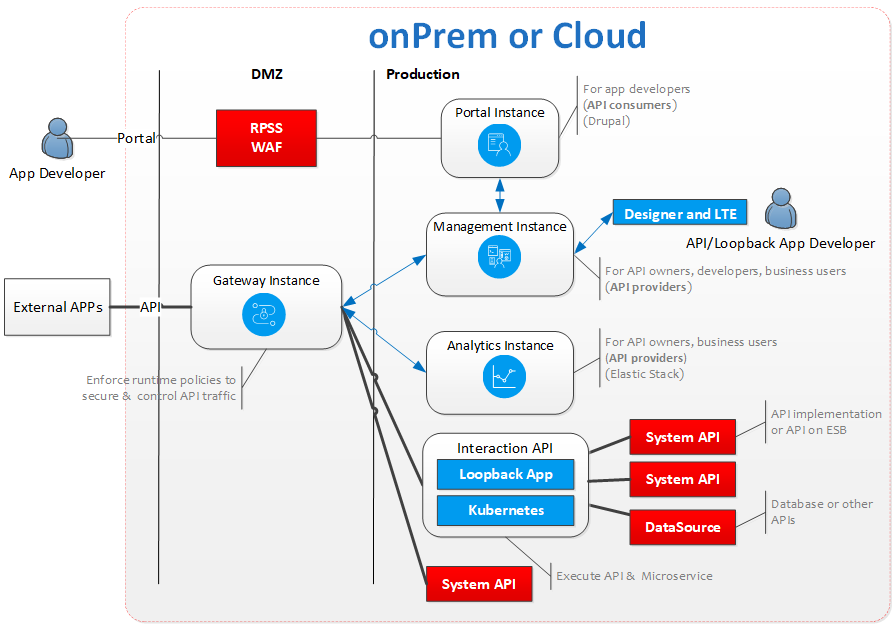
**Components in IBM API Connect 2018**

The main components composing API Connect are:

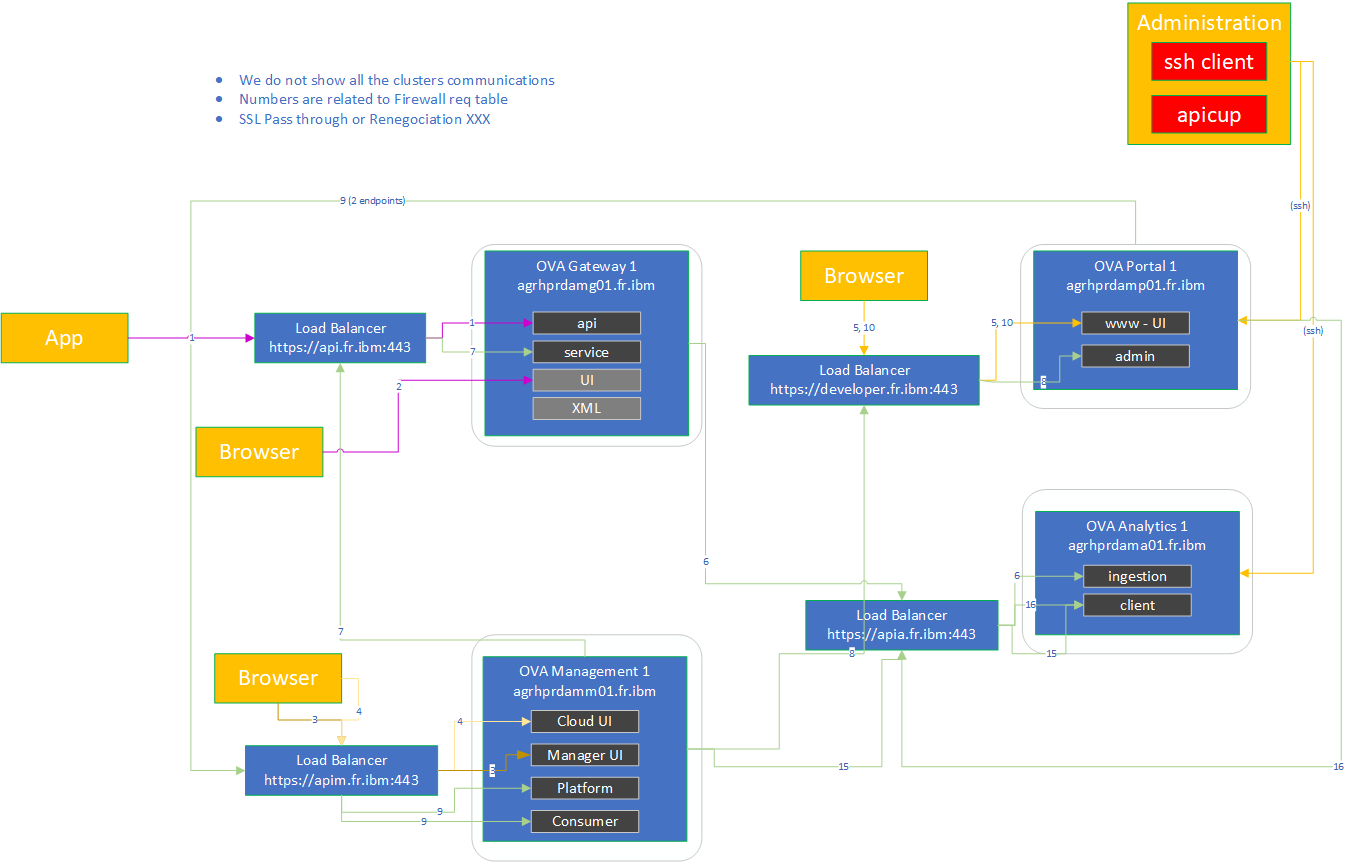
* The **Gateway** (IBM DataPower Gateway). The requests from apps are going through the gateway, policies are enforced and API events for the analytics are generated.
* The **Analytics** is a full Elastic Stack deployment. This is a new feature of APIC v2018. The analytics are non longer part of the manager, which allows for true multi cloud architecture, where the Gateways can be deployed in another separated environment along the analytics which require some kind of colocalization for performance reasons. Notice that the Elastic Stack is partially optional, in the case, where you do already have your own instance. In the case, where you really do not want to use the internal analytics then it is possible to only install the ingestion part.
* 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. In v2018, Drupal was updated to version 8.
* 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, especially if you do not have some ESB capabilities.
* Associated to the Loopback runtime is the **Kubernetes** that monitors the Loopback runtime and can provide advanced feature such as auto-scaling.
* The **Designer**, it offers the same web experience as the manager to manage APIs and allow development on the developer's machine.
* The apic toolkit, really the CLI for APIC. APIC is developed in a simple manner and accessible through REST/JSON API. So the Web UI, the apic CLI are just based on those REST API. This is also a new benefit of the V2018, where we have now a complete set of supported REST API, in order to configure initially the product (APIC Cloud), the Manager and the portal.

From an architecture point of view and it is important to consider that for HA the notion of quorum arise and I would advise to have a good understanding of what are the implications. Finally, APIC V2018 is a complete rewrite and redesigned version based on the use of Docker and Kubernetes. If you do not have a Kubernetes platform available, it is possible to use OVA that are encapsulating the Kubernetes environments. apic CLI encapsulate the kubectl/docker hiding the complexity of this platform. I would argue that even with OVA, it will be an advantage to understand Kubernetes and Docker principle.

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/v2018.apic-archi-on-prem.png)  
*Fig. 2: V2018 Architecture*

In more details some of the communications between each component in an OVA deployment non-HA. For more information, see the Required Ports between zones [here](https://www.ibm.com/support/knowledgecenter/SSMNED_2018/com.ibm.apic.install.doc/overview_apimgmt_portreqs_vmware.html) for OVA deployments or [here](https://www.ibm.com/support/knowledgecenter/SSMNED_2018/com.ibm.apic.install.doc/overview_apimgmt_portreqs.html) for Kubernetes deployments.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/V2018OVADeploymentCommunications.png)  
*Fig. 3: Communications between components*

**A word on quorum**

Nowadays a lot of systems containing data are distributed. This increases availability but at the same time data consistency between the various instances is highly required. There are several strategies to support this requirement, active-passive, active-active where it becomes a little bit more difficult. One approach to solve this is to use the notion of quorum, where using a simple mathematical decision (N-1)/2 a decision can be taken to identify whether the system should be shut down in order to avoid data corruption or to keep the system available but alert that data corruption has occurred and some reconciliation work may have to happen. Many components in API Connect or related to API Connect are based on distributed databases. Kubernetes etcd, Elastic Stack, Cassandra, Redis, etc ... When you design your topology, I would really advise that you understand what you want, what can be done and what may happen if losing quorum, how the individual component will behave. You should also perform some disaster testing according what you try to achieve. There are a lot of literature on this topic available. One final word, for fun, remember that ∀ n ∈ ℕ (n-1)/2 < n/2 , that means that if you lose half of your instances you are in trouble and you need to start worrying about what is happening!

**Terminology**

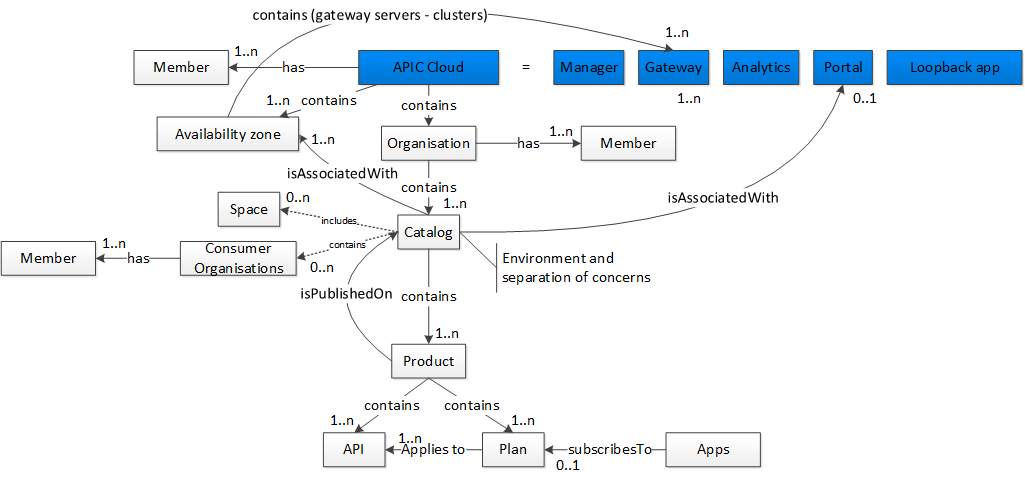
* An **Organization**: It is the highest logical level of separation; it may provide multitenancy if required. An organization has its own catalog, members, resources, etc ... It is managed from the cloud console (not the manager).
* An **API**: Can be SOAP or REpresentational State Transfer - REST API 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 organization and needs to separate the infrastructures.
* **Assembly panel**: this is where we specify the policies to be executed in the gateway for each transactions.

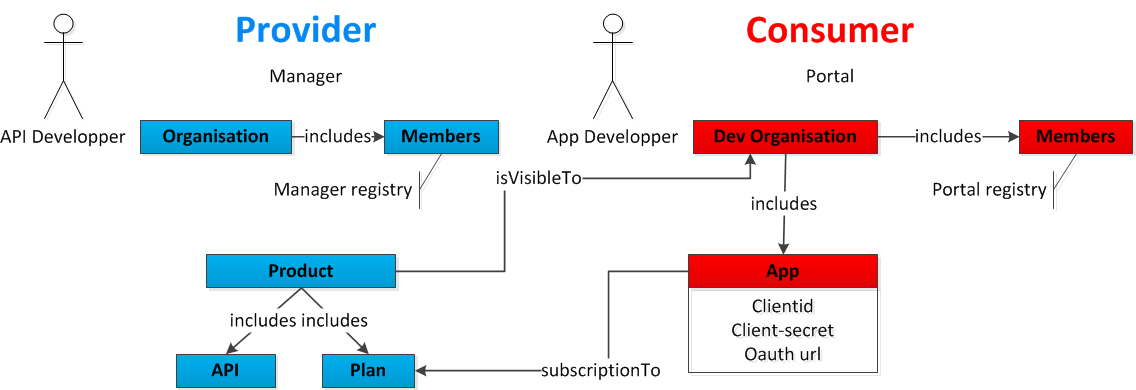
**Concepts maps**

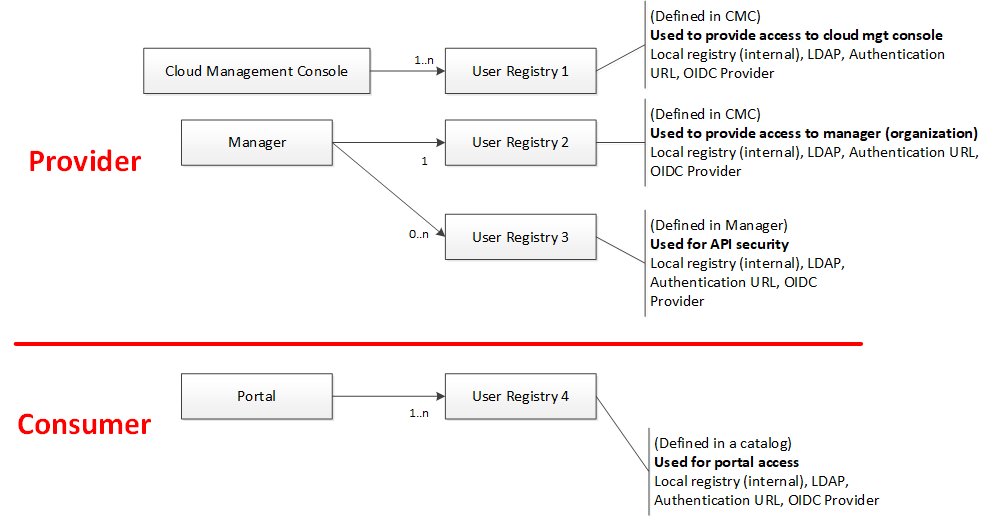
Below the concepts related to the overall product. There are many ways to choose the right deployment based on requirements.

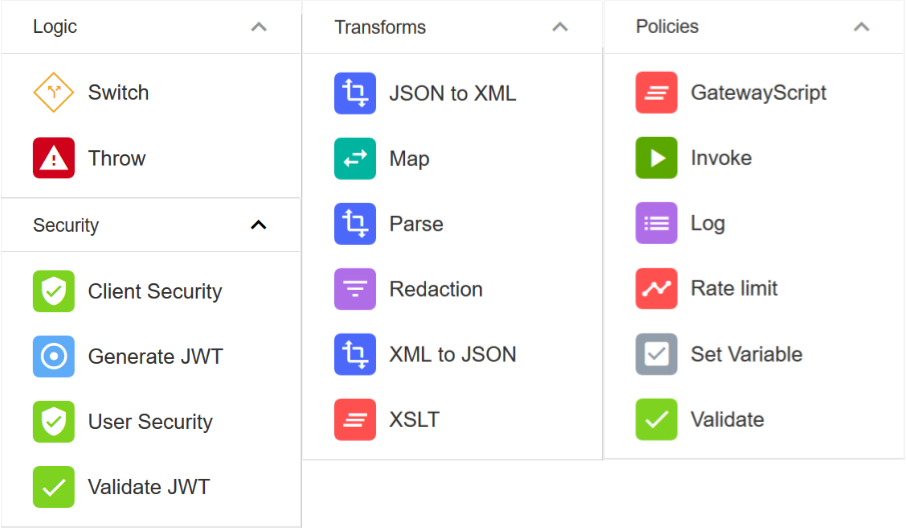
The questions to ask:

1. How many APIC clouds? (usually between 1 and 3, driver: production separated or not from non-production and other environments)
2. How many clusters of gateways? (separate instances, driver: usually security zones (DMZ or not and other zones))
3. How many gateway services? (drivers: the number of clusters of gateways above and isolation)
4. How many instances for each component? (how many instances of managers (usually 2 per APIC Cloud), instances of gateways (usually at least 2 depending on the traffic and high availability desired), instances of portals (usually 3 per APIC Cloud))

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/v2018.apic-ConceptsMap.png)

Below the concepts related to the subscription of an App to a Plan within a product referring to all APIs or a set of APIs within that product.   
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/subscription.png)

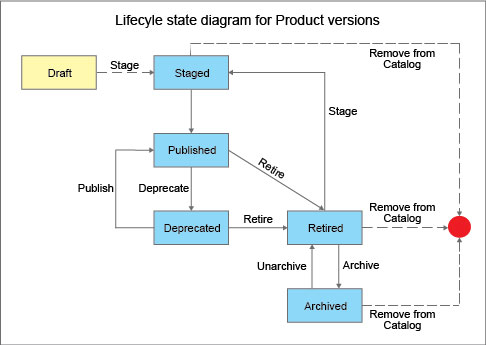
Below the concepts related to the organisation of the user registries.   
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/apic-registries2018.png)

Below the available policies in API Connect 2018.4.1.8fp1   
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/v2018.policies.png)

| **Category** | **Name** | **Description** | **-** | **Category** | **Name** | **Description** |
| --- | --- | --- | --- | --- | --- | --- |
| Logic | Switch | switch | - | Other | Invoke | invoke |
| Logic | Throw | exception handling | - | Other | Log | log |
| Transform | JSON to XML | transform | - | Other | Rate limit | quota mgmt |
| Transform | Map | mapping | - | Other | Set Variable | set |
| Transform | Parse | deserialise | - | Other | Validate | Schema validates |
| Transform | XML to JSON | transform | - | Transform | Security | Authenticate |
| Transform | Redaction | obfuscate | - | Other | Security | Generate JWT |
| Transform | XSLT | any code | - | Transform | Security | User Security |
| Other | GatewayScript | any code | - | Transform | Security | Validate JWT |

**Lifecycle of a Product**

The lifecycle of a product is as follow:

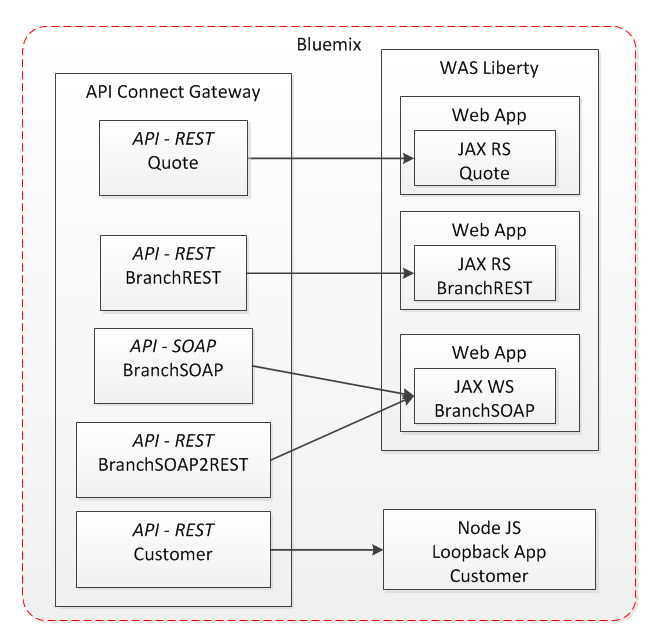
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/diagram_product_lifecycle.jpg)

It is possible to use an approbation control between each transition

**Overall design of the lab**

The goal of this lab is to build 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, JAX WS and NodeJS (with Loopback Application) for pure cloud APIs. The use of Secure Gateway in relation with a local deployment would demonstrate hybrid APIs.

**Steps for the lab**

1. [Check the development environment](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-1---check-the-development-environment)
2. [Expose an existing REST API](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-2---expose-an-existing-rest-api)
3. [Publish your API to the Sandbox catalog](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-3---publish-your-api-to-the-sandbox-catalog)
4. [Consumer Experience](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-4---consumer-experience)
5. [Invoke the API](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-5---invoke-the-api)
6. [Analytics](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-6---analytics)
7. [Create a SOAP API](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-7---create-a-soap-api)
8. [Create a SOAP to REST API](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-8---create-a-soap-to-rest-api)
9. [Create a Cloudant service](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-9---create-a-cloudant-service)
10. [Create a LoopBack application](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-10---create-a-loopback-application)
11. [Manage your API in API Designer](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-11---manage-your-api-in-api-designer)
12. [Using OAuth to protect your API](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-12---using-OAuth-to-protect-your-api)
13. [Testing the quality of your API](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/README.2018.md#step-13---testing-the-quality-of-your-api)

**Step 1 - Check the development environment**

For this lab, we are going to use the Designer instead of using the manager. We also use the Local Test Environment to perform basic testing. We will then deploy the API into the sandbox catalog from the designer.

**Note**: Using the toolkit/designer (locally) or using manager directly (remote server) is a pretty important decision. Using the toolkit has the benefit 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 usually a lower response time. Using the Manager simplifies sharing the API Drafts. In reality, there are ways to benefit of both approaches, especially considering a devOps approach. I'm using a Linux environment, you may have to adjust the commands with your specific environment being Windows or MacOS.

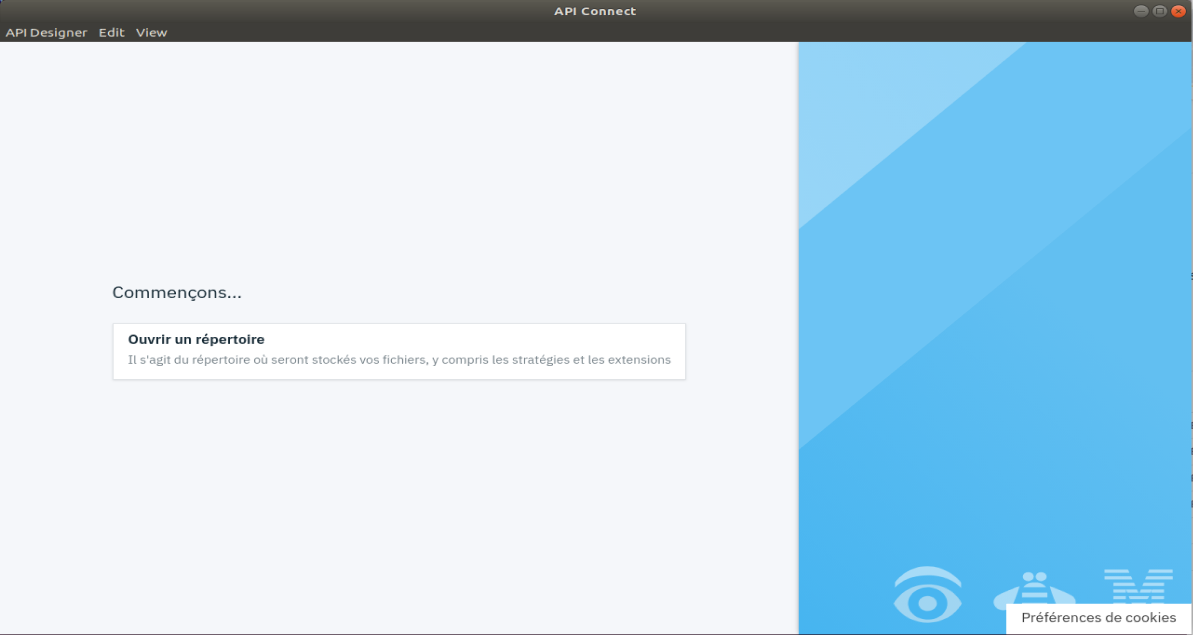
Let's check that development environment is ok. We first prepare the docker environment to start the local test environment. sudo docker load < apic-lte-images-2018.4.1.8-ifix2.0.tar.gz. The information on the local test environment can be found under the title *Testing an API with the Local Test Environment* in the IBM Knowledge Center. To start the LTE, in my case, cd ~/apic-lte/linux, then sudo ./apic-lte start.

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

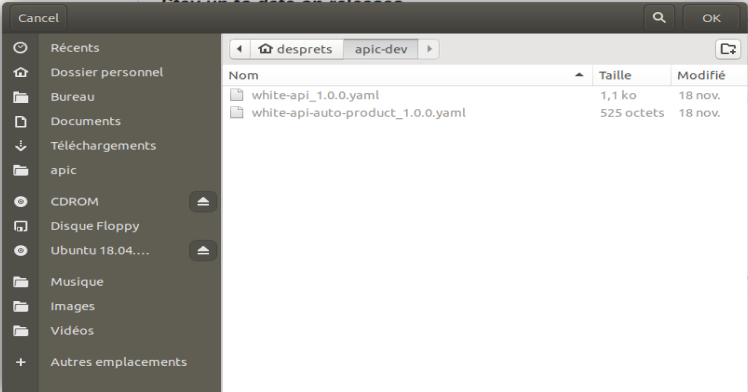
Take a note of the Platform API URL and the user/pwd to be used. we will need them when we start the Designer.

**Hint:** To check that the LTE is correctly started: apic login --server localhost:2000 --username shavon --password 7iron-hide --realm provider/default-idp-2

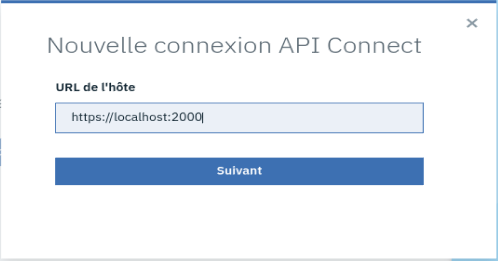
You should get the following message: Logged into localhost:2000 successfully

To start the designer, just execute the ./api\_designer-linux. I assume here you downloaded the version of the Designer from the IBM support site, the version must be similar to the API connect installation. It does start a window, the first information to specify is the working directory, where the artefacts (yaml) will be created. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-specify-work-dir.png).

Click on Open a folder and specify the directory where you work.

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

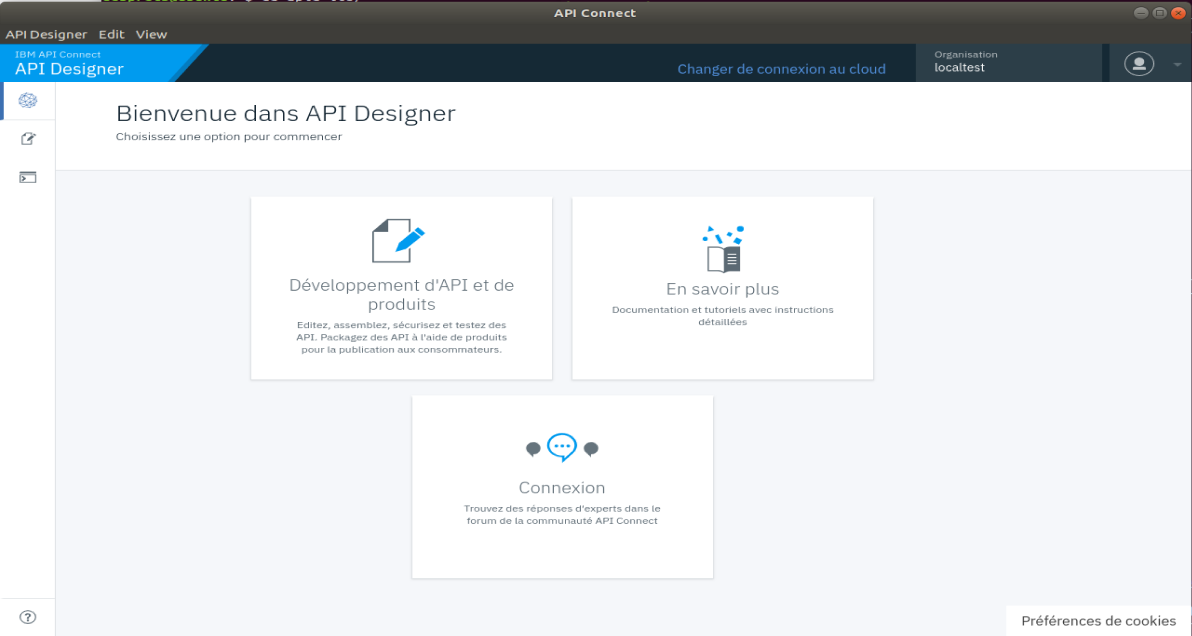
The specify the manager you want to work with, in our case, we will have two locations, the local test environment and the manager of a remote instance. First let's use the LTE. We enter the information for the local test environment, in our case <https://localhost:2000>.

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

The login screen appears, we enter the credentials that were indicated when starting the LTE, in our case shavon/7iron-hide.

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

The first screen appears, we are ready to create our first API.

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

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

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 ~/apic-dev assuming that you have created such directory.

1. Ensure that you have downloaded the yaml correctly cd ~/Downloads and the head QuoteManagementAPI\_AW\_S.yaml you should obtain the following:

swagger: '2.0'

info:

description: Quote API

version: 1.0.0

title: Quote API

contact:

name: Arnauld Desprets

url: 'http://thequoteapi/terms.html'

email: arnauld\_desprets@fr.ibm.com

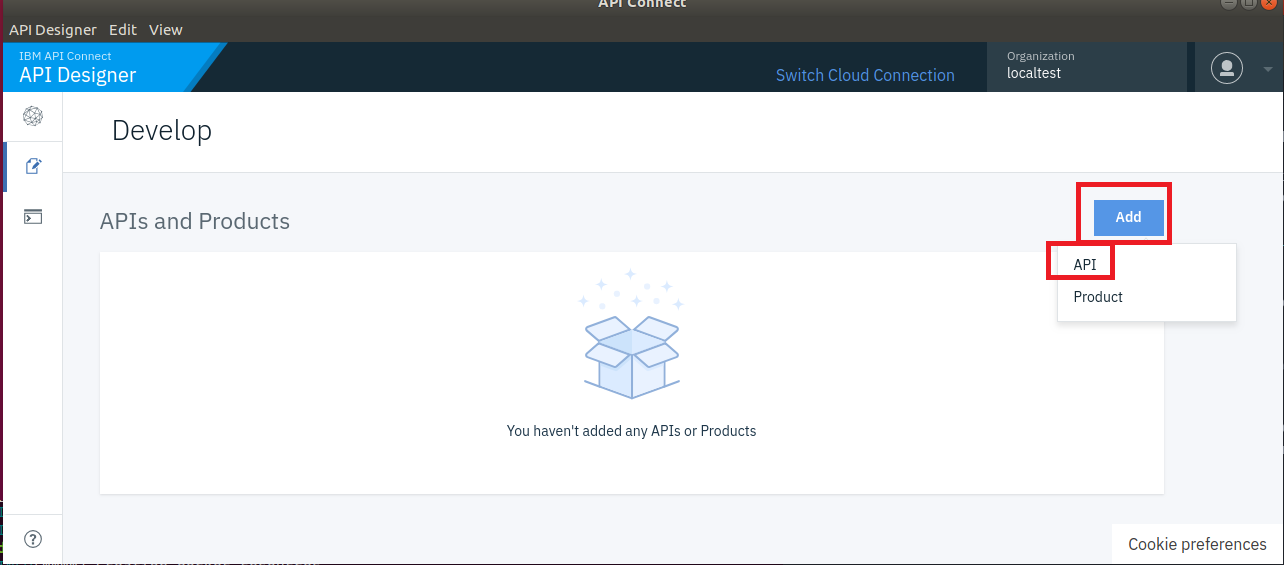
x-ibm-name: quote-api

If this is not the case, then download it again or copy it in a file.

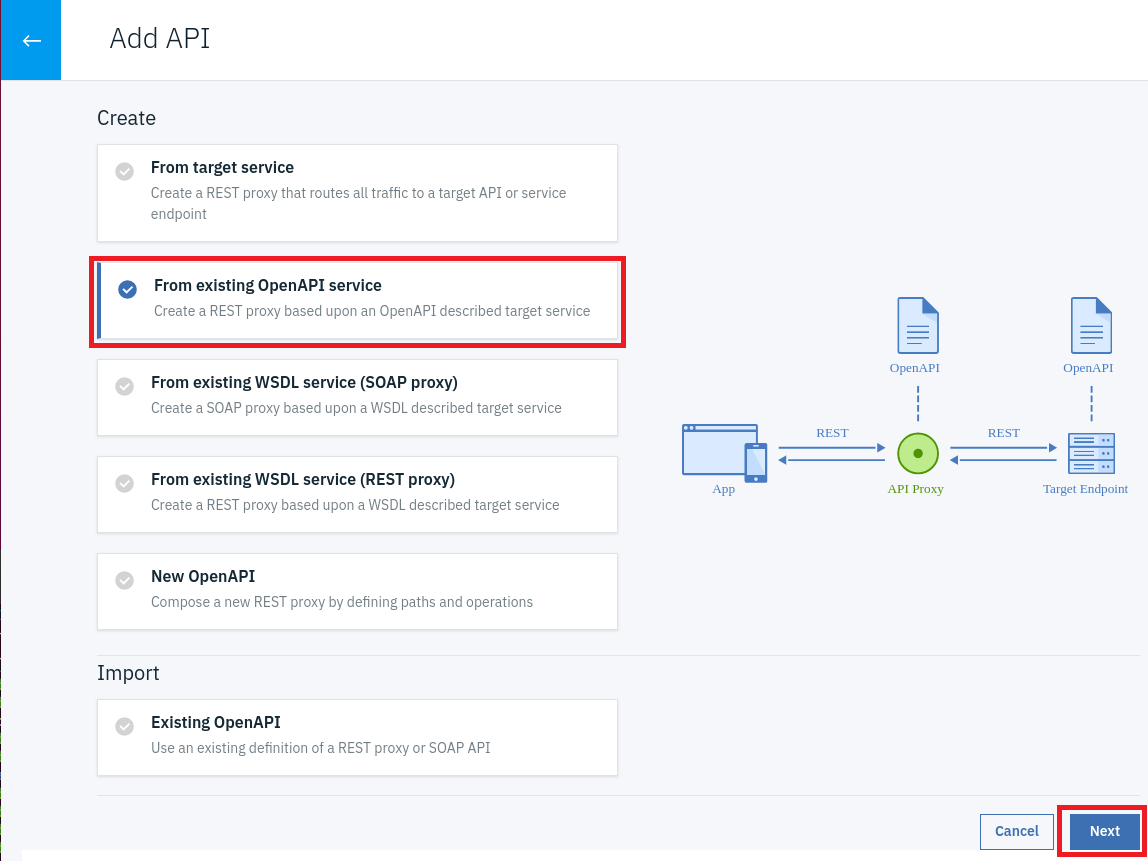
1. It is a good idea to check that the back-end API is running before exposing it. In this case, we are going to use the GET verb on the quote operation. Type the following: curl -H "Accept: application/json" "http://SampleJAXRS20-aw.eu-gb.mybluemix.net/loanmgt/resources/loans/v1/quote?loanAmount=10000&annualInterestRate=1.1&termInMonths=36" You should obtain the following results: {"loanAmount":10000,"annualInterestRate":1.1,"termInMonths":36,"monthlyPaymentAmount":282.51360281363674}.

This API is not exposed, is not protected, is not monitored, is not governed. Let's use API Connect to fix this.

1. We are using the designer that we opened earlier. Click on *Develop APIs and Products* button. Then click on *Add* and select *API* button on the top right of the screen.

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

Select the *From existing OpenAPI service* and click *Next* button.

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

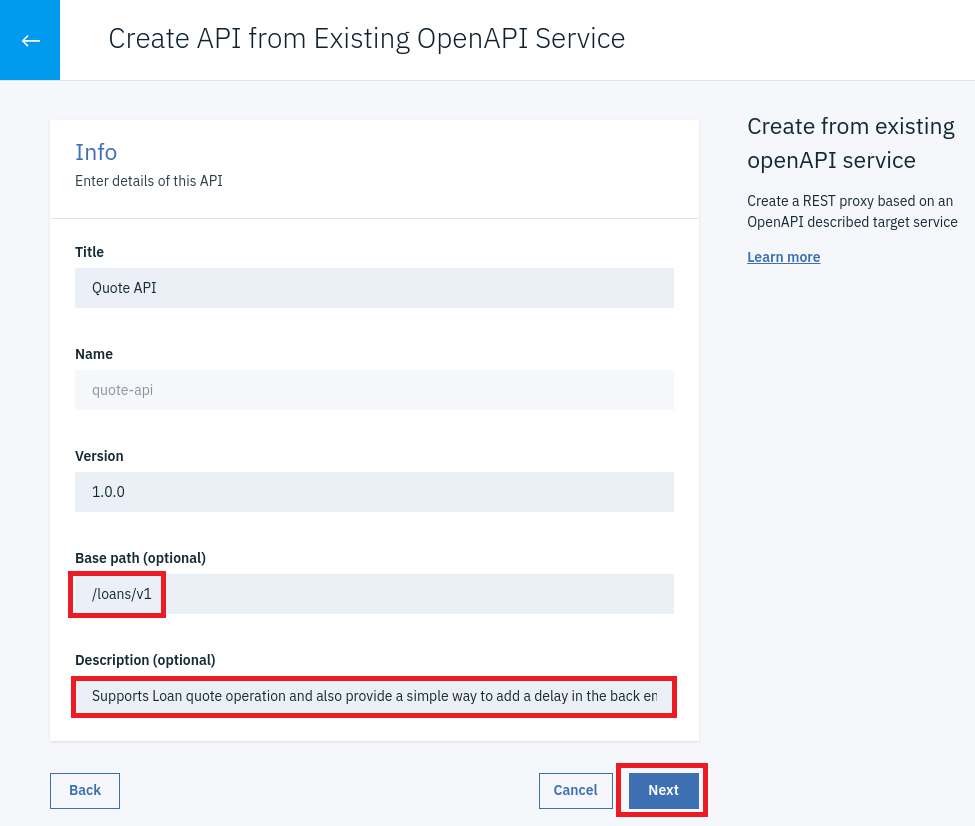
Select the file downloaded previously and click *Next* button.

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

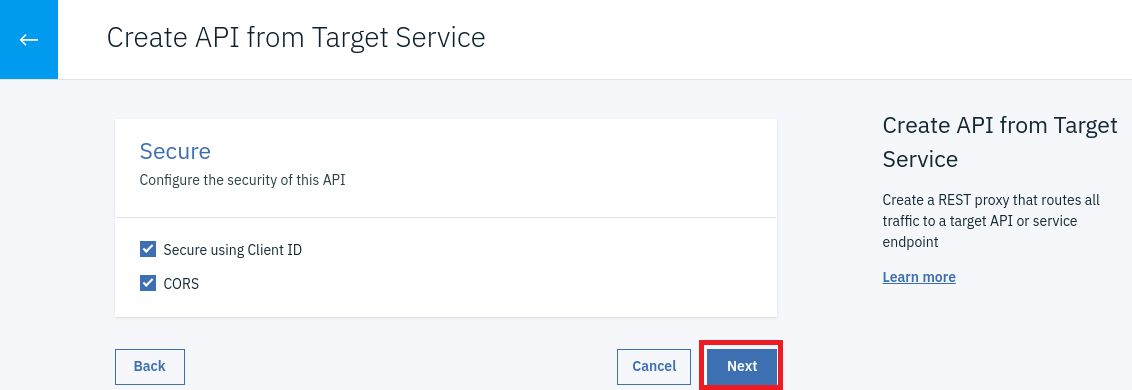
Change the following details

* Base path: /loans/v1
* Description: Supports Loan quote operation and also provide a simple way to add a delay in the back-end response time and get variable length messages from back-end. Then click *Next* button.

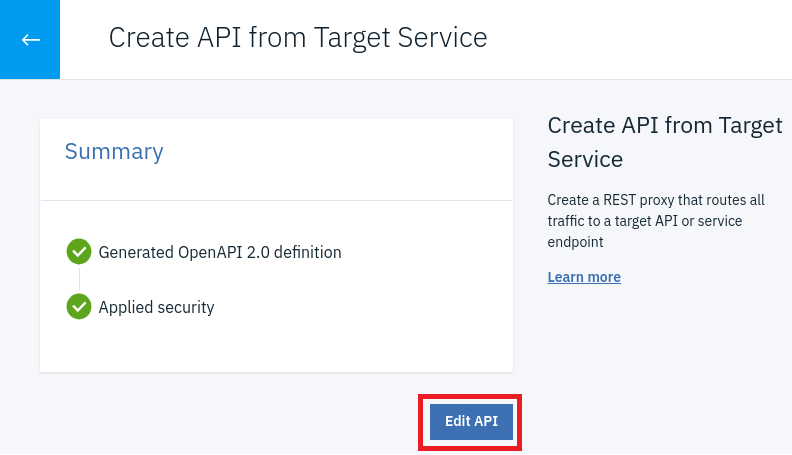
Hint: The base path has been chosen carefully to avoid URI rewriting and simplify this first example. You always have to be careful with the exposed URI and back-end URI and adopt strategy to avoid URI rewriting or if not possible to reduce the work required to do this mapping. The worst case is to have a specific URI for each combinations VERB + PATH.

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

Keep the default value for CORS and using a client id to secure the API, click *Next* button.

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

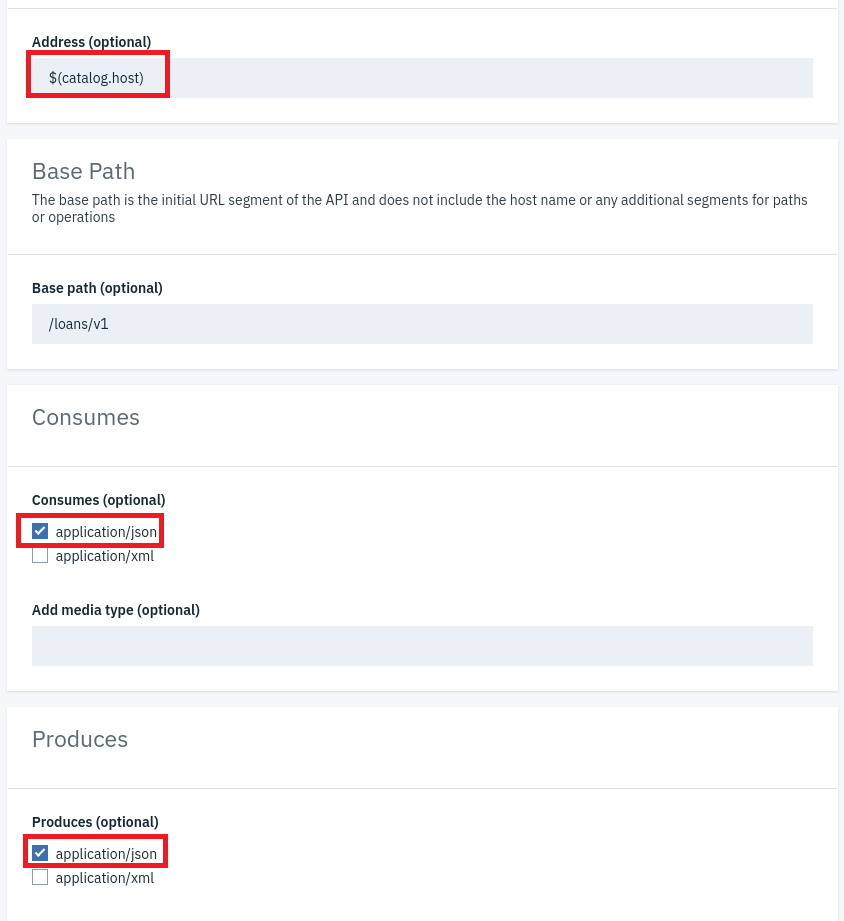
Click on the Edit *API button*.

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

Before testing it, in the development environment, let's review what has been created under the cover. The Designer can be considered in certain ways a Swagger (Open API) editor.

Let's see first the API Setup part. Notice that the Schemes supported by default is HTTPS, API Connect does not support HTTP scheme for security reason. Some specifications such as OAuth specifications do require the use of HTTPS scheme anyway.

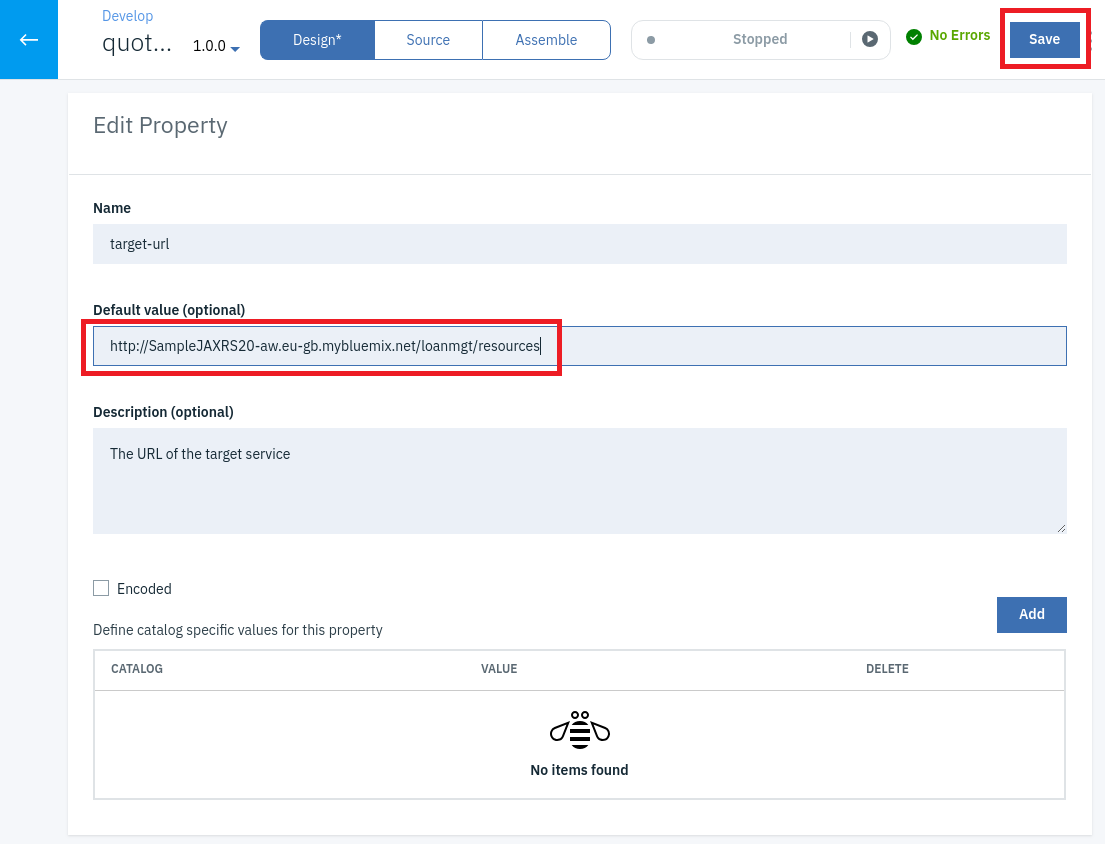
The host field has been set to $(catalog.host). This indicates where the API is deployed and it is dependent of where we deploy it, so it depends of the catalog, hence why this value for this variable.

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

A quick look at the Security Definitions and Security information, which are standard information within an Open API document shows that the API is as expected protected using an API Key, client-id only.

We see that there are 2 paths, /extquote (one verb, GET) and /quote (2 verbs GET and POST).

Now let's see the Properties section, there is a property called target-url. Properties is a very important concept. It allows the definition of any variable for each catalog. The target-url is by convention a variable to indicate the back-end url. In our case, we are going to adjust it to <http://SampleJAXRS20-aw.eu-gb.mybluemix.net/loanmgt/resources>. For now, we just use the default value, because we do not care of other catalogs than Sandbox. Click *Save* button.

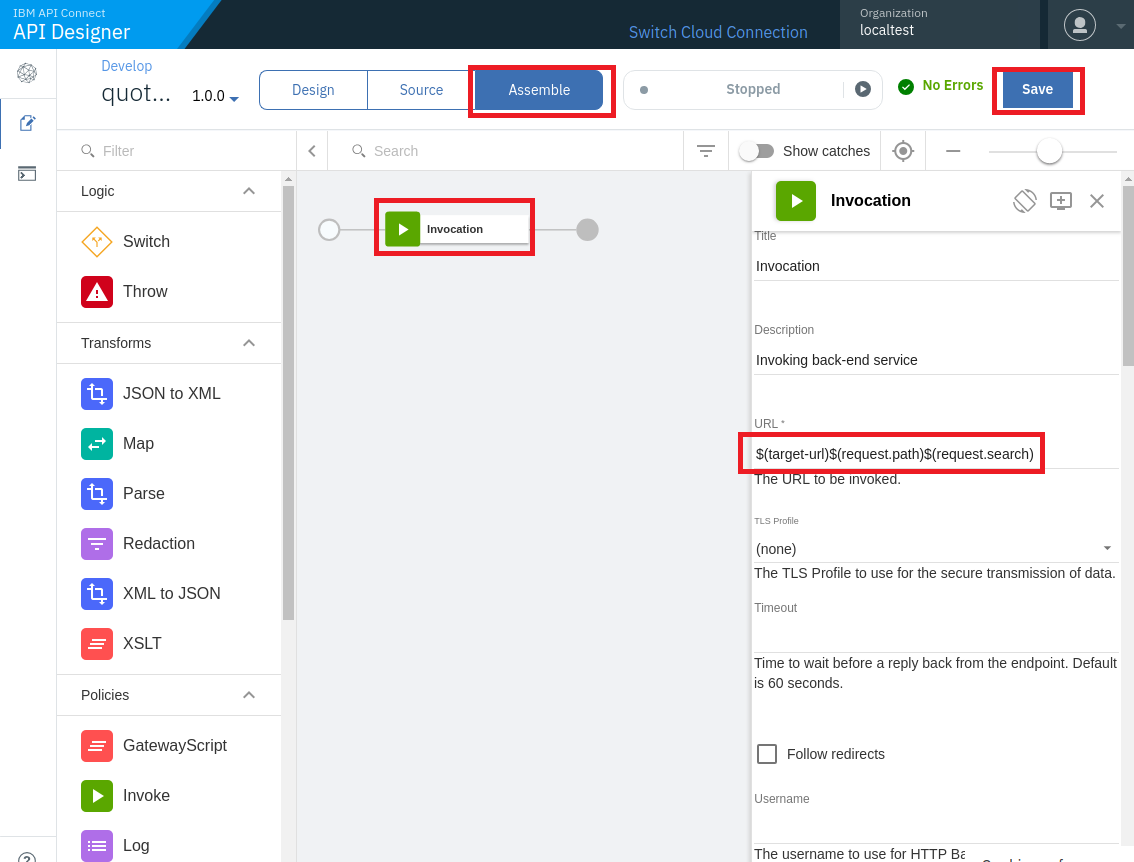
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-quote-api-properties.png)

Before testing we have one small adjustment to perform. The back-end URL invoked and we are going to use the target-url just set.

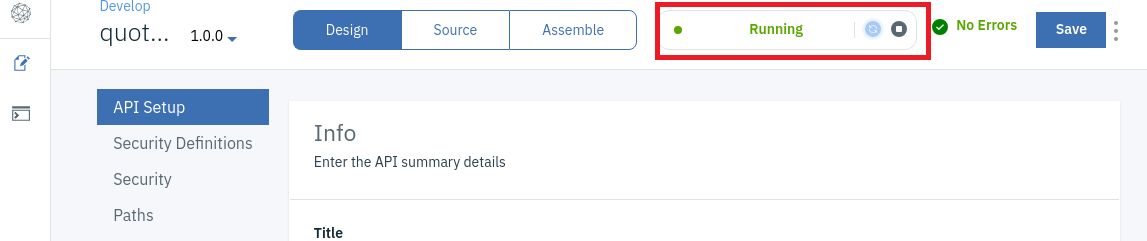
We go in the Assembly Panel and click on the Invocation policy. The panel with the properties is displayed on the right.

Hint: For compatibility of the gateway aspects, here delete the invoke policy and add it again. The choice here is based on which gateway type do we use. In our case, we use the new one, referred as DataPower API Gateway.

We change the value to $(target-url)$(request.path)$(request.search) and click *Save* button.

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

We can test the API which is available in the local Sandbox catalog with a generated auto product. We ensure that the API is running. If it stopped start it, so it goes in the running state.

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

In a terminal type curl -v -k -H "accept: application/json" -H "content-type: application/json" -H "x-ibm-client-id: c920f9c18395e6ecb3f15375a74fe8be" "https://localhost:9444/localtest/sandbox/loans/v1/quote?loanAmount=10000&annualInterestRate=1.1&termInMonths=3"

You should get as before when accessing the back-end API directly: {"loanAmount":10000,"annualInterestRate":1.1,"termInMonths":3,"monthlyPaymentAmount":3339.4463108727305}

Before moving on, let's discuss some debugging techniques.

In order to do this, we are introducing a few errors and see what we can do to handle them and how to understand them.

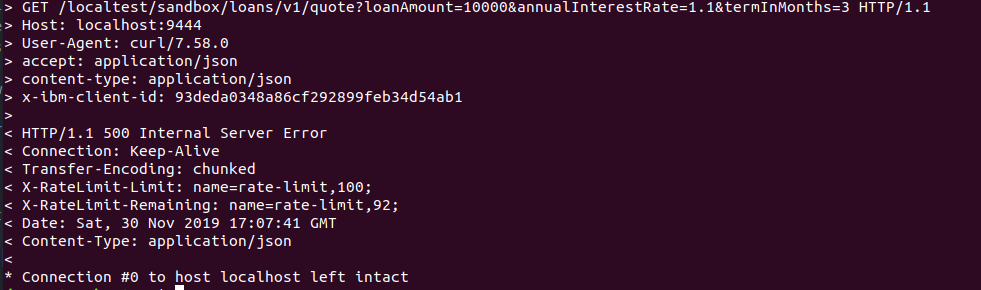
Using the wrong client-id: {"httpCode":"401","httpMessage":"Unauthorized","moreInformation":"Invalid client id or secret."} This is situation is pretty clear. Get an unauthorized message. One way to get a little bit more information is to use the -v option with curl. This will show you the headers sent and received, the TLS session information, etc ...

Now let's use a wrong host or URI for the back-end.

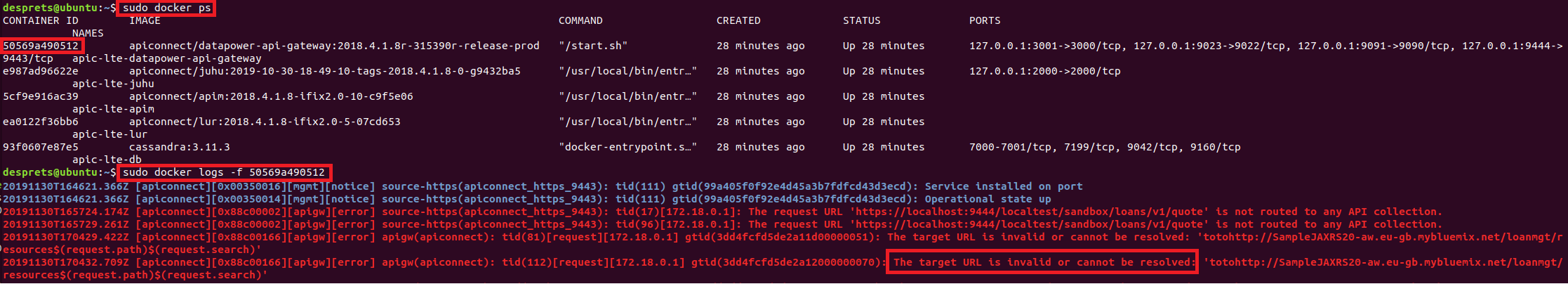
[Trouble shoot 1](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/troubleshoot-wrong-uri-1.png)

No information whatsoever!

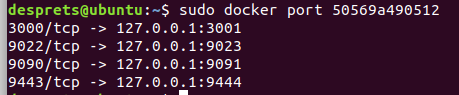
Let's use the -v option.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/troubleshoot-wrong-uri-2.png)

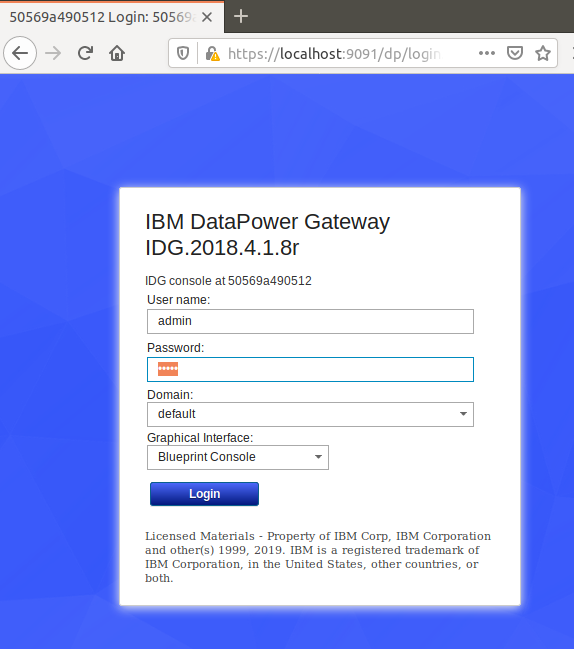
Now we see that there was a 500 error. This is better. We do not see any root cause, there is no problem with the plan (still 92 calls possible). It is not clear that the back-end URI is wrong. So, let's see the logs from the Gateway itself. We know that we are running DataPower as a docker container. So, let's get the container id by issuing sudo docker ps, then now we can check the logs of the gateway using the sudo docker logs -f <gateway-container-id>. (To get the gateway container id, issue the command sudo docker ps). It becomes very clear that the error is the URL...

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/troubleshoot-wrong-uri-3.png)

**Hint:** You can determine the port mapping for the gateway container and derive from it the gateway web console knowing that the default internal port for the web UI is 9090. Issue the command sudo docker port <gateway-container-id>

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/troubleshoot-wrong-uri-4.png)

You can then access the console at <https://localhost:9091/>, the default uid/pwd is admin/admin.

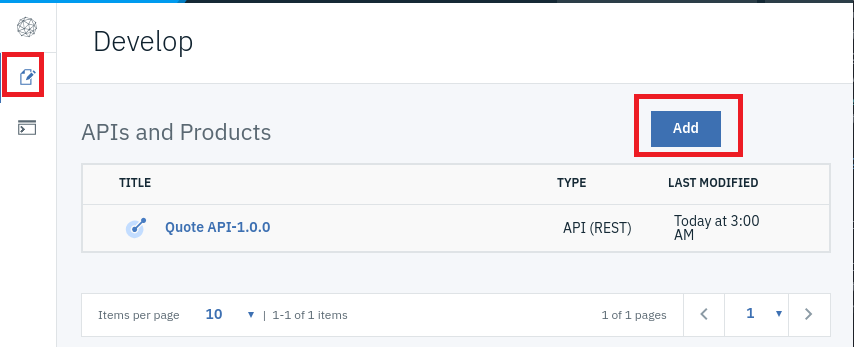
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/troubleshoot-wrong-uri-5.png)

Let's fix the URI before publishing the API to the remote manager since we are happy now that the API is correctly working.

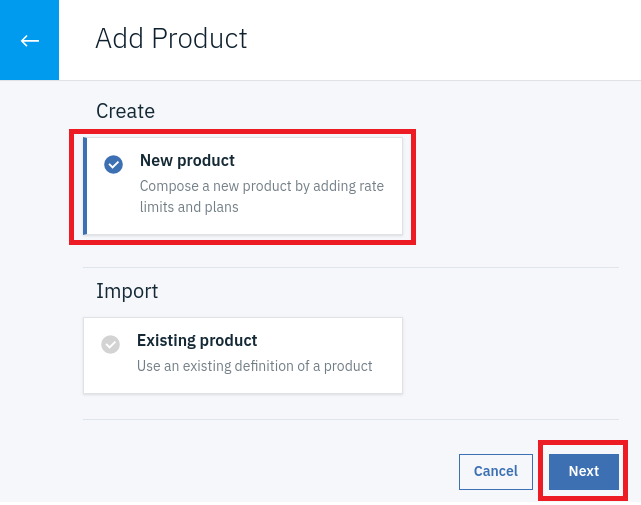
The API is definition is complete. We need now to add the Product and publish it and then we are ready to test our Quote API before publishing it to the remote Manager.

**Step 3 - Creating and publishing a Product**

To create a product, click on the Develop icon on the navigation panel (left). Then click on the *Add* button and select Product.

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

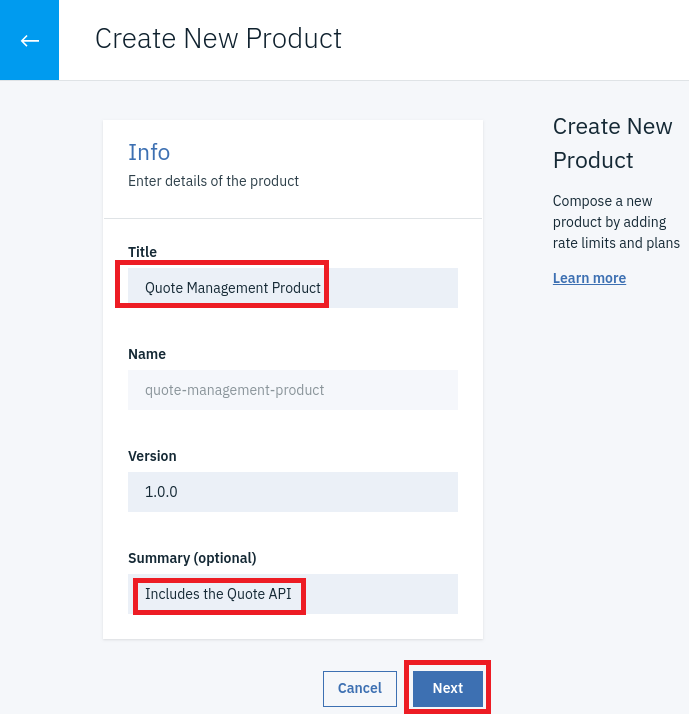
Select *New product* button and *Next* button.

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

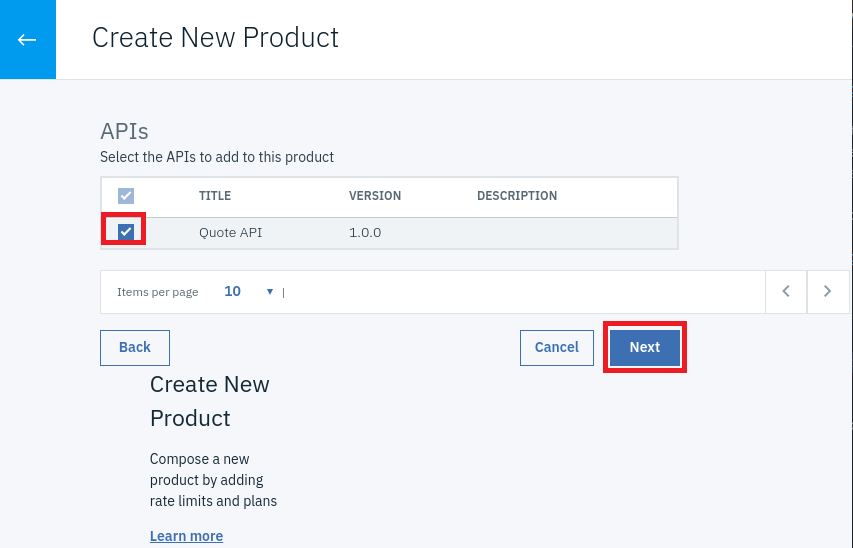
Enter the following information:

* Title: Quote Management Product
* Summary: Includes the Quote API

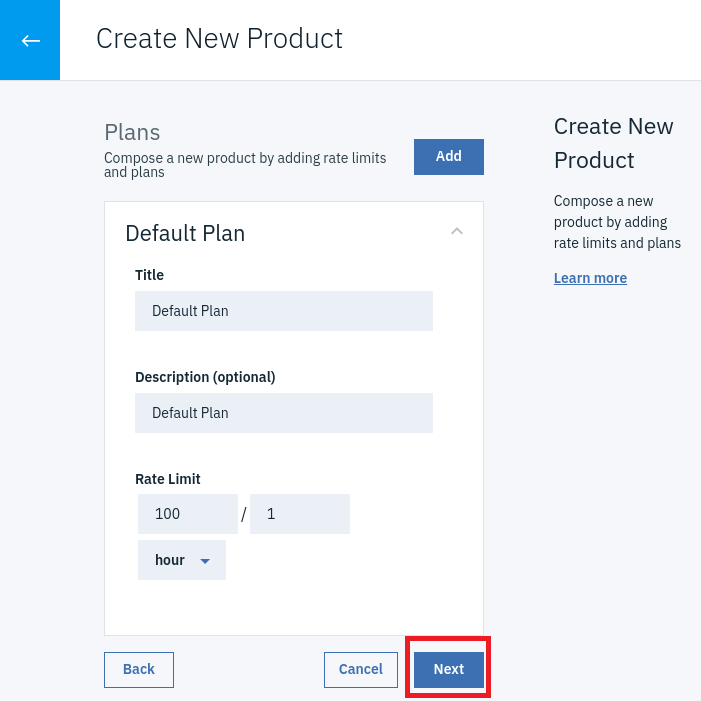
Click on *Next* button

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

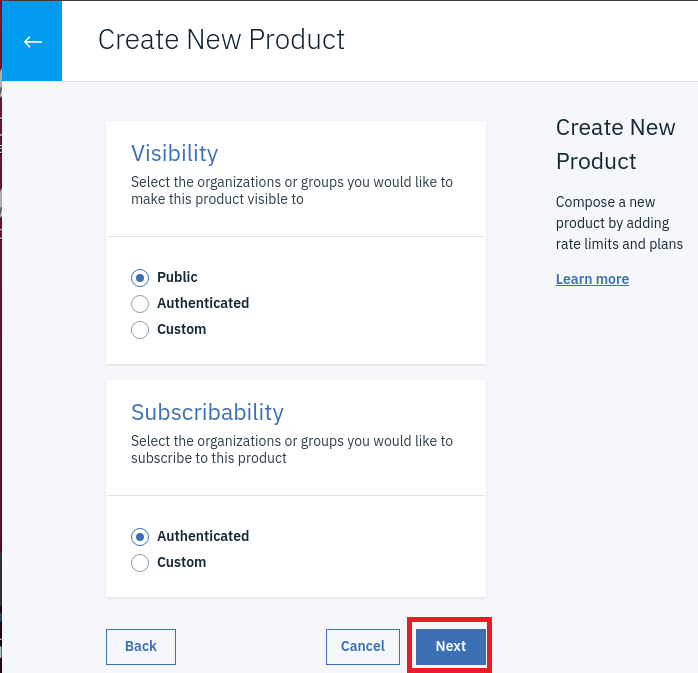
Select the Quote API by clicking on the check box and the click *Next* button.

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

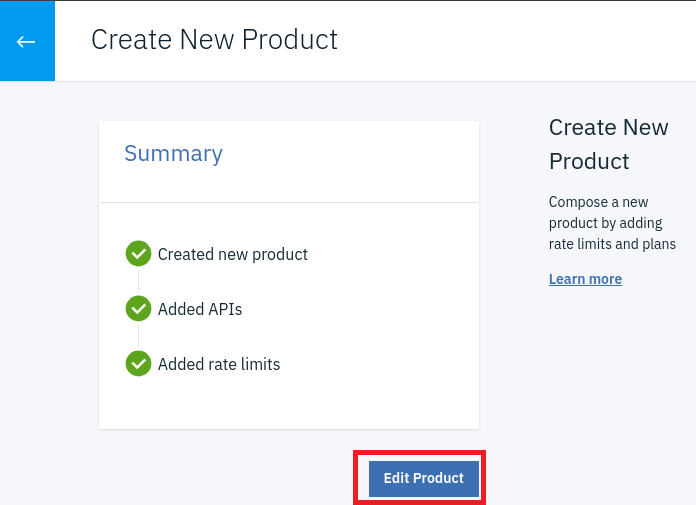
Change or adjust the plan according your requirements plan and then click on *Next* button.

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

Change or adjust the visibility or subscribability according your requirements plan and then click on *Next* button.

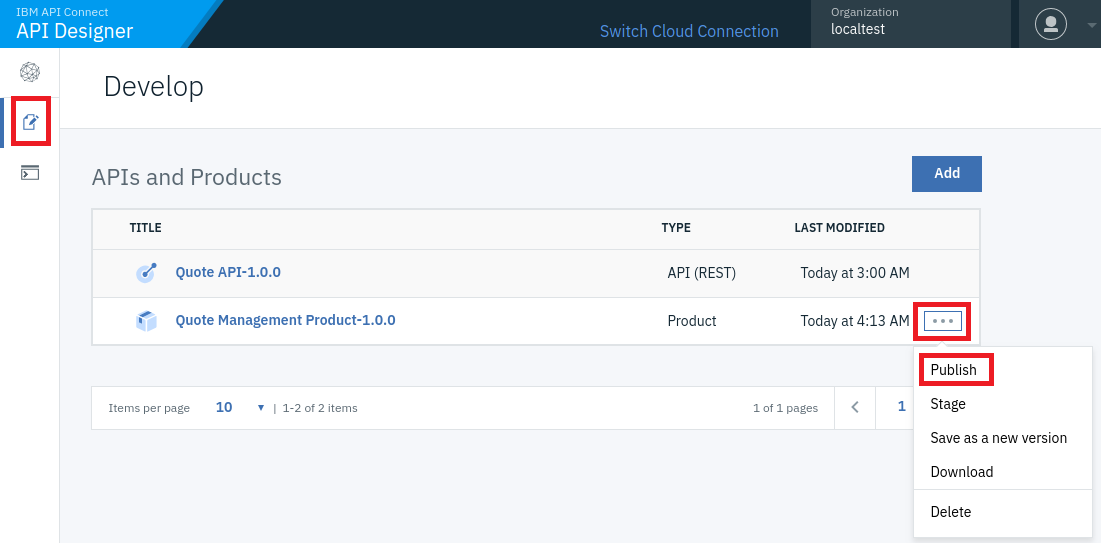
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-new-product-visibility.png)

click on *Edit Product* button.

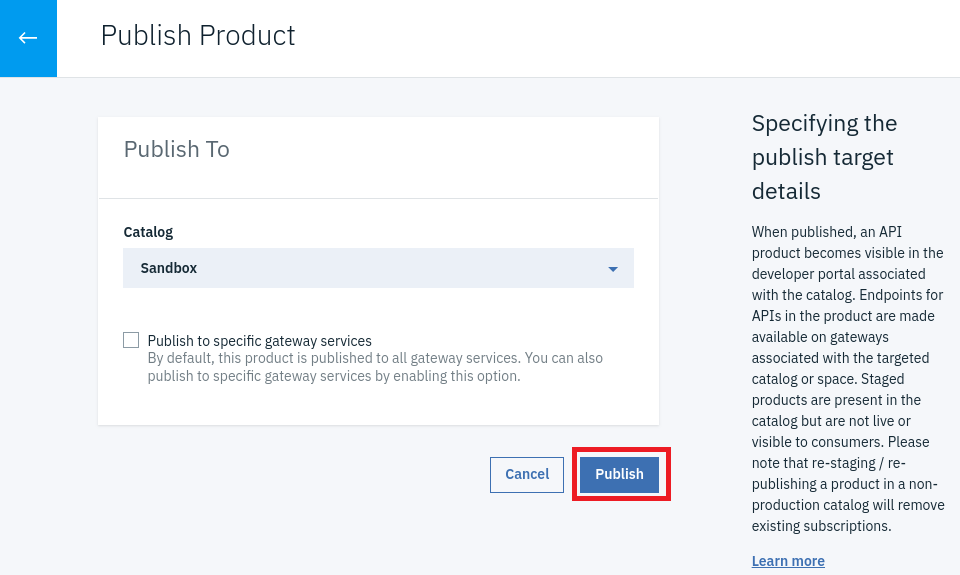
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-new-product-done.png)

We want to publish the API on the remote manager. So first, we start to add this manager to the Designer so we can choose where we want to deploy the product. We click on the *Switch cloud connection* link at the top of the window and then click on *Add Another Cloud* button. Enter the URL of the remote manager, in my case, <https://manager.159.8.70.38.xip.io>, then enter the credentials to access the organization you work with.

We publish the API, by clicking on the Develop icon, then clicking on the ... close to the new product and select Publish.

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

Then click on the *Publish* button.

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

We can check on the remote Manager that the Product containing the Quote API has been correctly published.

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

We want to automate the publishing and do not use the Designer and instead the CLI.

**Hint:** If you want to use the REST API, it is easy to use the CLI with the --debug option at the end. This will show you exactly the REST commands issued under the cover.

For example:

apic login -s manager.159.8.70.38.xip.io -u org1owner -p \*\*\*\*\*\*\*\* -r provider/default-idp-2 --debug

2020/03/17 07:27:11 CURL:

curl -X 'POST' -d '{"client\_id":"599b7aef-8841-4ee2-88a0-84d49c4d6ff2","client\_secret":"0ea28423-e73b-47d4-b40e-ddb45c48bb0c","grant\_type":"password","password":"\*\*\*\*\*\*\*\*","realm":"provider/default-idp-2","username":"org1owner"}

' -H 'Accept: application/json' -H 'Accept-Language: en-us' -H 'Content-Type: application/json' -H 'User-Agent: Toolkit/c81e13c07d3c2c7730827610fcaf08bbec88fe04' -H 'X-Ibm-Client-Id: 599b7aef-8841-4ee2-88a0-84d49c4d6ff2' -H 'X-Ibm-Client-Secret: 0ea28423-e73b-47d4-b40e-ddb45c48bb0c' 'https://manager.159.8.70.38.xip.io/api/token'

, Request dump:

POST /api/token HTTP/1.1

Host: manager.159.8.70.38.xip.io

User-Agent: Toolkit/c81e13c07d3c2c7730827610fcaf08bbec88fe04

Content-Length: 211

Accept: application/json

Accept-Language: en-us

Content-Type: application/json

X-Ibm-Client-Id: 599b7aef-8841-4ee2-88a0-84d49c4d6ff2

X-Ibm-Client-Secret: 0ea28423-e73b-47d4-b40e-ddb45c48bb0c

Accept-Encoding: gzip

, Request body:

{"client\_id":"599b7aef-8841-4ee2-88a0-84d49c4d6ff2","client\_secret":"0ea28423-e73b-47d4-b40e-ddb45c48bb0c","grant\_type":"password","password":"\*\*\*\*\*\*\*\*","realm":"provider/default-idp-2","username":"org1owner"}

2020/03/17 07:27:11 Response dump:

HTTP/1.1 200 OK

Transfer-Encoding: chunked

Access-Control-Allow-Credentials: true

Access-Control-Allow-Headers: DNT,X-CustomHeader,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Authorization

Access-Control-Allow-Methods: GET, PUT, POST, DELETE, PATCH, OPTIONS

Access-Control-Allow-Origin: \*

Cache-Control: no-store

Connection: keep-alive

Content-Type: application/json; charset=utf-8

Date: Tue, 17 Mar 2020 14:27:11 GMT

Etag: W/"796-uuzjeFcKQJgIb4beYqsTqyRZQww"

Pragma: no-cache

Strict-Transport-Security: max-age=31536000; includeSubDomains;

Vary: Accept-Encoding

X-Request-Id: 6c9542e7c7951b4512235eb5d404740a

796

{

"access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..PJ29zR15LMo1TcpF\_qc2iAISRnubNF4MkTD2SfQxnTg",

"token\_type": "Bearer",

"expires\_in": 28800

}

0

Logged into manager.159.8.70.38.xip.io successfully

2020/03/17 07:27:11 CURL:

curl -X 'GET' -H 'Accept: application/yaml' -H 'Accept-Language: en-us' -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..PJ29zR15LMo1TcpF\_qc2iAISRnubNF4MkTD2SfQxnTg' -H 'User-Agent: Toolkit/c81e13c07d3c2c7730827610fcaf08bbec88fe04' -H 'X-Ibm-Client-Id: 599b7aef-8841-4ee2-88a0-84d49c4d6ff2' -H 'X-Ibm-Client-Secret: 0ea28423-e73b-47d4-b40e-ddb45c48bb0c' 'https://manager.159.8.70.38.xip.io/api/me?fields=force\_password\_change'

, Request dump:

GET /api/me?fields=force\_password\_change HTTP/1.1

Host: manager.159.8.70.38.xip.io

User-Agent: Toolkit/c81e13c07d3c2c7730827610fcaf08bbec88fe04

Accept: application/yaml

Accept-Language: en-us

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..PJ29zR15LMo1TcpF\_qc2iAISRnubNF4MkTD2SfQxnTg

X-Ibm-Client-Id: 599b7aef-8841-4ee2-88a0-84d49c4d6ff2

X-Ibm-Client-Secret: 0ea28423-e73b-47d4-b40e-ddb45c48bb0c

Accept-Encoding: gzip

, Request body:

2020/03/17 07:27:11 Response dump:

HTTP/1.1 200 OK

Content-Length: 2

Access-Control-Allow-Credentials: true

Access-Control-Allow-Headers: DNT,X-CustomHeader,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Authorization

Access-Control-Allow-Methods: GET, PUT, POST, DELETE, PATCH, OPTIONS

Access-Control-Allow-Origin: \*

Connection: keep-alive

Content-Type: application/json; charset=utf-8

Date: Tue, 17 Mar 2020 14:27:11 GMT

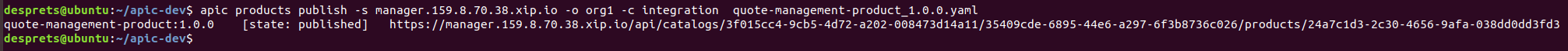
Etag: W/"2-vyGp6PvFo4RvsFtPoIWeCReyIC8"

Strict-Transport-Security: max-age=31536000; includeSubDomains;

Vary: Accept-Encoding

X-Request-Id: 68312d45ecdaa0a9cbf972f5dd8a49c5

{}

First let's make sure with in the right directory (where the swaggers are created), in my case, cd apic-dev, then let's login to the remote manager with apic. apic login -s manager.159.8.70.38.xip.io -u org1owner -p \*\*\*\*\*\*\*\*\* -r provider/default-idp-2 then we are ready to publish the product into the Integration environment for example. We issue the command: apic products publish -s manager.159.8.70.38.xip.io -o org1 -c integration quote-management-product\_1.0.0.yaml [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/cli_publish.png)

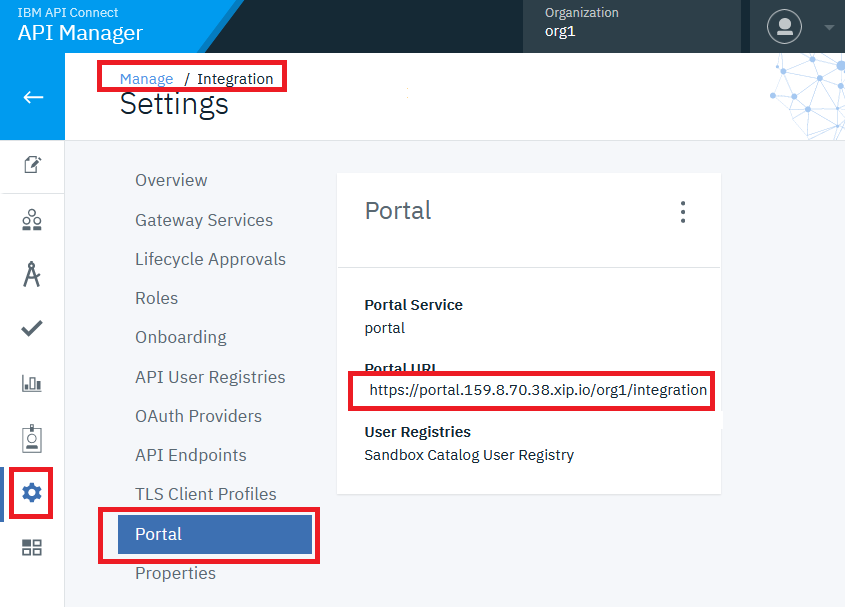
**Step 4 - Consumer Experience**

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. We are referring to the Portal that is associated with the "remote" API Connect Cloud.

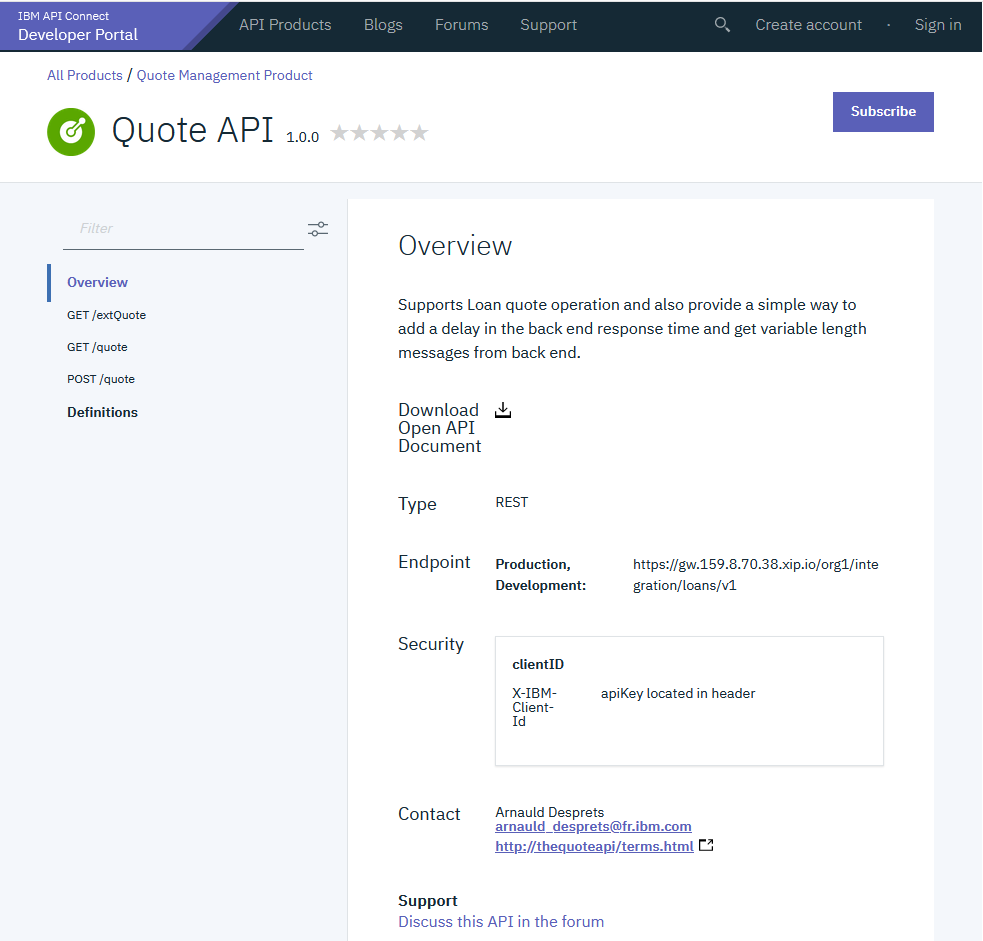
**Open the Portal login page**

You can get the URL of the portal associated to a catalog in the settings of this catalog.

1. Go to the API Manager screen, in my case, <https://manager.159.8.70.38.xip.io>.
2. Click on Manage and click on the Integration Catalog tile (created previously and we assume that the Portal was instantiated for this catalog here)
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-portal2018.png)

1. Click on **Quote Management Product 1.0.0 Product** **API Products** to explore the API

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

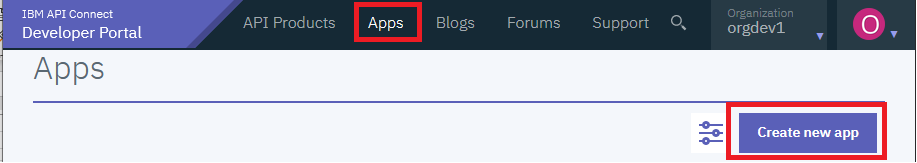
**Register an Application as a developer**

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.

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

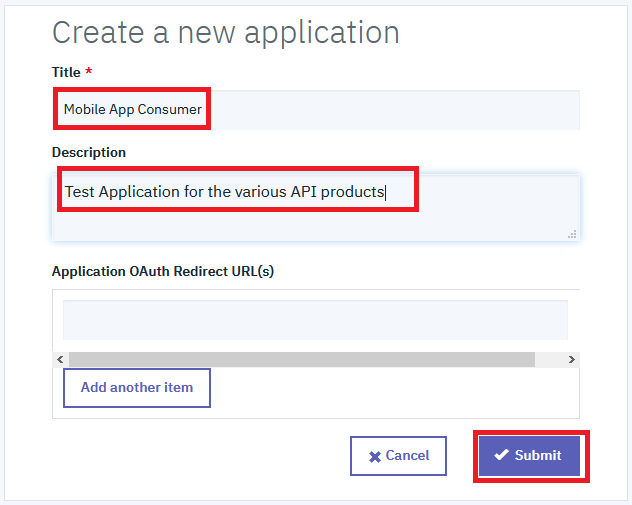
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/portal_createAppButton.png)

1. Enter a title and description for the application and click the Submit button.

Title: Mobile App Consumer

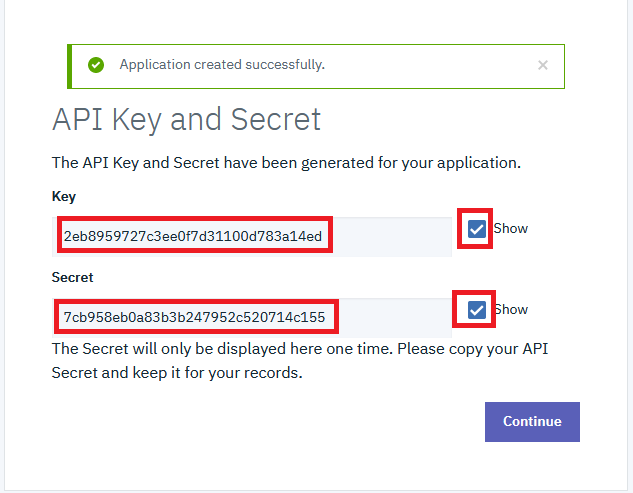
Description: Test Application for the various API products

OAuth Redirect URI: < leave blank >

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

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 check boxes for the Key and Secret.

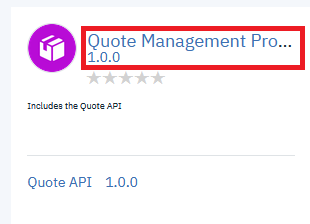
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/portal_createAppKeys.png)

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

**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 Quote Management Product (1.0.0) API product link.

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

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

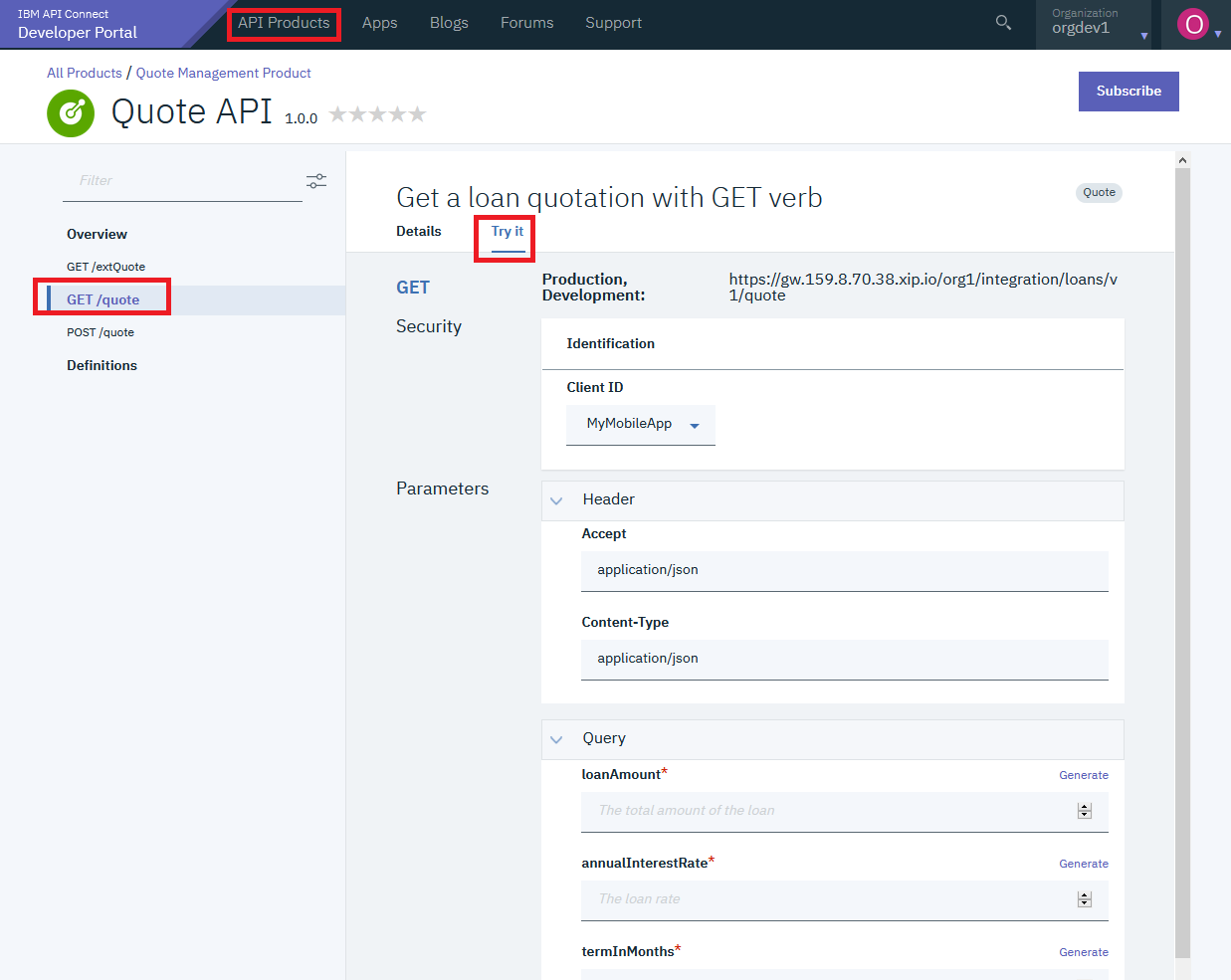
1. Click on the **Subscribe** button under the ***Default plan***.
2. Select the **Mobile App Consumer** toggle and click the **Subscribe** button.

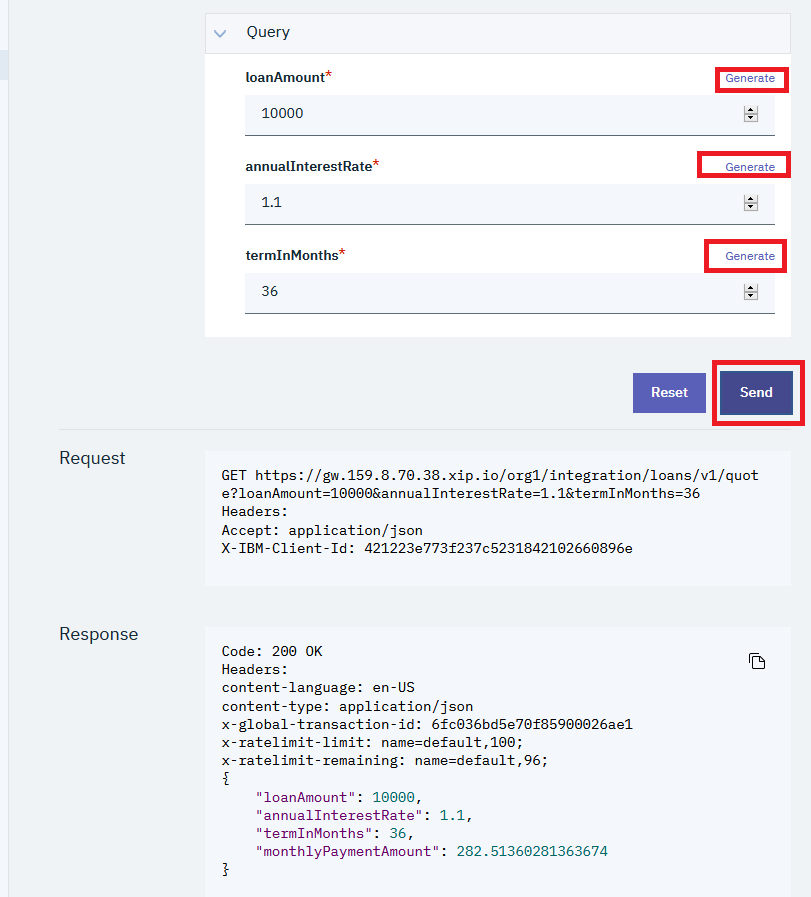
The MobileApp Consumer application is now subscribed to the **Default plan** for the QuoteMgmt product.

**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 API** link on the left-hand navigation menu and then expand the GET /quote path by clicking on the twisty next to the path, click on the Try it tab.

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

1. Scroll down, click on the three *Generate* link to populate the values and click the Send button
2. Scroll down below the Call operation button. You should see a 200 OK and a response body as shown below. You get the response from the back-end. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/portal-testsend.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 API (1.0.0)** API link within the Quote Management Product.
4. Now, you can discover all operations with their properties and on the right hand side sample code.
5. Select an Operation, for example, **POST /quote**
6. On the right hand side you'll see the **curl** expression
7. Copy it into your text editor window replacing **Client ID** with your client id and your client secret saved from the prior step

curl --request POST \

--url https://gw.159.8.70.38.xip.io/org1/integration/loans/v1/quote \

--header 'accept: application/json' \

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

--header 'x-ibm-client-id: Client ID' \

--data '{"loanAmount":"10000","annualInterestRate":"1.1","termInMonths":"36"}'

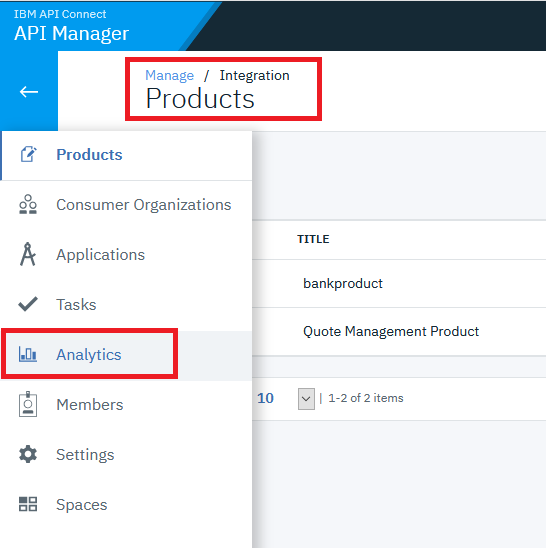
1. Copy and try it into your terminal windows

If all is OK, you should see the result of the quote in JSON format. (Below slightly modified for Windows)

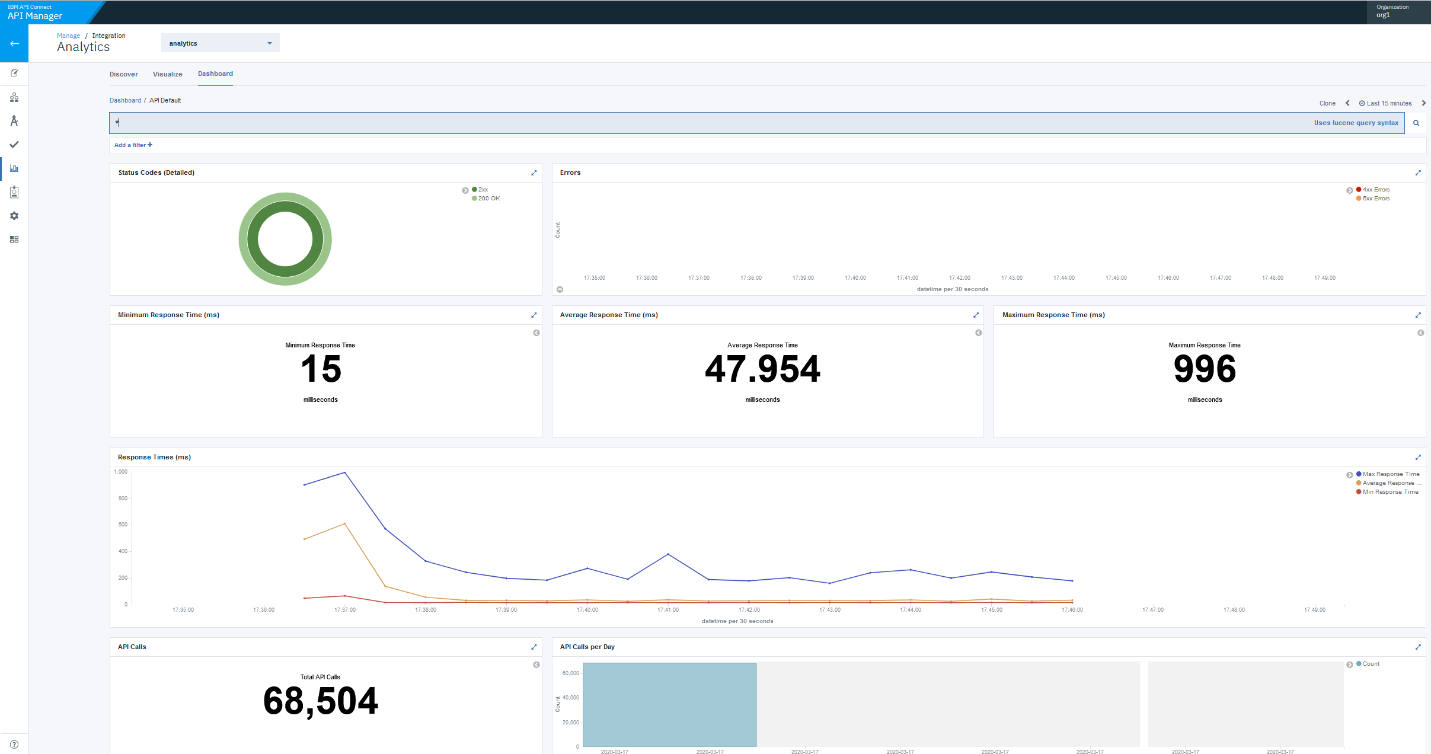
[Portal curl](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/portal-testcurl.png)

**Step 6 - Analytics**

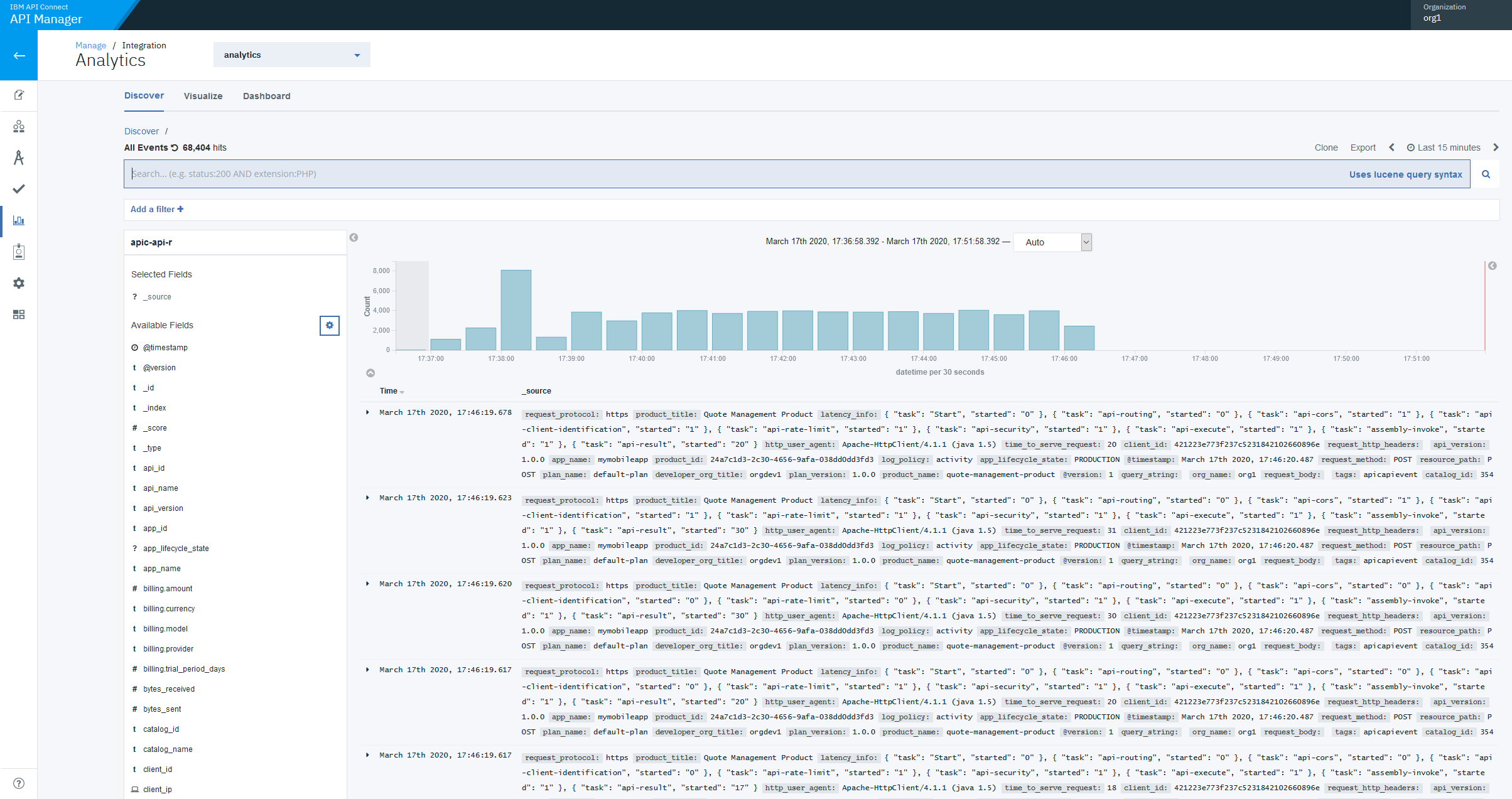
1. Return to the API Manager screen.
2. Click on the Manage Catalogs tile and click on the **Integration** catalog tile.
3. Click on the Analytics navigation menu.

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

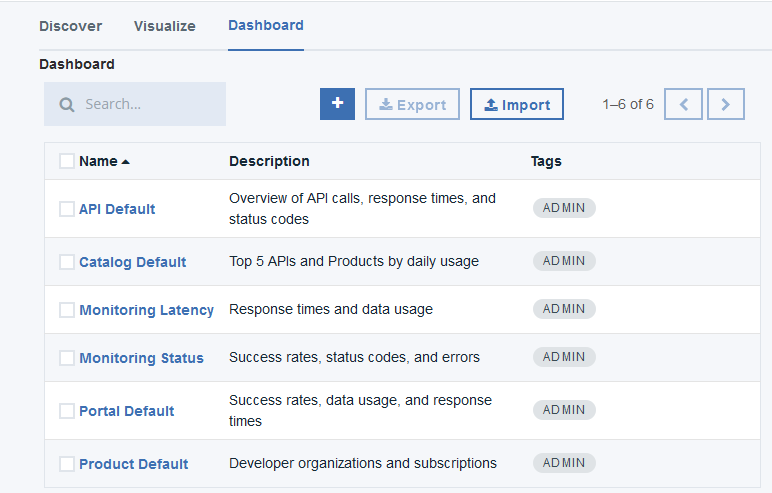
1. Select the API Default dashboard.

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

1. You can also look at all the events, click on Discover link and then on All events link

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

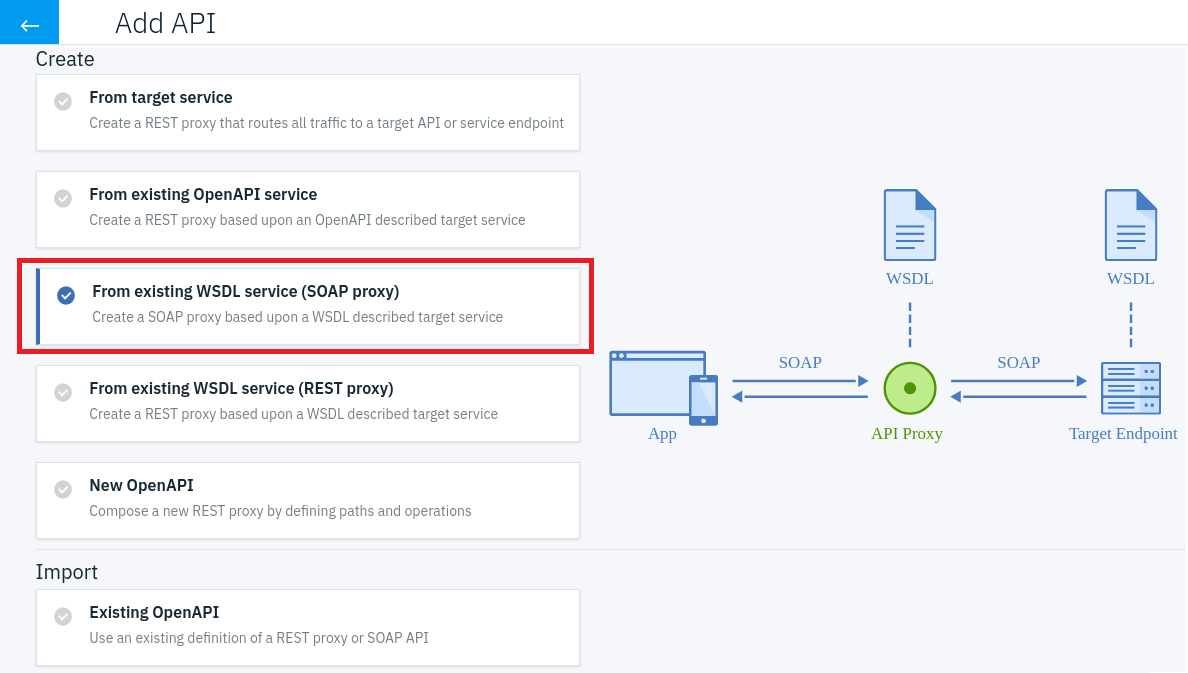
There are many dashboards provided out of the box and also events discoveries that focus on specific aspects such as the latencies, the errors, the data sizes, etc... Do not hesitate to explore the various dashboards and events lists. You can create your own dashboards and events lists with predefined filters.

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

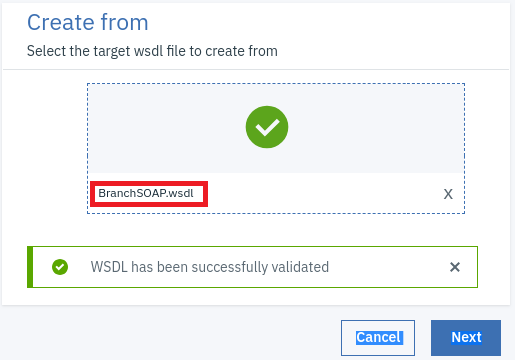
**Step 7 - Create a SOAP API**

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. Before starting, it is important to understand what we try to achieve. In the designer (or the Manager), you find two different possibilities regarding the integration of an existing SOAP API from the WSDL defining this service. The first one is a SOAP proxy which means that we create an API (An Open API document) that exposes SOAP and accesses the SOAP service back-end, the second one is a REST proxy which means that we create an API (An Open API document) that exposes REST/JSON and accesses the SOAP service back-end. This is a very different scenario, we will discuss the difference of implementation in the next Step.

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). You can also get the WSDL at the following URL <https://addressmanagementwebservice.eu-gb.mybluemix.net/branches/Branches?WSDL>
2. In the Designer, click on Add API and select From existing WSDL service (SOAP proxy)

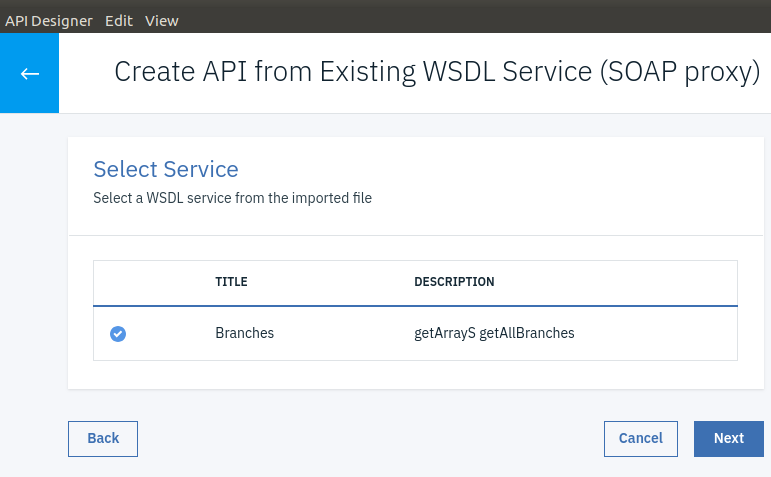
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/manager-soap-create.png)

1. Select the BranchSOAP.wsdl file to load

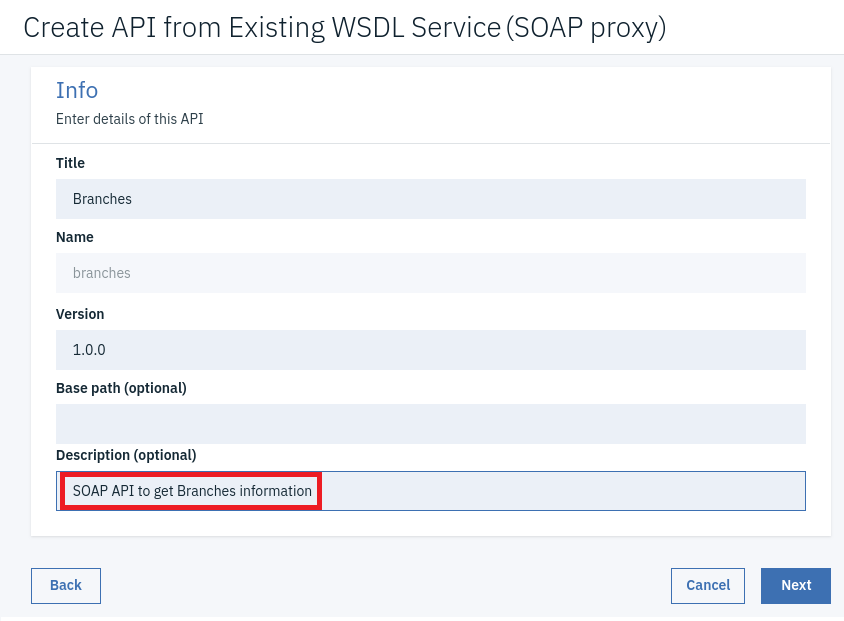
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-soap-create-load.png)

and click Next.

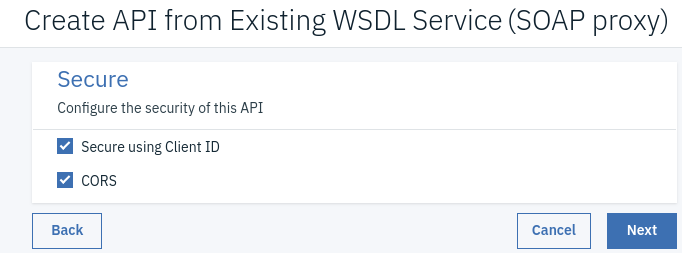
1. You can check the Service defined in the WSDL, click Next

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

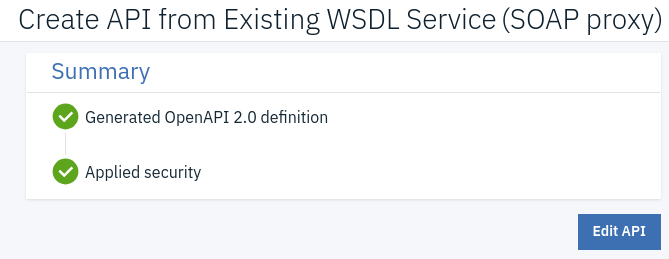
1. Review the API definitions such as name, version and description For example, add SOAP API to get Branches information for the description field, click Next button.

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

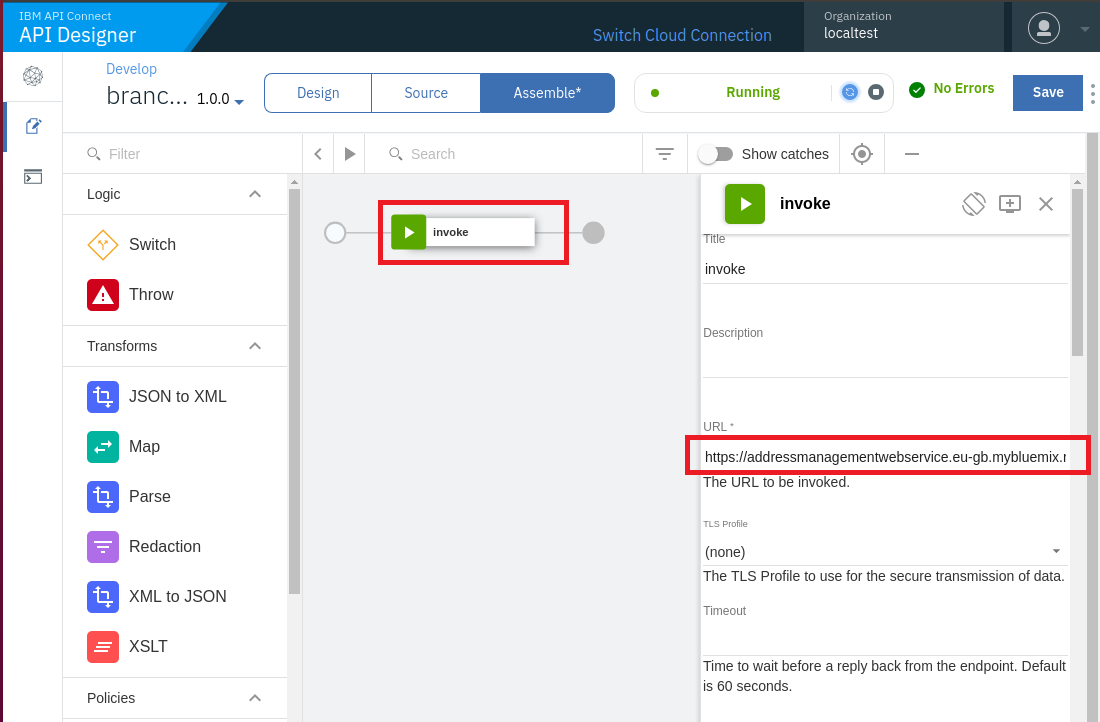
1. Review the Security and CORS definitions, click Next

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

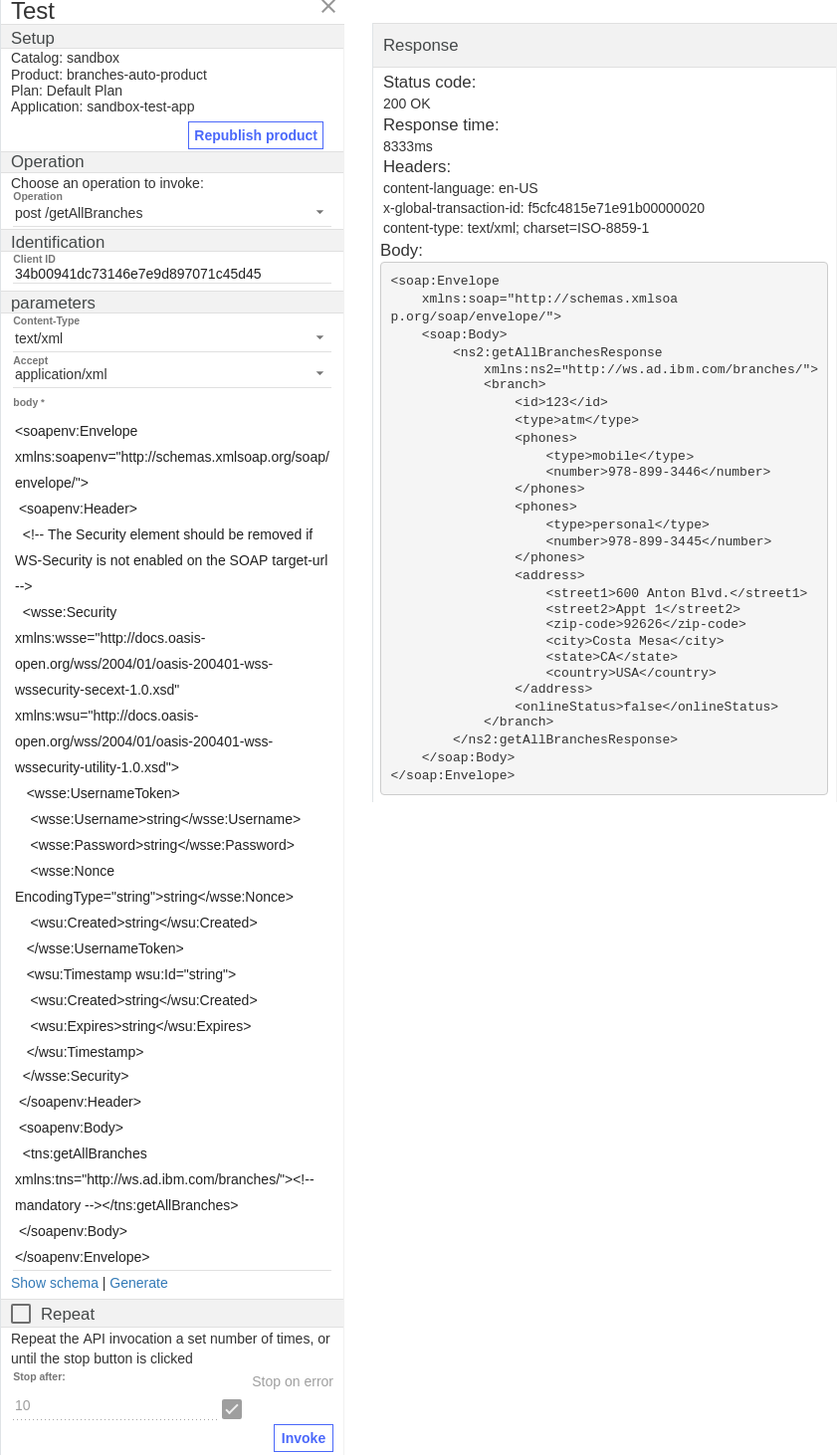
1. Click Edit API button

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

You can see what has been generated under the cover. As expected, the consume content type is text/xml and produce content type is application/xml in the assembly panel. In the assembly panel we see only one Invoke action, the invoke URL is the one extracted from the WSDL.

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

We are going to test it. Go to the Assembly panel. Click on the Test icon and the Activate API button. Then choose an operation to test for example, post /getAllBranches, click on the generate link for the body parameter and click Invoke button. You should get a response from the SOAP backend service.

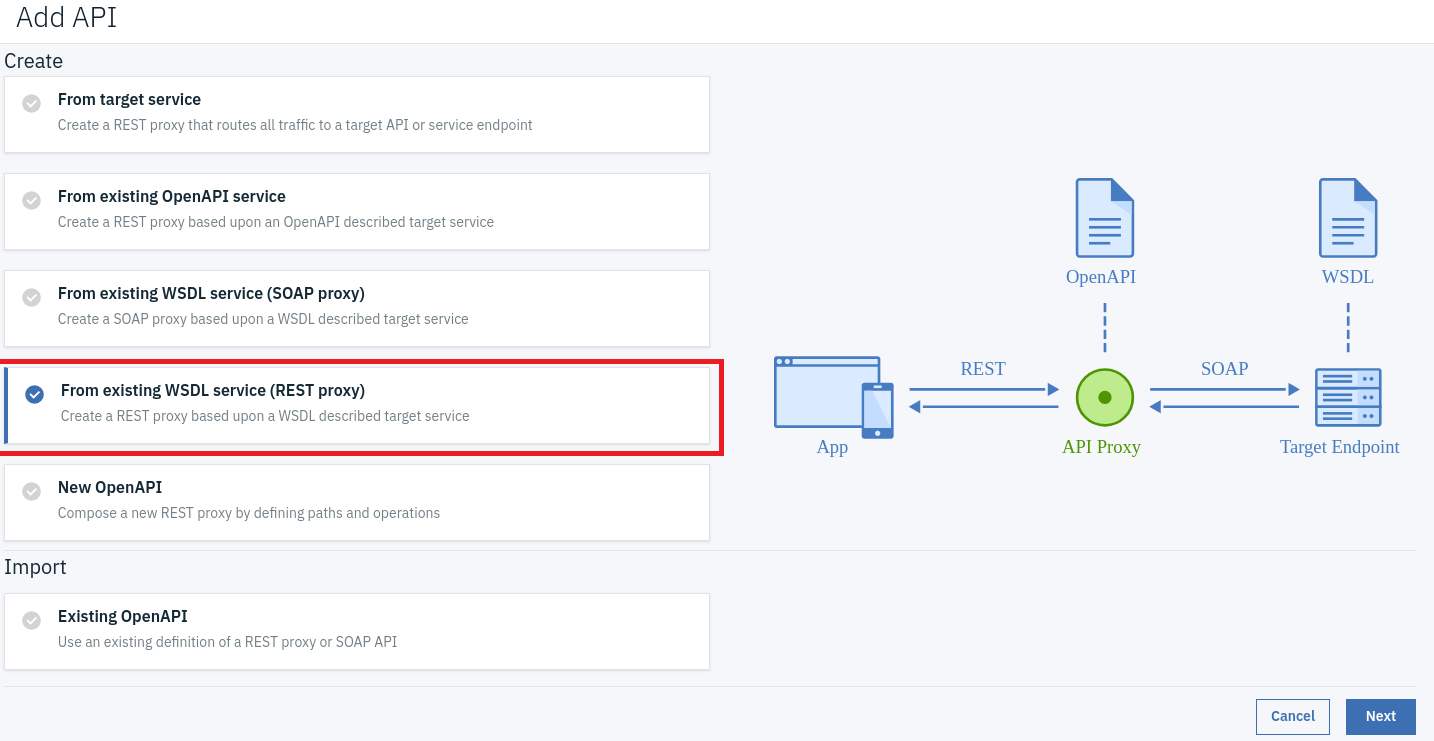
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-soap-proxy-test-allB.png)

**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 property that will point to the right endpoint depending on the environment. We did not publish the service in the integration catalog on the remote Manager, since this is exactly the same procedure as for a REST Service.

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

With the 2018 version, this step has been greatly simplified. Similarly to the previous step, we use the designer but instead of using SOAP proxy, we use a REST proxy. We will see that API Connect generates a lot of configurations automatically. 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 another view and call it REST to SOAP :-). We are not documenting all the steps to add the API because it is very similar to the previous step, the only difference being the selection of REST proxy at the very beginning.

1. 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/designer-soap-rest-create.png)

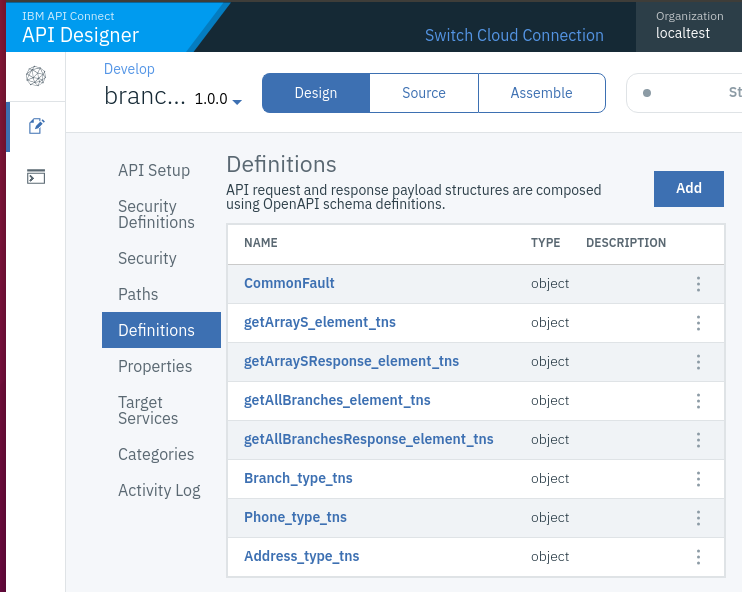
The next steps are as before. For the Info panel, I used:

Title: BranchesREST

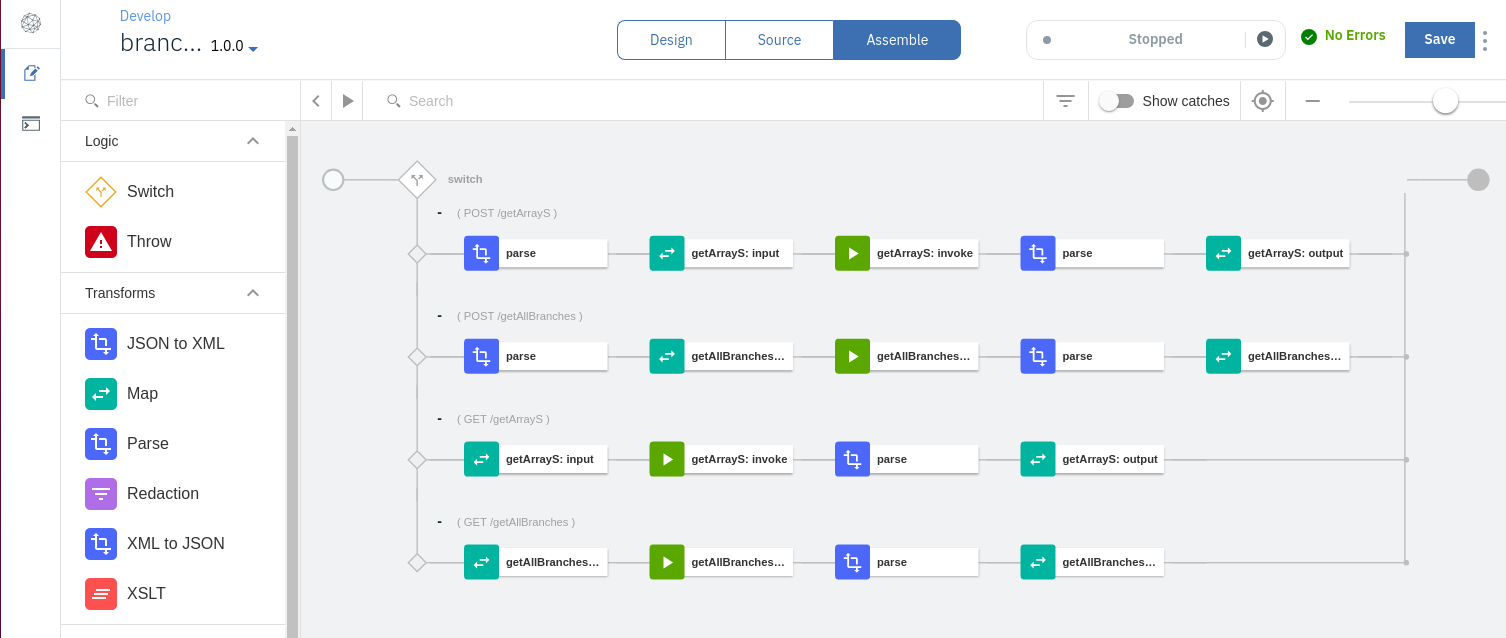
Description: REST exposure for the back-end SOAP service

At the end click on the Edit API button.

Let's see two seconds the definition, unlike the previous step, the consume and produce content-types are now application/json. The definitions have been auto generated. And we can see for the XSD elements corresponding JSON elements. This is a great feature and nice enhancement from the V5 version.

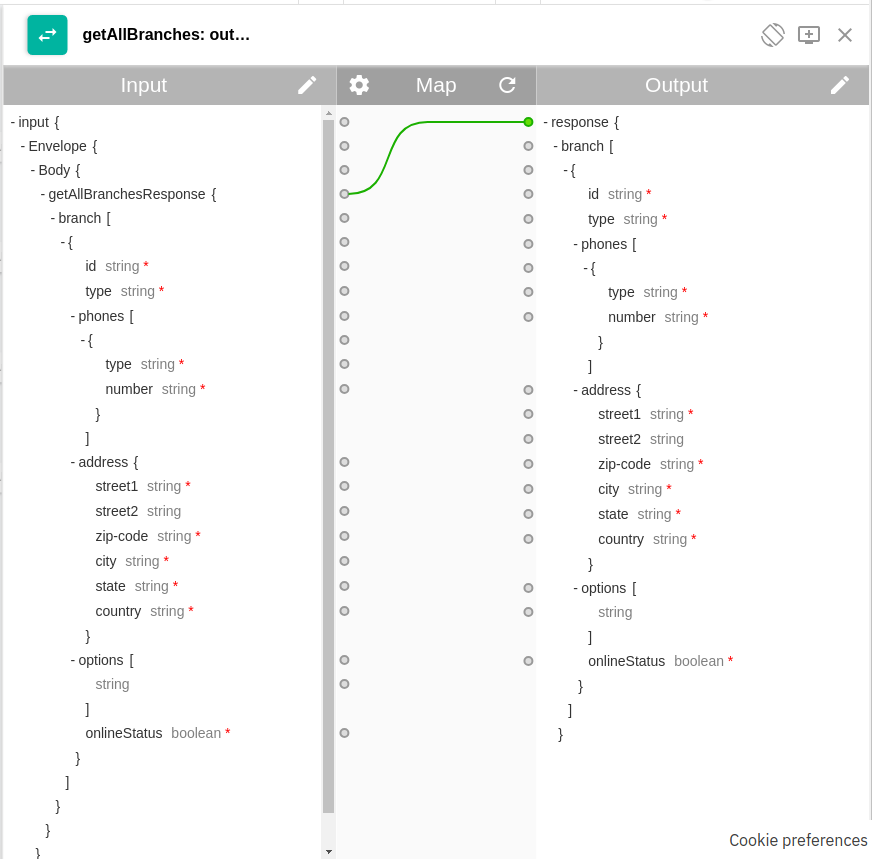
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-soap-rest-definitions.png)

We can see that in the Assembly, it is quite different than before.

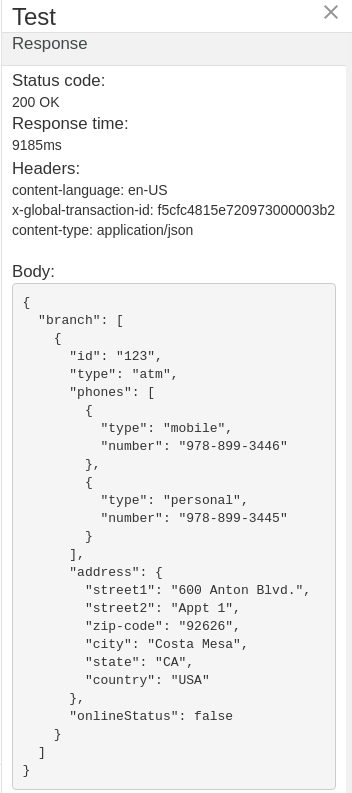
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/designer-soap-rest-assembly.png)

You can see that for each operation, there are two REST operations created a GET and a POST. For each operation, you can see two mapping actions, REST to SOAP (for the request) and SOAP to REST (for the response). You can also see that for each operation there are one or two parse actions. This is a very important concept to understand and also when to use it or not. By default, with the new API Gateway (Native Gateway) messages are streamed and messages are not buffered. This is a very efficient way to serve messages, since the messages can be sent directly to the back-end even if the message was not received completely. This is a very common approaches when dealing with video streaming. Now, when you want to manipulate the message, you cannot do that, let's say you want to remove some pieces especially at the beginning, so you have to get the full message and in order to manipulate you need to get the message and parse it (in other words deserialize it). This is what the parse action does in those flows. Notice that this applies to JSON or XML since you may need to deserialize both types of messages. Of course, parsing a message has a cost in terms of resources and latency and the price will vary a lot depending on the size and complexity of the message.

Looking at the mapping actions, we see that it is a very simple one and you have also the opportunity to make more complex ones. As you can see in the following sample, the response of the XML/SOAP message is simply copied to the response to the JSON response.

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

Let's test it, as usual we use the Test feature within the Assembly panel. We have a similar result than before, except that the response is in JSON (as the request).

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

**Step 9 - Create a Cloudant service**

**This section has not been updated yet because we want to show the new advanced security features included in 2018**

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 Cloudant 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 database : **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 your preferred text editor. we will use these values later.

**Step 10 - Create a LoopBack application**

**This section has not been updated yet because we want to show the new advanced security features included in 2018**

API Connect comes with a developer toolkit. This toolkit provides an offline graphical user interface 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 organize 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-favourite-working-dir>/apic/myfirstproject

$ cd <your-favourite-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 data sources)

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. yes 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 data sources)

1. Change directory to your application directory

cd Customer

**Create a Data source Connector to Cloudant**

The data source 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 data source 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 (e.g.: 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 an 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 Open API (Swagger 2.0) template file by using the '--template template-name' option.

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

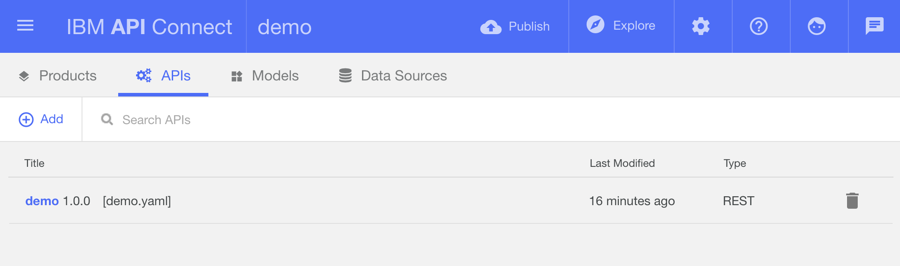
1. Launch API Connect Designer

apic edit

If the designer started correctly, a webpage will automatically open 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:

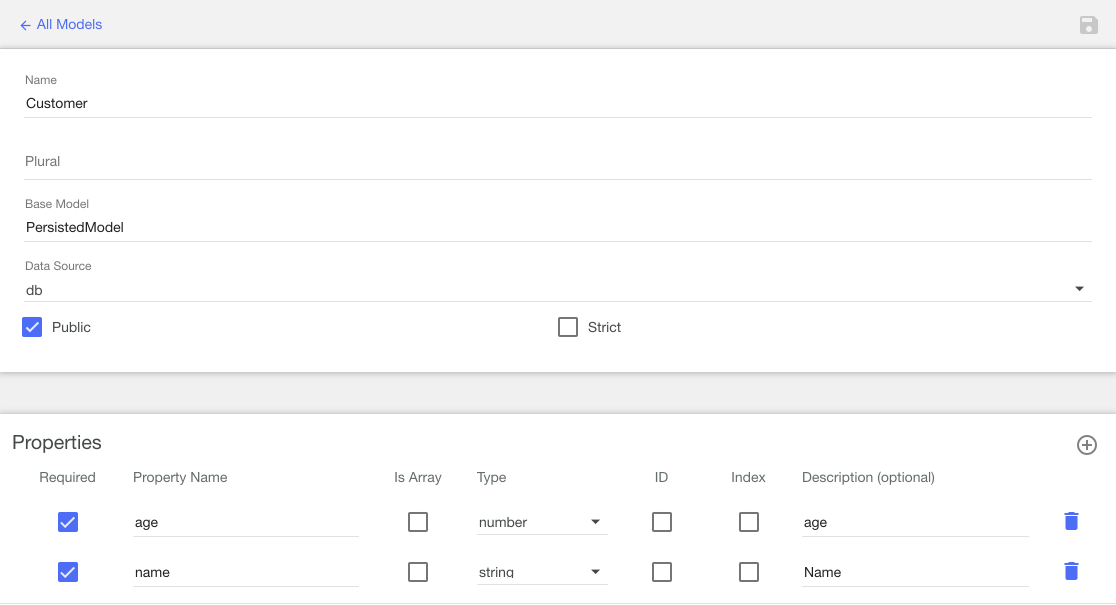
####Create Properties for the Customer Model

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 - Using OAuth to protect your API**

**WARNING: The following section is under construction, but I want to show you what is coming soon**

**Introduction**

OAuth - Open Authorization is a great and modern security mechanism. It is used for two main cases: authentication and authorization. The very nice thing with OAuth is that there is a full control on the life of the token (client side or server side), it is possible to refresh the token, meaning being able to recreate an access token without the need of re-entering the user's credentials, it is possible to perform authorization with the notion of scope, it is possible to authorize a third party to access your data without authenticating (or using your credentials) to this third party, it is possible to revoke the token, a lot of very good things. The only limitation was the content of the token regarding the identity of the parties, this is basically a UUID, but this limitation is corrected with OpenID Connect. One difficulty with OAuth is coming from its flexibility, it is so flexible that it implies a lot of various ways to use OAuth, choices to use different grant types, the way to extract the identity, to perform authentication, to control the revocation and introspection, the way the scope and the consents are handled, the redirection, etc …

In the materials, you also find a POSTMAN collection (alongside the environment definitions). You may have to change the env file in order to have it working for your environment.

In order to perform all the scenarios below, we are going to use the same API that will be versioned, each version will have a different security scheme and a different path /fakemagento/v, for example, /fakemagento/v1.

| **Version** | **Security scheme** | **Referred as** |
| --- | --- | --- |
| V1 | API Key + Basic Authentication |  |
| V2 | Resource Owner Password Credentials Grant | Resource owner |
| V3 | Authorization Code grant + OIDC | Access code |
| V4 | Client Credentials grant | Application |
| V5 | External OAuth Provider |  |
| V6 | Custom JWT Generate and Validate |  |

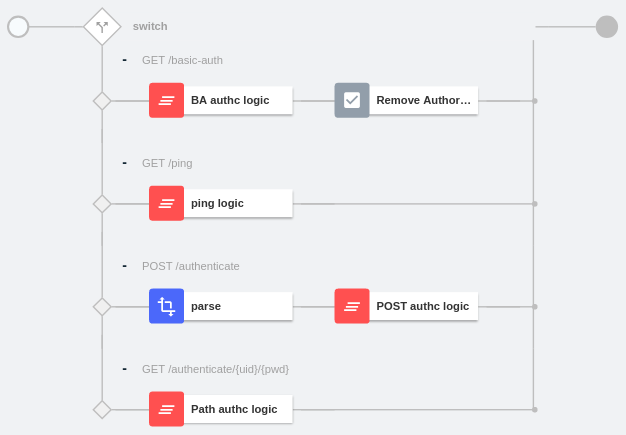
**Preparing the environment - Fake Authentication URL API**

To perform some more advanced scenarii with security, we need a user registry where all the users are defined. There are several types of user registry for user authentication supported in API Connect:

* Authentication URL User Registry - Based on an authentication URL (Following a simple HTTP/S based invocation)
* LDAP User Registry - Based on a LDAP server (Standard LDAP integration)
* Local User Registry - Based on API Connect Local User Registry (Internal registry of the solution)
* OpenID Connect (OIDC) - Configure user authentication using JSON Web Tokens (External OIDC provider)

Because we do not want to spend too much time to install an LDAP server, for simplicity of usage, we create a small API that will perform the role of an Authentication URL User Registry. The principle is very easy, if the password is equal to the uid, the user is authenticated, if not equal then the user in Unauthenticated. **This is for educational purpose only and is of course not secured and should not be used in production environment.**. But this is perfect for educational purpose and it is also a sample of using API Connect with some gateway script samples.

The API provided contains a few more paths (operations) than what we describe here. We only describe the /basic-auth path. Below a screen capture of the API assembly.

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

Below the processing performed in the "BA authc logic" gateway Javascript:   
Line 1: Get the Basic Authorization header and split it based on space   
Line 2: Takes the uid:password base 64 and decode it. Then separate uid and password, separator :.   
Line 7: Create a response Header called *api-authenticated-credential* with the the CN od the user with an hard coded email domain name.   
Line 10: Provide the body of the response following the expected body as defined in the documentation.   
Line 12: If username is different from password then returns UNAUTHENTICATED.

1 var reqauth = context.get('request.headers.authorization').split(' ');

2 var splitval = new Buffer((reqauth[1] || ''), 'base64').toString('utf8').split(':');

3 var username = splitval[0] || '';

4 var password = splitval[1] || '';

5 console.error('>>> User credentials: [' + username + ':' + password + ']');

6 if (username === password ) {

7 context.set('message.headers.api-authenticated-credential', 'cn=' + username + ',email=' + username + '@fr.ibm.com');

8 context.set('message.status.code', 200);

9 context.set('message.headers.content-type', 'application/json');

10 context.message.body.write({username: username, email: username + '@fr.ibm.com', first\_name: username, last\_name: username});

11 } else {

12 context.set('message.status.code', 401);

13 }

Here are some characteristics of this API:

Base path: /fakeauth/v1   
No security, not even an API Key.   
Four paths: /basic-auth (GET), /authenticate (POST), /authenticate/{uid}/{pwd} GET, /ping (GET)   
Two definitions:

* UserCredential object Object containing the credentials in order to perform authentication (uid and password)
* AuthenticatedUser object Object returned when a user is authenticated

Here are the YAML definitions: [Fake Authentication API Open API Document](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/materials/step12/fakeauthenticationurl_1.0.0.yaml) and [Fake Authentication Product Open API Document](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/materials/step12/fakeauthenticationproduct_1.0.0.yaml).

You need to publish the API, let's say in our Integration catalog. I'm going to use the CLI to do that.

apic login -s <manager endpoint> -u <uid> -p <pwd> -r provider/default-idp-2

apic products:publish -c integration -o org1 -s <manager endpoint> --scope catalog fakeauthenticationproduct\_1.0.0.yaml

Sample invocation: curl -k -H "Accept: application/json" -H "Authorization: Basic: Zm9vOmZvbw" https://gw.159.8.70.38.xip.io/org1/integration/fakeauth/v1/basic-auth

Here is a sample response (formatted):

{

"username" : "foo",

"email" : "foo@fr.ibm.com",

"first\_name" : "foo",

"last\_name" : "foo"

}

At this stage, we have configured the Fake Authentication URL API that we will use in the next chapters. Of course, in real life that would be more of a user registry or OIDC provider that should be used.

**Protecting an API with Basic Authentication**

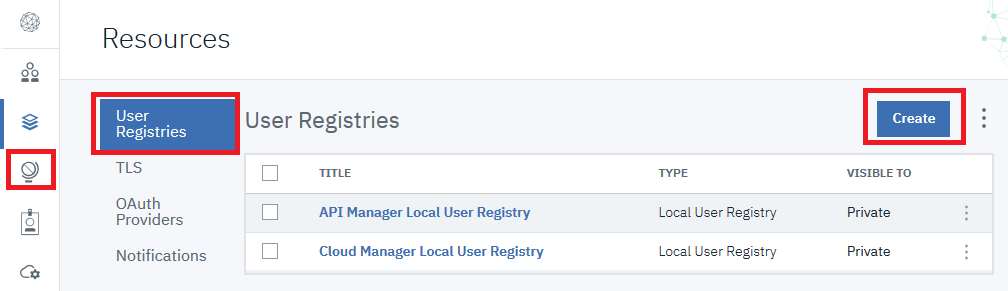
First, please consider that using Basic Authentication is not the best and most secured approach! The reason we have this test, is because it is a simple way to check that the *Fake Authentication URL API* is correctly working and can be used to secure an API. If I may make a parallel with Web application, using Basic Authentication is as secured as using it for a web application. A 401 challenge compared to a Form based authentication will imply that every request will contain the uid/pwd, not very secured indeed.

There is an important design decision regarding what is the scope of the resource we are going to configure. Resource, here, means User Registries, TLS configurations and OAuth Providers. Should they apply and be visible for only one organization, or should it be defined for all organizations. In our case, we have taken the decision that the resources will be defined for all organizations and so we defined them in the Cloud Management Console. We could have decided to do it for each organization so they all would have their specific configurations.

The list of Steps are the following:

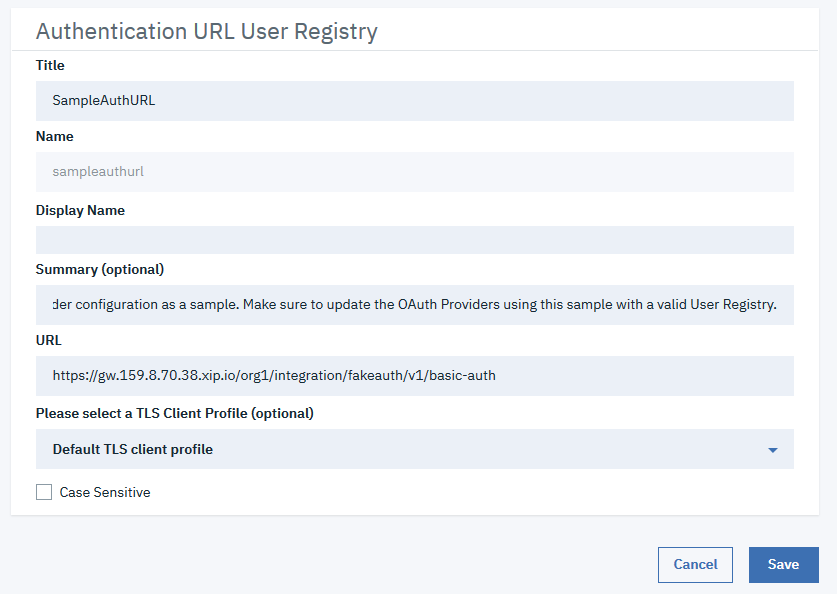
* In the Cloud Management console, define the User Registry based on the *Fake Authentication URL API*
* In the Manager console, associate the user registry with the Catalogs
* Configure the API security to use Basic Authentication and publish it into the catalog
* Test and validate that everything is working

To add the User Registry, go in Cloud Management Console, click on Resources on the navigation panel, click on Create button.

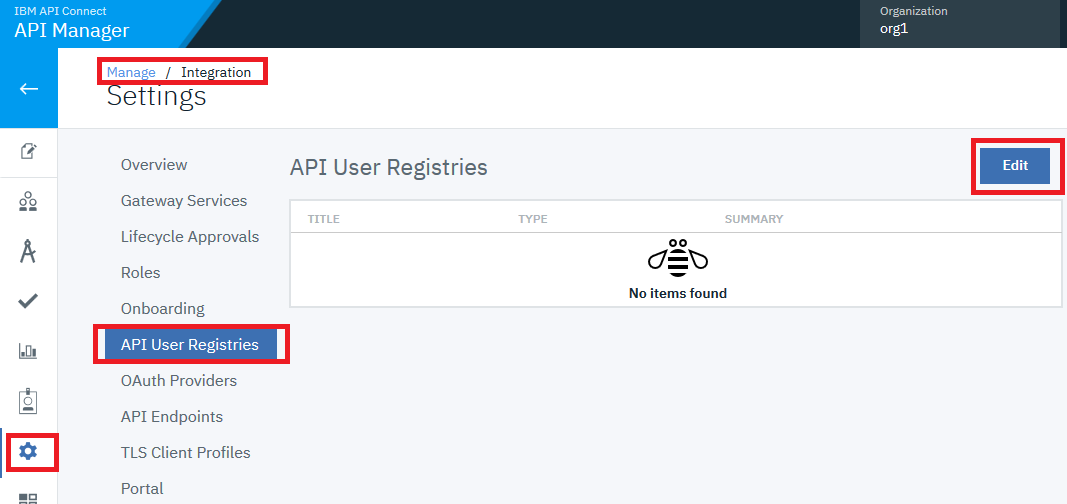
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/cmc-create-user-registry.png)

Select Authentication URL User Registry, enter the following Information:

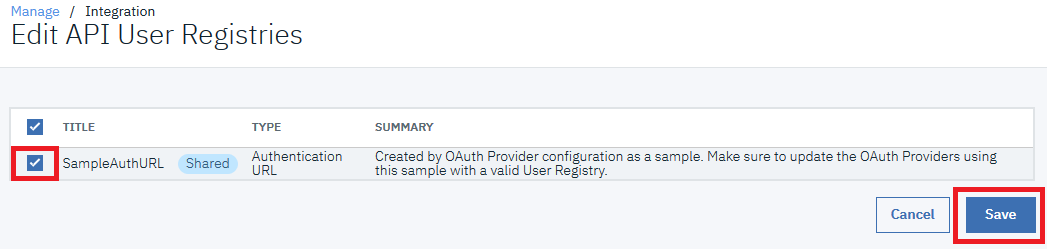
Title: SampleAuthURL   
Summary: Created by OAuth Provider configuration as a sample. Make sure to update the OAuth Providers using this sample with a valid User Registry.   
URL: <https://gw.159.8.70.38.xip.io/org1/integration/fakeauth/v1/basic-auth>   
TLS Client Profile: Select Default TLS Client Profile

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

To associate the user registry with the Catalog, go in the Manager console, click on Manage, select the Integration catalog, then on Settings and API User Registries

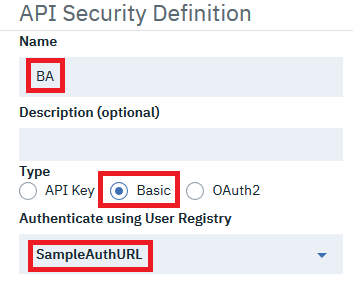
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/manager-user-registry-edit.png)

Click on the checkbox for the SampleAuthURL and click Save button.

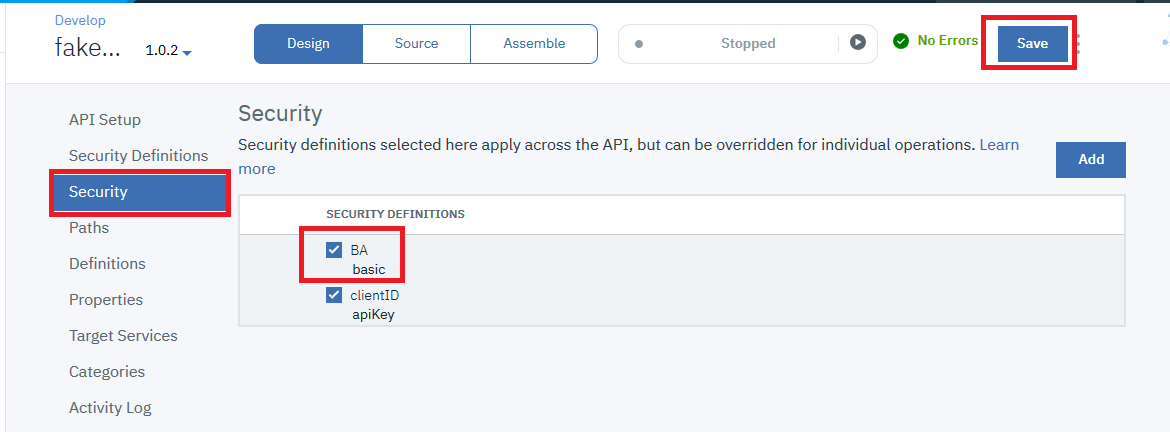
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/manager-user-registry-select.png)

Now, let's take configure the security for the API. In the Manager, click on Develop menu, import the fakemagento API. This is the initial API to work with an is available in the materials folder. It is version 1.0. Now, we can edit the Security Definitions section and Add the Basic Authentication, click on Add. Enter

Name: BA   
Type: Select Basic   
Select SampleAuthURL for the *Authenticate using User Registry*

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

In the Security section, select BA.

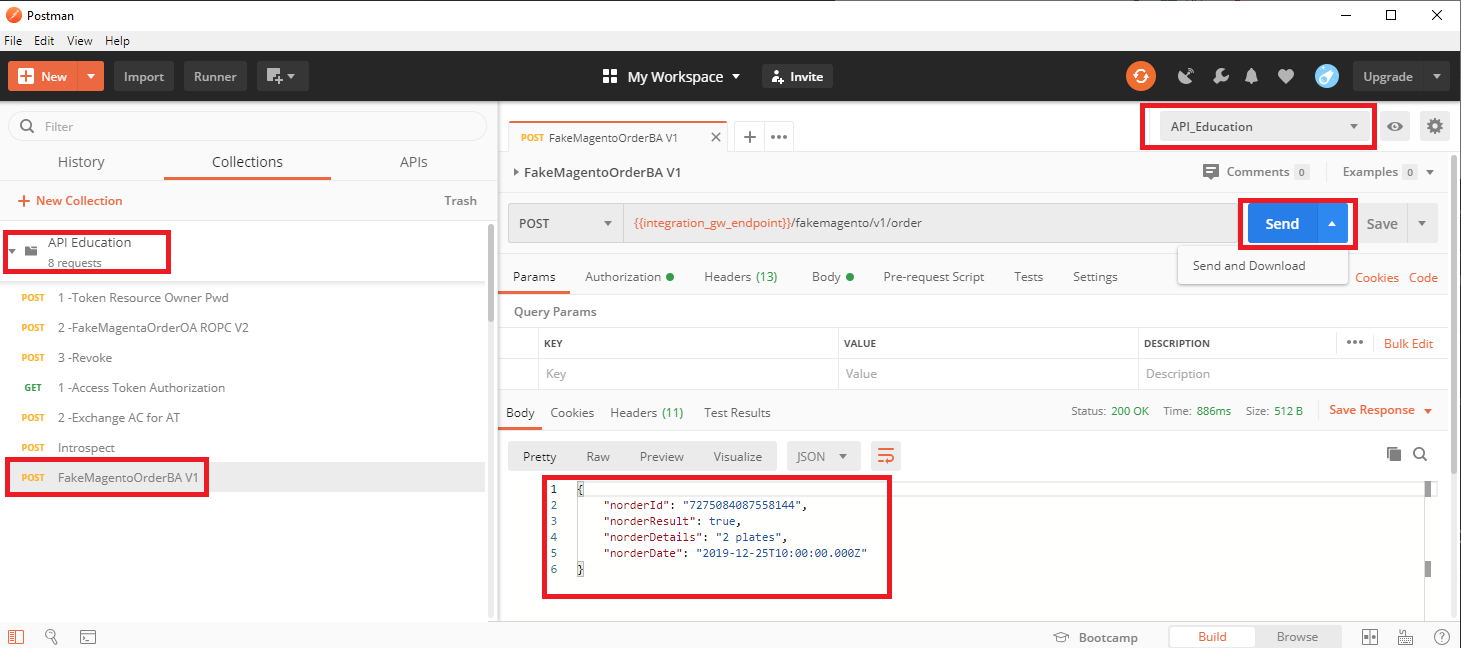
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/manager-BA-Security-section.png)

Create a product and publish it to the Integration Catalog, then subscribe to it. (Not explained here, see previous steps).

We can now test the API. The credentials are located in the Authorization header. curl -k -H "Content-Type: application/json" -H "Accept: application/json" -H "X-IBM-Client-Id: 421223e773f237c5231842102660896e" -H "Authorization: Basic Zm9vOmZvbw==" -d "{ "orderDetails": "2 plates", "orderDate": "2019-12-25T10:00:00.000Z"}" "<https://gw.159.8.70.38.xip.io/org1/integration/fakemagento/v1/order>" You should obtain a response:

{"norderId":"7275084087558144","norderResult":true,"norderDetails":"2 plates","norderDate":"2019-12-25T10:00:00.000Z"}

Here, I'm introducing the POSTMAN collection. In the POSTMAN, there is a number of requests that you can adapt for your environment. The collection is called: API Education. The environment that you need to adjust for your environment is called: API\_Education.

To test the API with Basic Authentication security. You can use the *FakeMagentoOrderBA V1* API. [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/postman-ba-test.png)

Clicking on Send button will give you the expected result.

Below the equivalent with curl (on Windows):

curl -k "https://gw.159.8.70.38.xip.io/org1/integration/fakemagento/v1/order" -H "Content-Type: application/json" -H "Accept: application/json" -H "X-IBM-Client-Id: 421223e773f237c5231842102660896e" -H "Authorization: Basic Zm9vOmZvbw==" -H "Accept-Encoding: application/json" -d "{ \"orderDetails\": \"2 plates\", \"orderDate\": \"2019-12-25T10:00:00.000Z\" }"

returns

{"norderId":"7275084087558144","norderResult":true,"norderDetails":"2 plates","norderDate":"2019-12-25T10:00:00.000Z"}

**Protecting an API with OAuth - Resource Owner Password Credentials grant**

The first test was not based on OAuth, all the following one are. We will use two OAuth providers only. One for native support, API Connect acting as an OAuth/OIDC provider and one for third party integration, using IBM AppId, one of IBM's cloud solution for OAuth/OIDC support. The Resource Owner Password Credentials grant type is specified in [RFC 6749 - OAuth 2.0 Authorization Framework](https://tools.ietf.org/html/rfc6749), likewise the other grants used in this article.

In this lab, we start with a very simple case, but still very useful: the use of the Password flow which really is the Resource Owner Password Credentials grant type in OAuth terminology. It is easy because it is 2-legged, for simplicity we also use Basic Authorization to extract identity, the user will be authenticated against the User Registry. We use API Connect as the OAuth provider, notice that it is also possible to use API Connect with an external OAuth provider, we will see it later. This grant type is used when there is a high trust between the client and the application. Some people may find it less secure than using the Authorization grant, but it does not mean that it cannot be used when the conditions of trust are met. The client authenticates directly to the native OAuth provider specified in API Connect with its uid/pwd along the application credentials (client-id/cliend-secret) and gets an Access Token. Then the application accesses API Connect.

**Note**: In this lab, we do not explain how to propagate the user information with a JWT token, it will be done in another version of this lab. But this is an important question and there are different ways to get user information like having the back-end performing a callback with the OAuth token to get information. We are not explaining the generation of OIDC token. As of today, this is not a mandatory scenario required by OpenID and we are still in discussions to decide to implement it.

**Create the OAuth Provider**

So let's start! First let's create the native OAuth provider. As discussed earlier, we use the Cloud Management console, but it was a design decision and we could use the API Manager console and manage OAuth resources, if we wanted.

Login to the Cloud Manager console and select Resources (or Manage Resources from home). Select OAuth Providers and click on Add and select Native OAuth Provider button.

Enter

Title: NativeProvider   
Description: Internal OAuth Provider shared among organisations   
Base path: /nativeprovider   
Select DataPower API Gateway

Click Next button

Select the grant types you want to support, in our case we are going to use : Application, Access code and Resource owner - Password. We also select the Public client type, as well as Confidential by default selected. We keep the default values for the endpoints.

Click Next button

Specify the scopes you want to support, in our case we will use only one scope *details*. To provide detailed access.

Click Next button

The following settings are very important and there are many ways to configure OAuth, API Connect is very flexible and extensible on this part. The processing has been described in the specific steps happening during the token(s) creation procedure. It is similar in the way the AAA framework in DataPower is separating clearly each step. I do not want to explain all the combinations below, but provide the different values for each step.

| **Identity Extraction** | **Authentication** | **Authorization** |
| --- | --- | --- |
| Basic Authentication | User Registry (LDAP) | Authenticated |
| Default HTML Form | Authentication URL | Default HTML Form |
| Custom HTML Form | Disabled | Custom HTML Form |
| Context variable |  | Disabled |
| Redirect |  |  |

There are many possible combinations, but you can have even further possibilities to add more processing as we will see later, especially in how the consents are managed, meta data are added, scopes are managed, etc ... In our case, it is simple and we are keeping all the default values, so nothing to do!

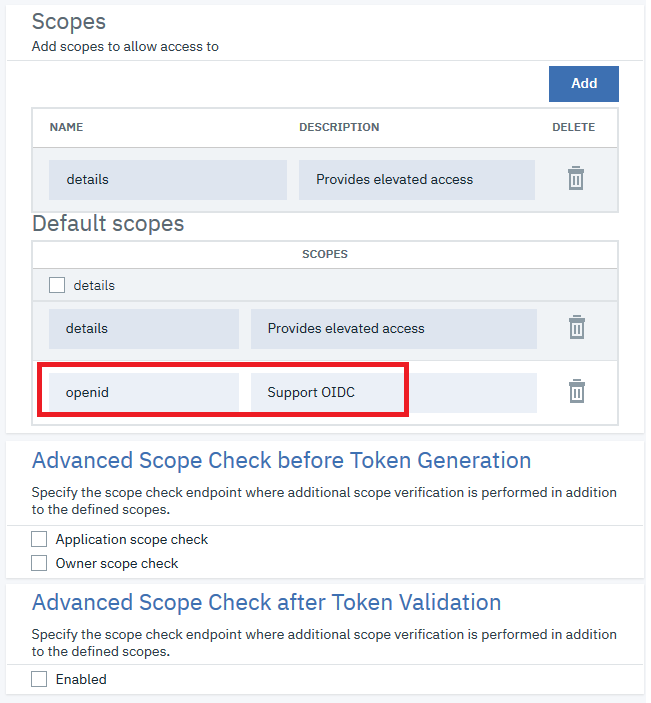
Click Next button

You get a Summary screen.

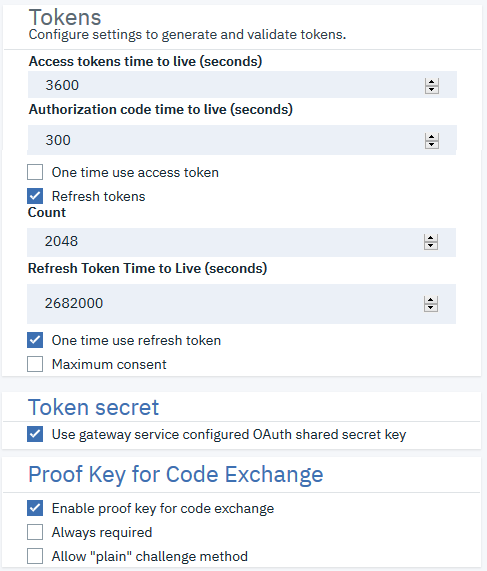
Click on Finish button.

We are going to add a few features to the OAuth Provider, such as OIDC and use of other endpoints such as introspection. Edit the NativeProvider:

* In the Scopes panel, add openid as a scope for the support of OIDC.

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

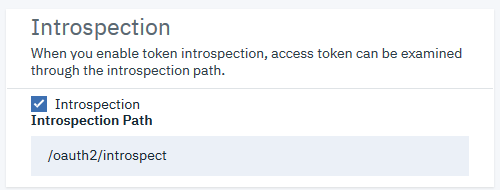
* In the Tokens panel, click on the Refresh tokens checkbox and click Save button

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

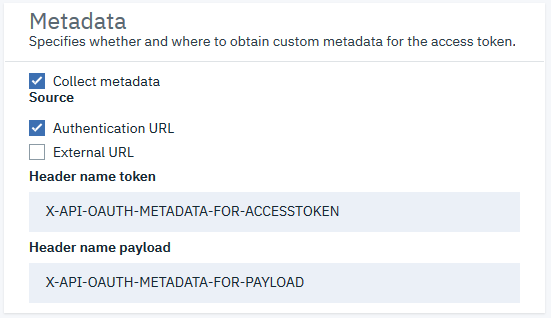
* In the Token Management panel, click on the Token Management checkbox, Resource owner revocation path checkbox and Client revocation path checkbox and then click Save button. It will ask you if you want to update the Assembly, keep yes and click on the Confirm button.

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

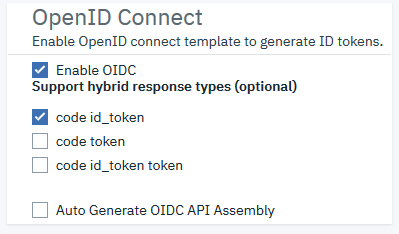
* In the Introspection panel, click on the Introspection checkbox and then click on the Save button

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

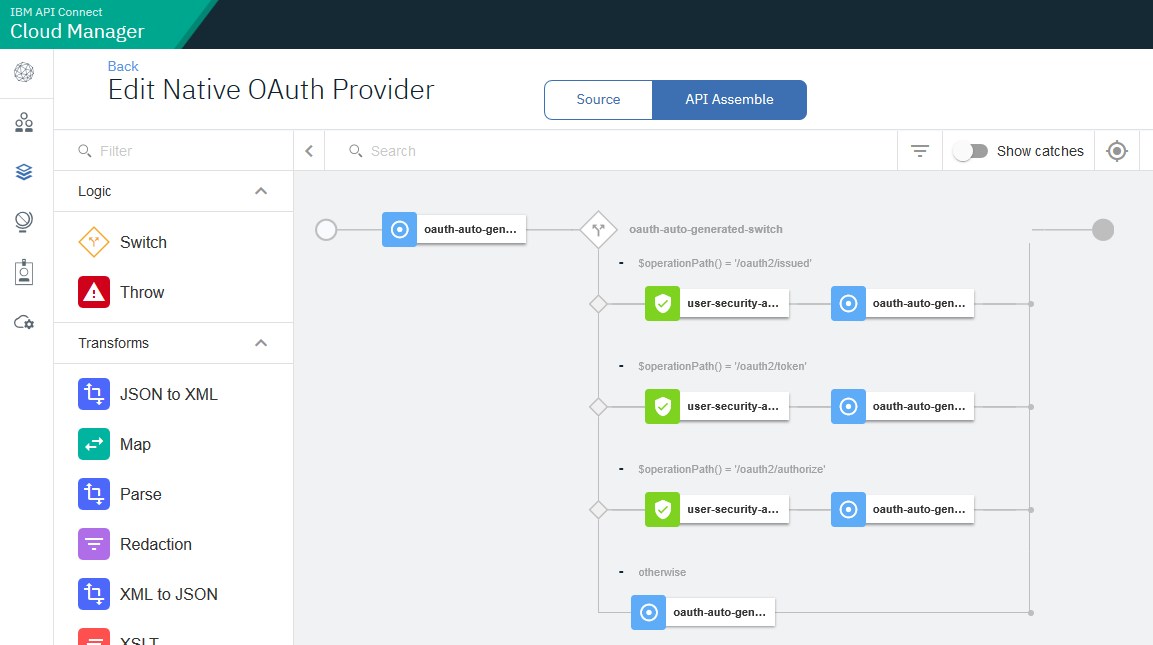
* Look at the Metadata Panel, you will see other settings to collect metadata.

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

* In the OpenID Connect panel, click on the Enable OIDC checkbox and keep the other values, then click on the Save button.

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

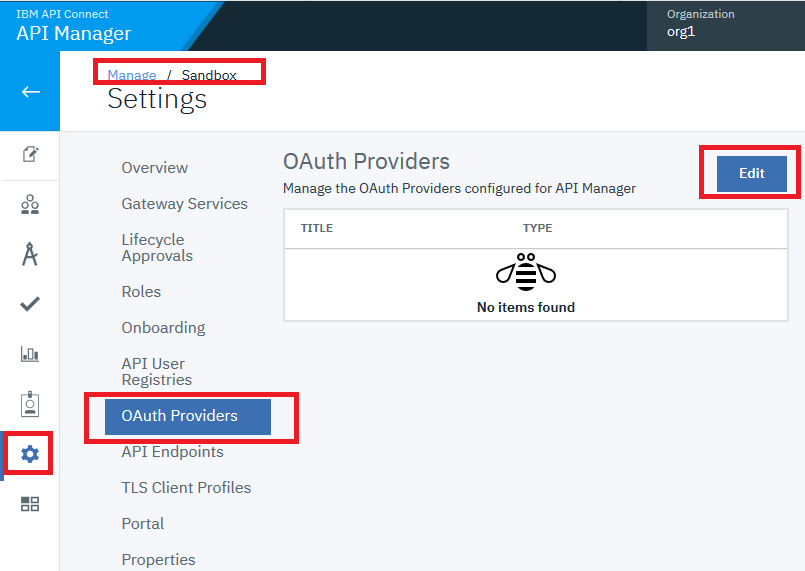
* Look at the API Editor Panel, this is where the code is implemented based on the settings used to configure the OAuth provider. This allows to extend even further more how you want to manage your token, given the possibility to use your own code (Gateway script or XSLT). Click on Back and Save button.

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

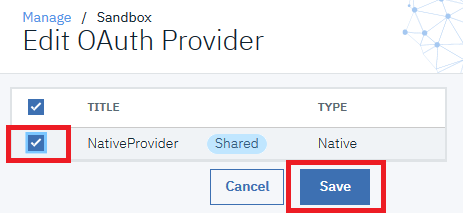
We have an OAuth provider definition.

**Make the OAuth provider usable in the catalogs**

We need to make this OAuth Provider accessible in the various catalogs where we want to use it. We leave the Cloud Manager console and go to the Manager Console. Click on Manage and select Sandbox, then Settings and OAuth Providers. Click on Edit button on the top left.

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

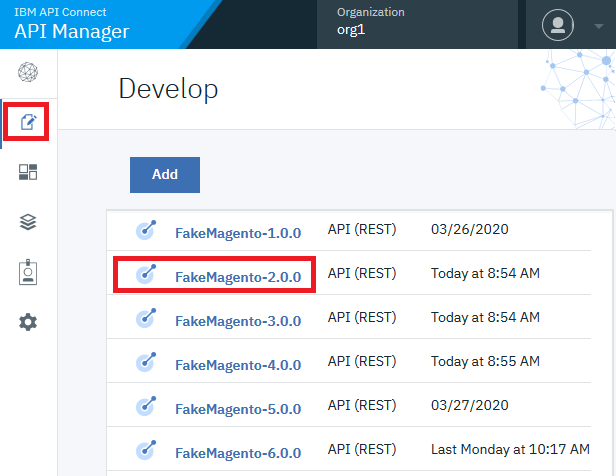
Click on the checkbox close to the NativeProvider

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

Repeat the same operation with the Integration Catalog. It is not yet accessible because we are not using it in any API.

**Protect the API with OAuth**

Now let's protect, the FakeMagento version 2.0.0 API. Click on Develop and select the FakeMagento-2.0.0 API.

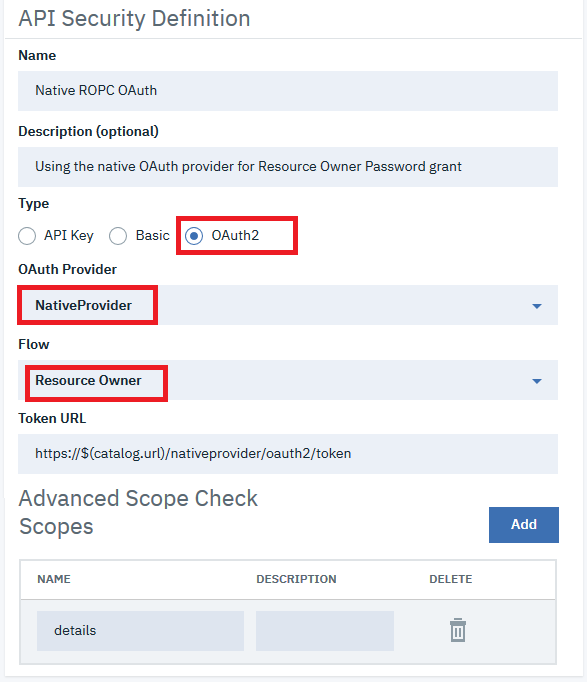
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/ropc-secure-API.png)

Click on Security Definitions and click on Add button.

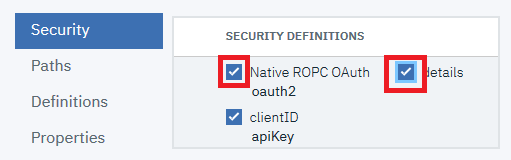
enter:

Name: Native ROPC OAuth   
Description: Using the native OAuth provider for Resource Owner Password grant   
Select OAuth2   
Select NativeProvider for the OAuth Provider   
Select Resource Owner for the Flow

Click Save button.

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

In the Security selection, select Native ROPC OAuth and the details scope.

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

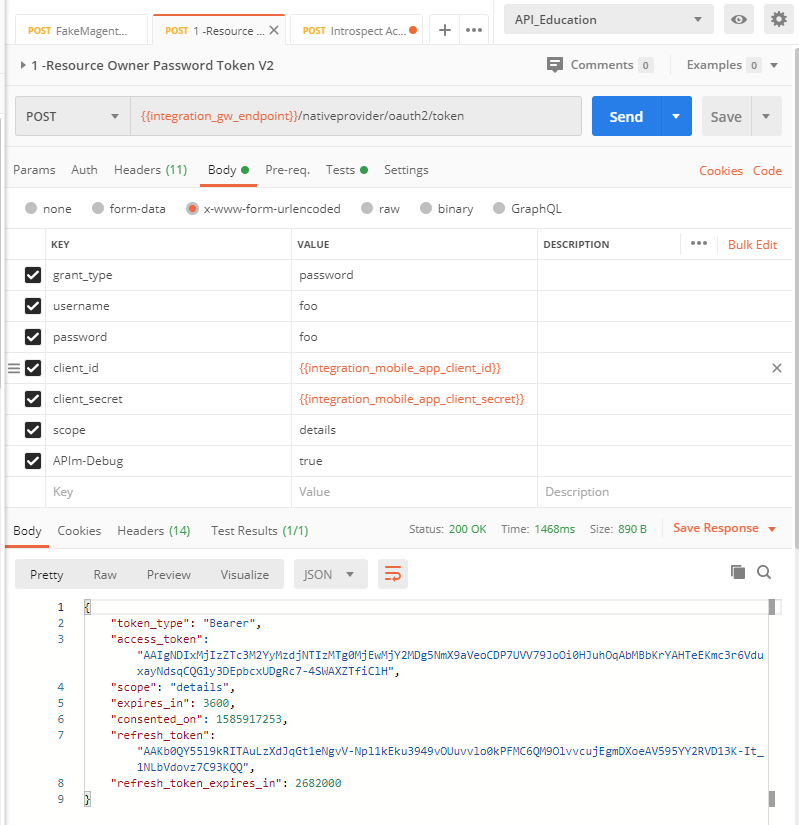
**Tests**

Not explained in detail here, but we publish the product (or use versioning with the publish capability), we are using the Integration environment. Then we subscribe to the Product with the Gold Plan and approve the subscription. The API is published and ready to use.

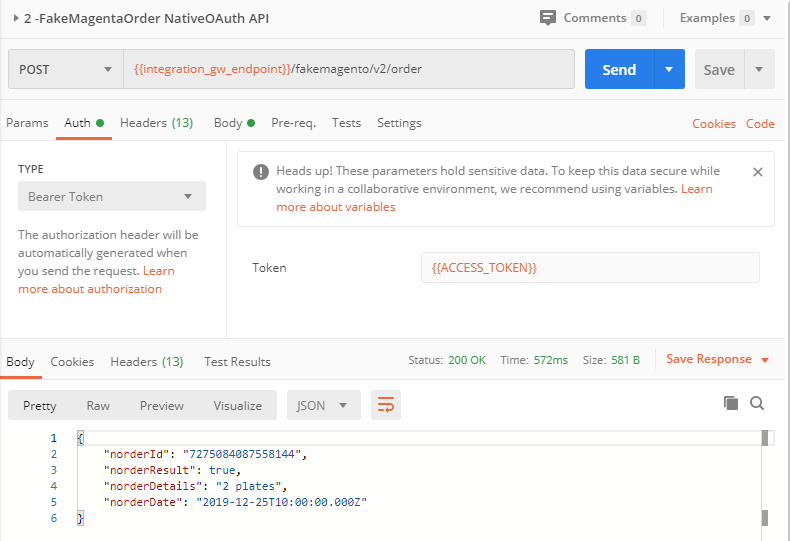
I'm going to test it in different ways: using Postman, curl and using the developer portal.

**Using POSTMAN**

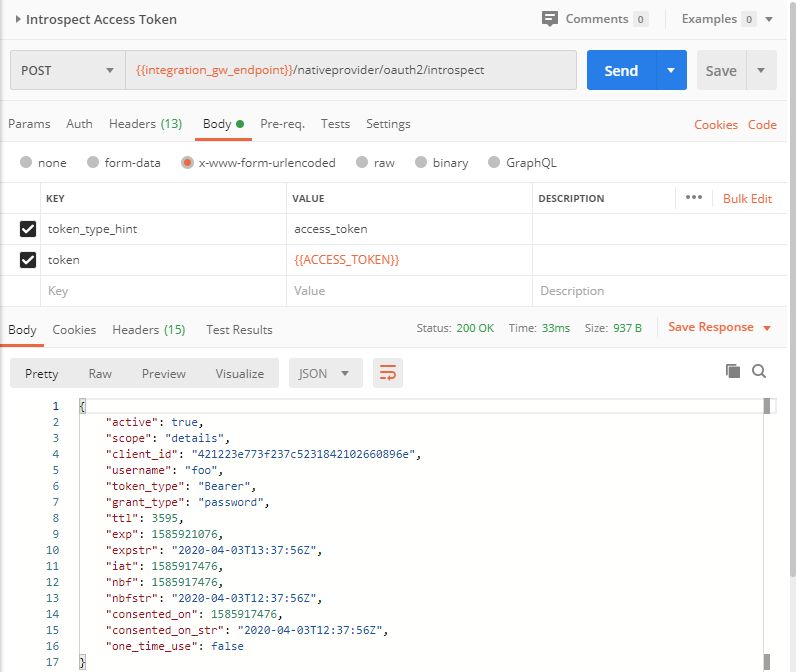
1. Get token using the "1 -Resource Owner Password Token V2" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-11.png)

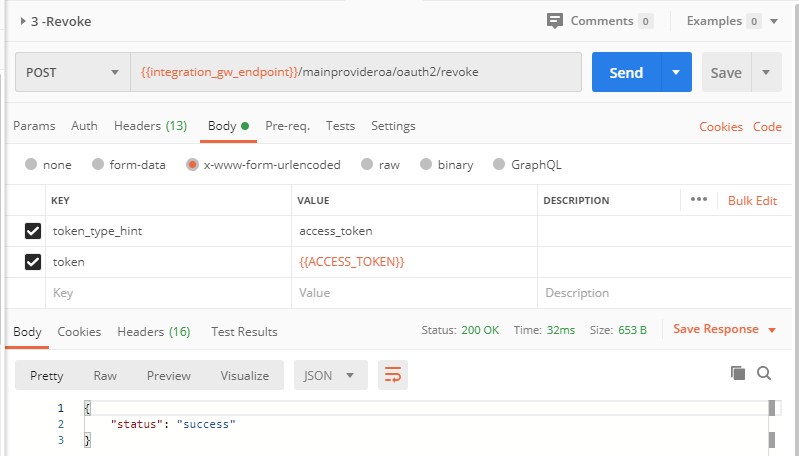
1. Use token to access FakeMagento V2.0.0 API using the "2 -FakeMagentaOrder NativeOAuth API" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-12.png)

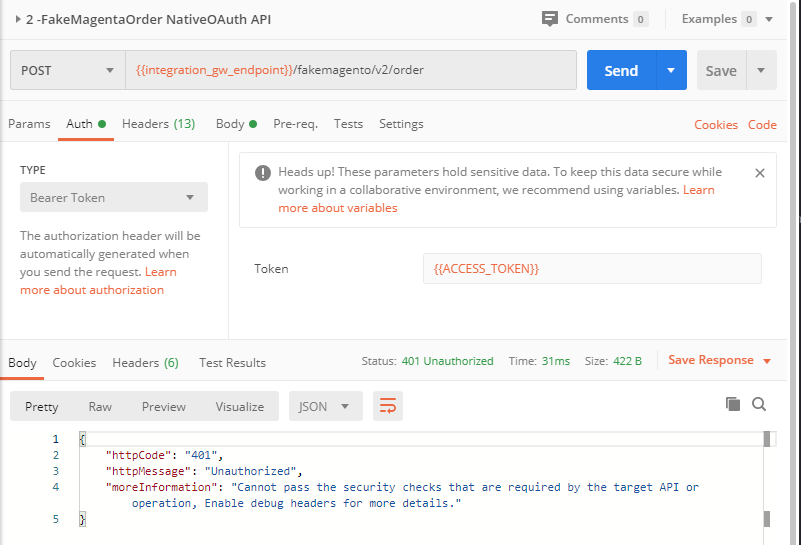
1. Introspect token API using the "Introspect Access Token" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-13.png)

1. Revoke token API using the "3 -Revoke" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-14.png)

1. Access API again with revoked token API using the "2 -FakeMagentaOrder NativeOAuth API" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-15.png)

**Using curl**

1. Get token Request

POST /org1/integration/nativeprovider/oauth2/token HTTP/1.1

Content-Type: application/x-www-form-urlencoded

accept: application/json

grant\_type=password&username=foo&password=foo&client\_id=421223e773f237c5231842102660896e&client\_secret=556a75ce26097f96ea281ed47c1cf2e7&scope=details&APIm-Debug=true

Response

HTTP/1.1 200 OK

Transfer-Encoding: chunked

X-RateLimit-Limit: name=default,100;

X-RateLimit-Remaining: name=default,92;

accept: application/json

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e87307000066131

Content-Type: application/json

Pragma: no-cache

Date: Fri, 03 Apr 2020 12:47:45 GMT

{

"token\_type": "Bearer",

"access\_token": "AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmW513MzVpeO\_t4EViZ\_M9Nb\_xWVKN0qah8cQUsosmbkMbVbfxwTMBUtuhSMXs-5MzT4MxG9eqdzODLHzfv00CP4",

"scope": "details",

"expires\_in": 3600,

"consented\_on": 1585918065,

"refresh\_token": "AAIbeP1VikS6hIMuLClKVQF1LxQAhZqlcY5TFZ6wC4MBJ7xwcE7kBQ3Dp7v\_SskGuSQmIHSZKBvEAuZ61sfPAU\_28L11EzZpd4zzl4l9LNRefg",

"refresh\_token\_expires\_in": 2682000

}

1. Use token to access FakeMagento V2.0.0 API Request

POST /org1/integration/fakemagento/v2/order HTTP/1.1

X-IBM-Client-Id: 421223e773f237c5231842102660896e

Content-Type: application/json

Accept: application/json

Authorization: Bearer AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmW513MzVpeO\_t4EViZ\_M9Nb\_xWVKN0qah8cQUsosmbkMbVbfxwTMBUtuhSMXs-5MzT4MxG9eqdzODLHzfv00CP4

{

"orderDetails": "2 plates",

"orderDate": "2019-12-25T10:00:00.000Z"

}

Response

HTTP/1.1 200 OK

Transfer-Encoding: chunked

X-RateLimit-Limit: name=ten,10;

X-RateLimit-Remaining: name=ten,9;

Accept: application/json

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e873073000805e9

Content-Type: application/json

Date: Fri, 03 Apr 2020 12:47:47 GMT

{

"norderId": "7275084087558144",

"norderResult": true,

"norderDetails": "2 plates",

"norderDate": "2019-12-25T10:00:00.000Z"

}

1. Introspect token API Request

POST /org1/integration/nativeprovider/oauth2/introspect HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Accept: application/json

x-ibm-client-id: 421223e773f237c5231842102660896e

x-ibm-client-secret: 556a75ce26097f96ea281ed47c1cf2e7

token\_type\_hint=access\_token&token=AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmW513MzVpeO\_t4EViZ\_M9Nb\_xWVKN0qah8cQUsosmbkMbVbfxwTMBUtuhSMXs-5MzT4MxG9eqdzODLHzfv00CP4

Response

HTTP/1.1 200 OK

Transfer-Encoding: chunked

X-RateLimit-Limit: name=default,100;

X-RateLimit-Remaining: name=default,90;

Accept: application/json

x-ibm-client-id: 421223e773f237c5231842102660896e

x-ibm-client-secret: 556a75ce26097f96ea281ed47c1cf2e7

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e873077000805f9

Content-Type: application/json

Date: Fri, 03 Apr 2020 12:47:51 GMT

{

"active": true,

"scope": "details",

"client\_id": "421223e773f237c5231842102660896e",

"username": "foo",

"token\_type": "Bearer",

"grant\_type": "password",

"ttl": 3594,

"exp": 1585921665,

"expstr": "2020-04-03T13:47:45Z",

"iat": 1585918065,

"nbf": 1585918065,

"nbfstr": "2020-04-03T12:47:45Z",

"consented\_on": 1585918065,

"consented\_on\_str": "2020-04-03T12:47:45Z",

"one\_time\_use": false

}

1. Revoke token API Request

POST /org1/integration/mainprovideroa/oauth2/revoke HTTP/1.1

Content-Type: application/x-www-form-urlencoded

Accept: application/json

x-ibm-client-id: 421223e773f237c5231842102660896e

x-ibm-client-secret: 556a75ce26097f96ea281ed47c1cf2e7

token\_type\_hint=access\_token&token=AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmW513MzVpeO\_t4EViZ\_M9Nb\_xWVKN0qah8cQUsosmbkMbVbfxwTMBUtuhSMXs-5MzT4MxG9eqdzODLHzfv00CP4

Response

HTTP/1.1 200 OK

Transfer-Encoding: chunked

X-RateLimit-Limit: name=default,100;

X-RateLimit-Remaining: name=default,89;

Accept: application/json

x-ibm-client-id: 421223e773f237c5231842102660896e

x-ibm-client-secret: 556a75ce26097f96ea281ed47c1cf2e7

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e87307b000661a1

Content-Type: application/json

Pragma: no-cache

Date: Fri, 03 Apr 2020 12:47:55 GMT

{"status":"success"}

1. Access API again with revoked token API Request

POST /org1/integration/fakemagento/v2/order HTTP/1.1

X-IBM-Client-Id: 421223e773f237c5231842102660896e

Content-Type: application/json

Accept: application/json

Authorization: Bearer AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmW513MzVpeO\_t4EViZ\_M9Nb\_xWVKN0qah8cQUsosmbkMbVbfxwTMBUtuhSMXs-5MzT4MxG9eqdzODLHzfv00CP4

{

"orderDetails": "2 plates",

"orderDate": "2019-12-25T10:00:00.000Z"

}

Response

HTTP/1.1 401 Unauthorized

Content-Type: application/json

X-RateLimit-Limit: name=ten,10;

X-RateLimit-Remaining: name=ten,8;

WWW-Authenticate: Bearer error='invalid\_token'

Date: Fri, 03 Apr 2020 12:48:01 GMT

{

"httpCode": "401",

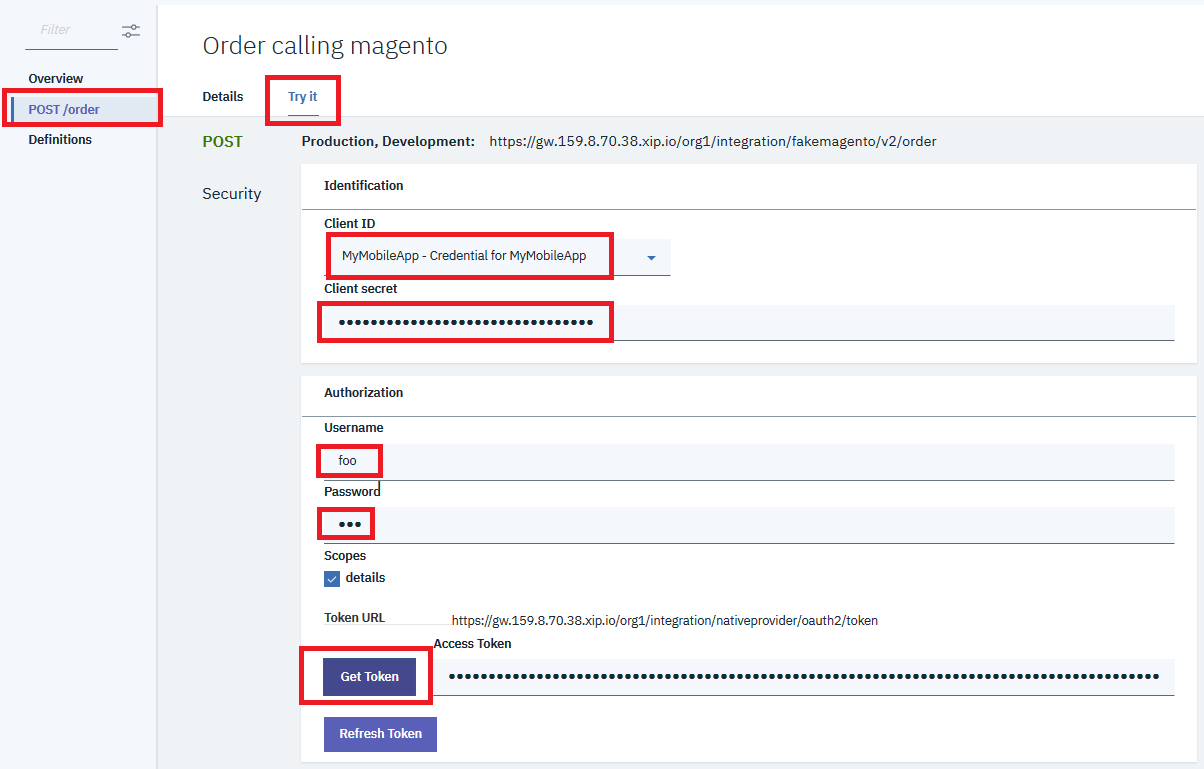
"httpMessage": "Unauthorized",

"moreInformation": "Cannot pass the security checks that are required by the target API or operation, Enable debug headers for more details."

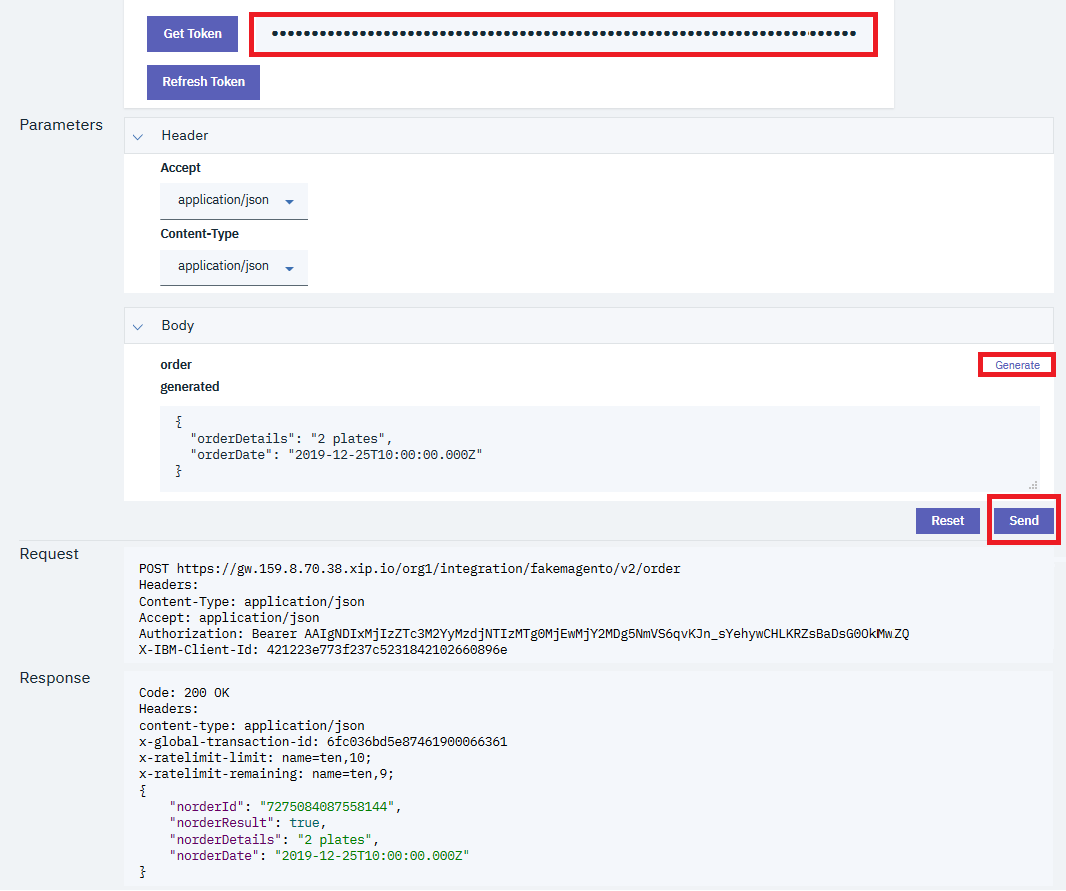
}

**Using the Developer portal**

We do not explain all the steps because it has been done in previous chapters. We sign into the developer Portal, then select the FakeMagento V2.0.0 API and click on Try it. In the first step, we get the token. We select the MyMobileApp application, then enter the client\_secret, enter the username and password and click on the Get Token button. It returns the Access Token.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-21.png)

In the second step, we click on Generate link to automatically populate the parameters needed to call the API. Then click on Send button. This invokes the FakeMagento API using the access Token as a Bearer.

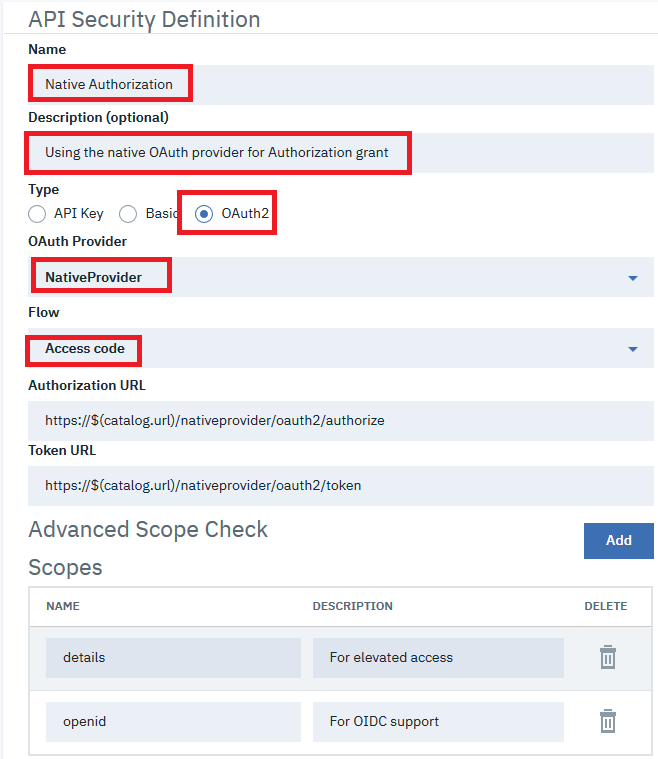
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-ropc-22.png)

**Protecting an API with OAuth - Authorization Code grant and OIDC**

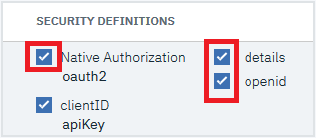
**Protect the API with OAuth - Authorization**

Let's protect, the FakeMagento version 3.0.0 API. Click on Develop and select the FakeMagento-3.0.0 API. Click on Security Definitions and click on Add button. Enter:

Name: Native Authorization   
Description: Using the native OAuth provider for Authorization grant   
Select OAuth2   
Select NativeProvider for the OAuth Provider   
Select Access code for the Flow   
Select details and openid for the scopes

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

Click Save button. In the Security section, select Native Authorization and the two scope details and openid, then click Save button.

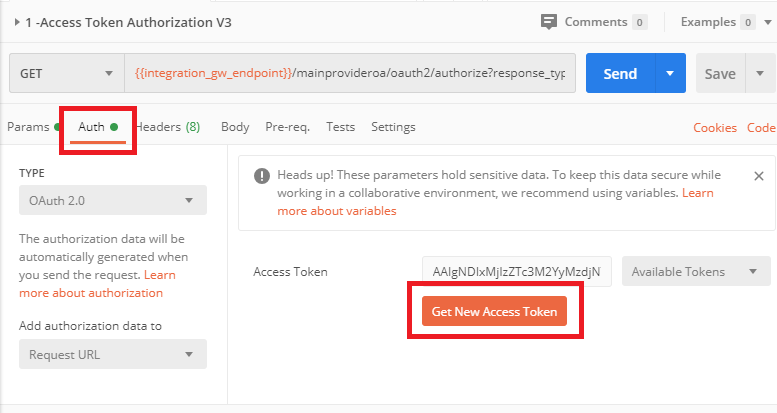
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/access-secure-API-sec.png)

**Tests**

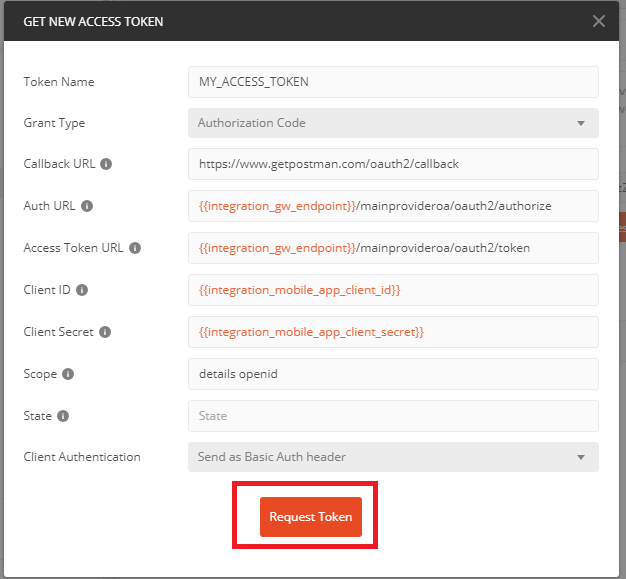
Not explained in detail here, but we publish the product (or use versioning with the publish capability), we are using the Integration environment. Then we subscribe to the Product with the Gold Plan and approve the subscription. The API is published and ready to use.

**Using POSTMAN**

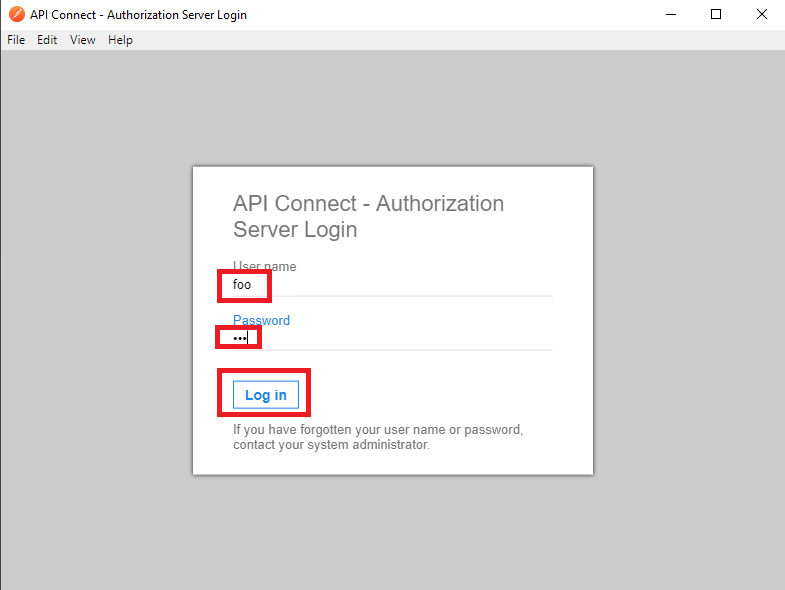
We are using one nice feature of Postman to directly get the access token. It hides a little bit the use of the Access Code. I have provided the request in Postman to exchange the Access Code for the Access Token. Using the "1 -Access Token Authorization V3" request Click on the Authorization link, then click on Get New Access Token

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-11.png)

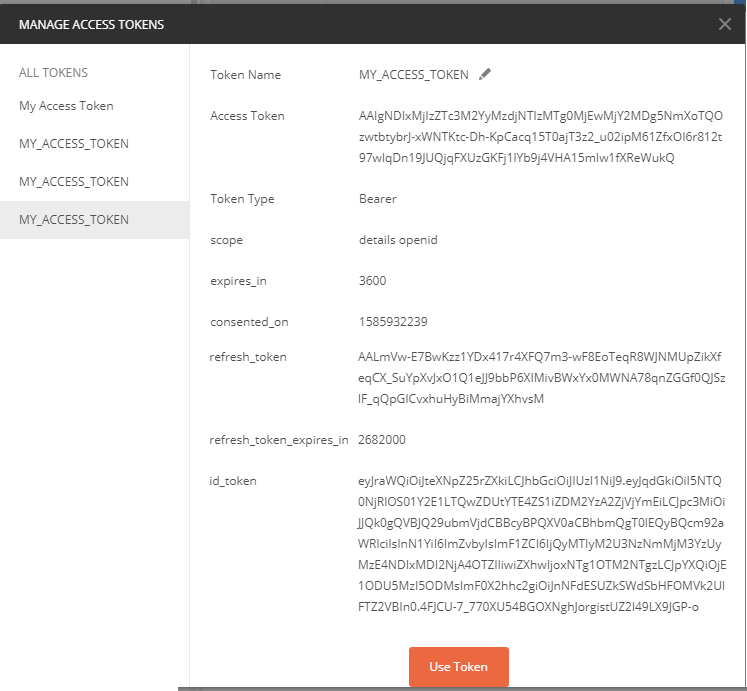
Then click on Request Token button

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-12.png)

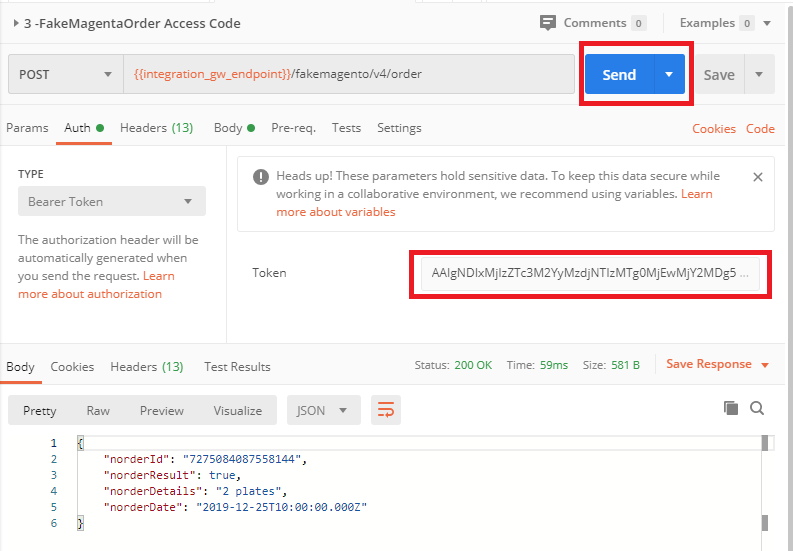
Enter the credentials for the user. Click Login button. Notice that this page is generated by API Connect, this is the default page for authentication.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-13.png)

You get the Access Token and the idtoken. Copy the Access Token.

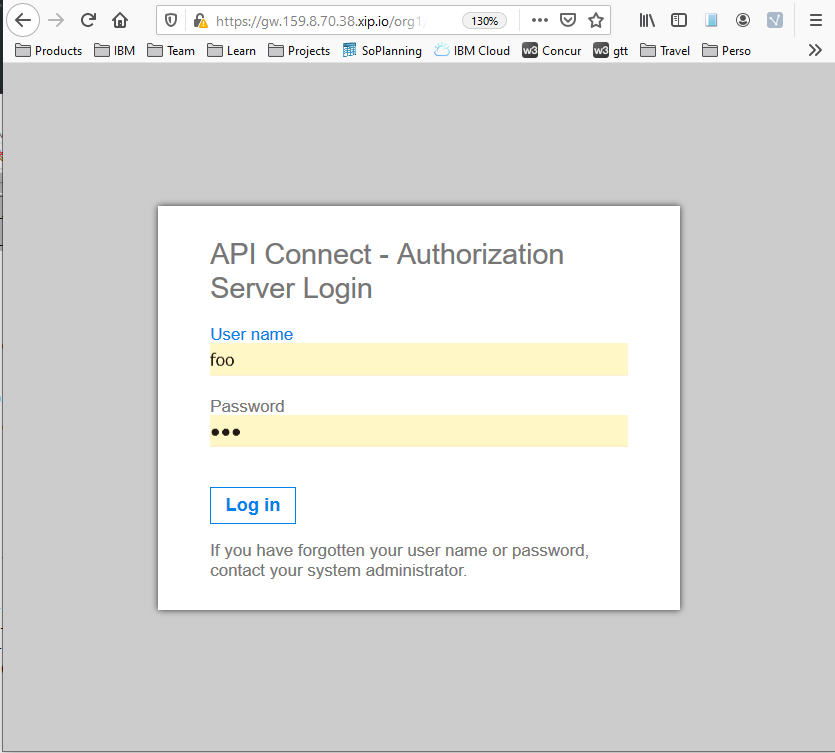
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-14.png)

Use the Access Token to access the FakeMagento V3 API.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-15.png)

**Using curl**

1. Get the Access Code Use a browser and enter <https://gw.159.8.70.38.xip.io/org1/integration/mainprovideroa/oauth2/authorize?response_type=code&redirect_uri=https://www.getpostman.com/oauth2/callback&client_id=421223e773f237c5231842102660896e&scope=details%20openid>   
   You get the login pages.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-31.png)

1. Enter the user credentials and click on Login button. In the Browser URL, you get the Access Code

https://app.getpostman.com/oauth2/callback?code=AAIANrOjEOMYqicDr7MkV3khttTEMasEphtiBZz3ieiYXj2qmFToG6mH6MtUvmnceAlbNhFtlHEBsgSrV8Z3otBRXMOXuGg7V-F\_DCy13V1dbg

1. Send the Access Code to get the Access Token Request

POST /org1/integration/mainprovideroa/oauth2/token HTTP/1.1

Accept: application/json

Postman-Token: cadf4de6-713a-4741-ae42-e49c2a6552ff

Content-Type: application/x-www-form-urlencoded

grant\_type=authorization\_code&client\_id=421223e773f237c5231842102660896e&client\_secret=556a75ce26097f96ea281ed47c1cf2e7&code=AAIANrOjEOMYqicDr7MkV3khttTEMasEphtiBZz3ieiYXj2qmFToG6mH6MtUvmnceAlbNhFtlHEBsgSrV8Z3otBRXMOXuGg7V-F\_DCy13V1dbg&redirect\_uri=https%3A%2F%2Fwww.getpostman.com%2Foauth2%2Fcallback&scope=details%20openid

Response

HTTP/1.1 200 OK

X-RateLimit-Limit: name=default,100;

X-RateLimit-Remaining: name=default,81;

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e876d7200066931

Content-Type: application/json

{

"token\_type": "Bearer",

"access\_token": "AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmVqjyeCu56Y1cVo\_QEk0y4MR4WNBR8Zmq3RKEWsp4J89lQ5qyEYWzTaFN9mYWNJt26o\_NIYWWElfl3OMliuc5hDgDhEVV6LOYlKnaBvuHJH5w",

"scope": "details openid",

"expires\_in": 3600,

"consented\_on": 1585933682,

"refresh\_token": "AAIZ\_fOxRAPcGz8CHCEDTfcGoNMICMj8I-NpcMtiHmeBbtmOsDWNPGD8LhlRq3t6ESqleJgcVogHd7oaDohU0tRBLpW\_r-bE6H1B1iXhftWi6FTeHGvtfq2nINWhcRUHn4s",

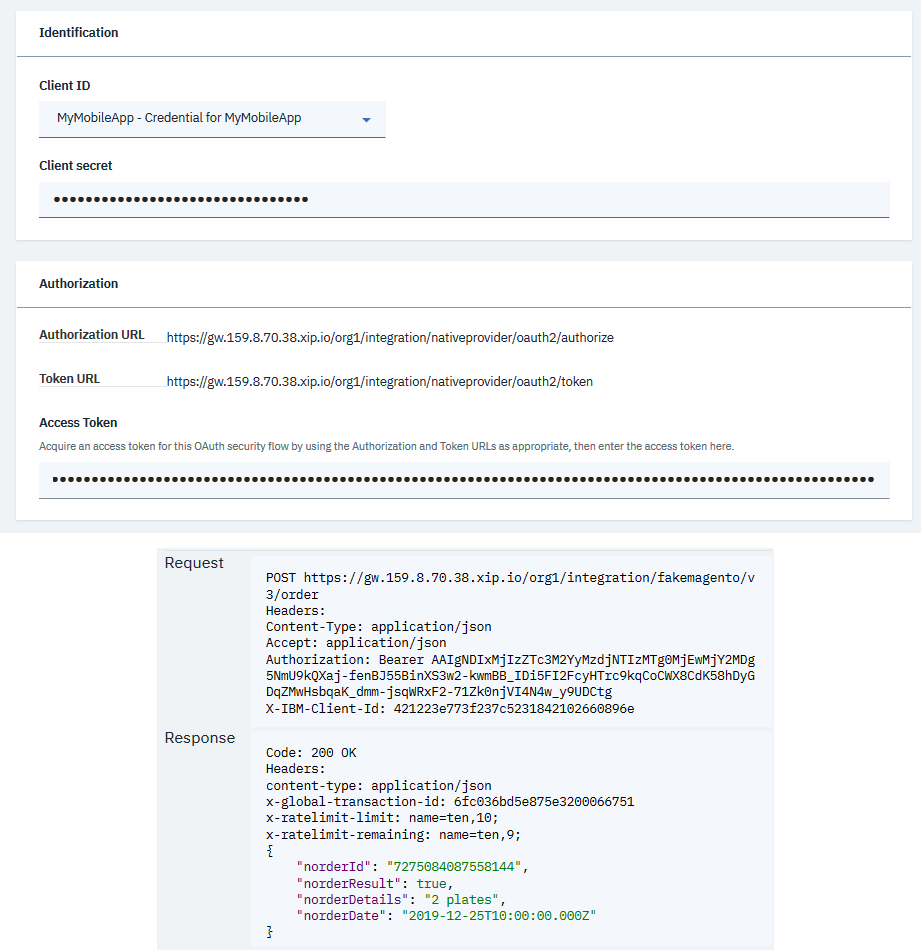
"refresh\_token\_expires\_in": 2682000,

"id\_token": "eyJraWQiOiJteXNpZ25rZXkiLCJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJmNDUwMzIwOS00OTI3LTQ0NDgtODdjMy1iZDM2YzA2ZjMyZjgiLCJpc3MiOiJJQk0gQVBJQ29ubmVjdCBBcyBPQXV0aCBhbmQgT0lEQyBQcm92aWRlciIsInN1YiI6ImZvbyIsImF1ZCI6IjQyMTIyM2U3NzNmMjM3YzUyMzE4NDIxMDI2NjA4OTZlIiwiZXhwIjoxNTg1OTM3MjgyLCJpYXQiOjE1ODU5MzM2ODIsImF0X2hhc2giOiJVV3p4aW1NeTNZRzJHOFhQd0xGbFV3In0.oa845Xm9hN8F0e1Vaij\_Xys0riGdkCV-7Q\_PAM1E\_zM"

}

**Using the Developer portal**

You get the Access Token directly using curl or Postman. In my case, I use Postman to get the Access Token. We select the MyMobileApp application, then enter the client\_secret,and the Access Token. We click on Generate link to automatically populate the parameters needed to call the API. Then click on Send button. This invokes the FakeMagento API using the access Token as a Bearer.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-access-21.png)

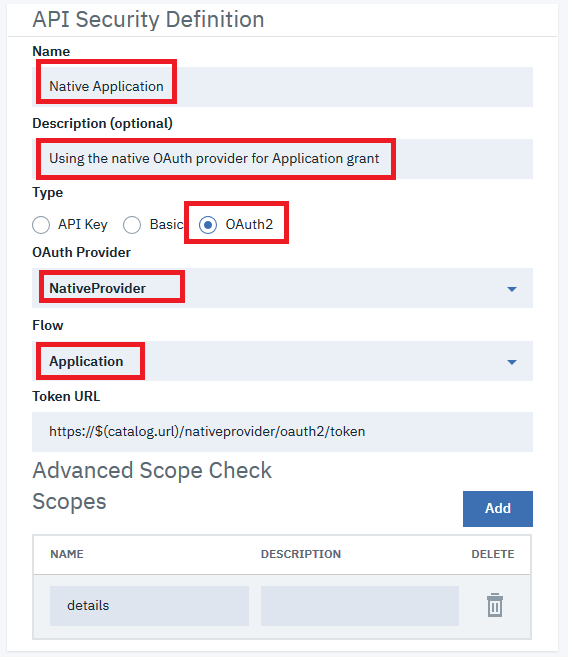
JSON Web Key (JWK) is specified at [RFC 7517](https://tools.ietf.org/html/rfc7517). A JSON Web Key (JWK) is a JavaScript Object Notation (JSON) data structure that represents a cryptographic key. I'm using a Simple JSON Web Key generator: [mkjwk](https://mkjwk.org/). OIDC specification is based on the use of the idtoken which is a JSON Web Token - JWT specified at [RFC 7519](https://tools.ietf.org/html/rfc7519).

**Protecting an API with OAuth - Client Credentials grant**

**Protect the API with OAuth - Application**

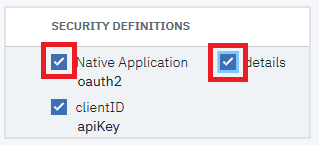
Let's protect, the FakeMagento version 4.0.0 API. Click on Develop and select the FakeMagento-4.0.0 API. Click on Security Definitions and click on Add button. Enter:

Name: Native Application   
Description: Using the native OAuth provider for Application grant   
Select OAuth2   
Select NativeProvider for the OAuth Provider   
Select Application for the Flow   
Select details for the scopes

Click Save button [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/application-secure-API-sec-def.png)

In the Security selection, select Native Application oatuh2 and the details scope.

Click Save button.

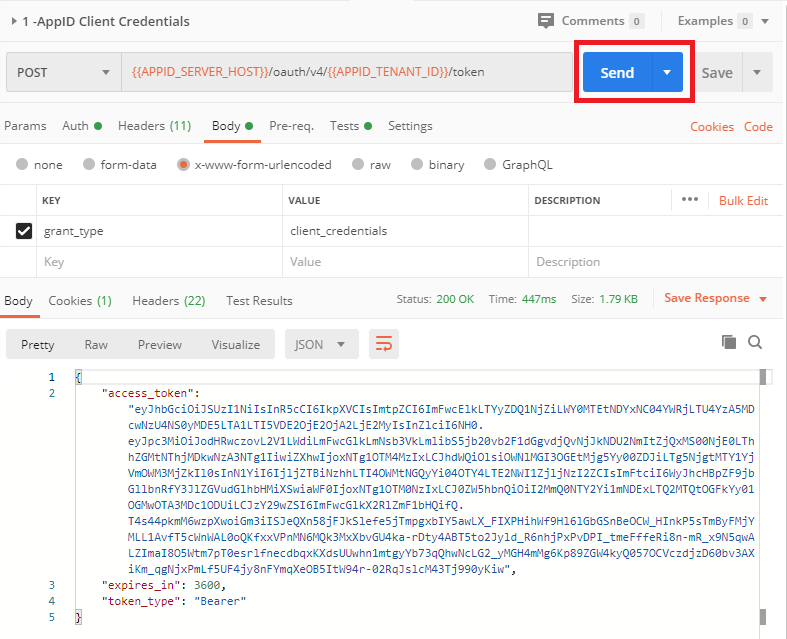
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/application-secure-API-sec.png)

**Tests**

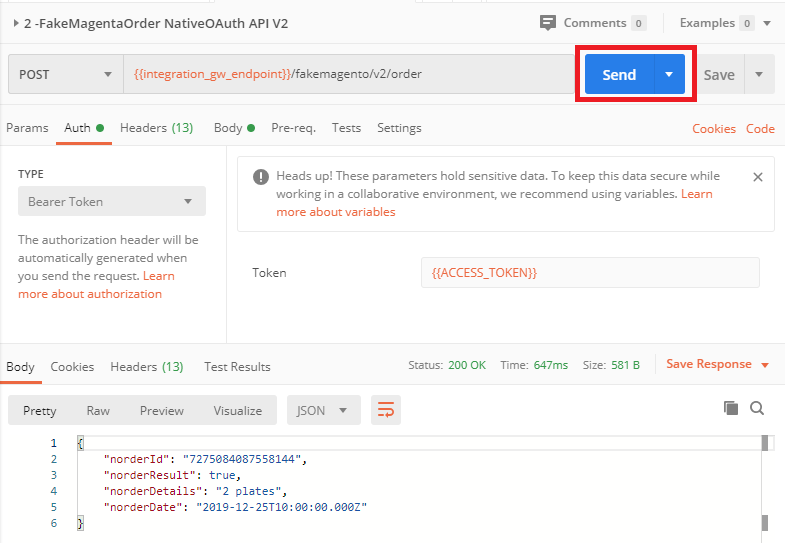
Not explained in detail here, but we publish the product (or use versioning with the publish capability), we are using the Integration environment. Then we subscribe to the Product with the Gold Plan and approve the subscription. The API is published and ready to use.

**Using POSTMAN**

1. Get token using the "1 -Client Credentials V4" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-application-11.png)

1. Use token to access FakeMagento V4.0.0 API using the "2 -FakeMagentaOrder Native OAuth API" request

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-application-12.png)

**Using curl**

Request

POST /org1/integration/nativeprovider/oauth2/token HTTP/1.1

Content-Type: application/x-www-form-urlencoded

accept: application/json

grant\_type=client\_credentials&client\_id=421223e773f237c5231842102660896e&client\_secret=556a75ce26097f96ea281ed47c1cf2e7&scope=details&APIm-Debug=true

Response

HTTP/1.1 200 OK

X-RateLimit-Limit: name=default,100;

X-RateLimit-Remaining: name=default,76;

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e87726800066a61

Content-Type: application/json

{

"token\_type": "Bearer",

"access\_token": "AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmV0AD6QpfILhTsnddGNJQ5MQIHdQI9eESHYs0QyOFDUc4x3A55wFGH2jfbiVLlhDhlddI5tUIe9x-pGgFgHMZZ8IGYVL2wYiB\_aOsElY1h8yNRymzvCzDWTFRU4UiaPGH4",

"scope": "details",

"expires\_in": 3600,

"consented\_on": 1585934952

}

Request

POST /org1/integration/fakemagento/v2/order HTTP/1.1

X-IBM-Client-Id: 421223e773f237c5231842102660896e

Content-Type: application/json

Accept: application/json

Authorization: Bearer AAIgNDIxMjIzZTc3M2YyMzdjNTIzMTg0MjEwMjY2MDg5NmV0AD6QpfILhTsnddGNJQ5MQIHdQI9eESHYs0QyOFDUc4x3A55wFGH2jfbiVLlhDhlddI5tUIe9x-pGgFgHMZZ8IGYVL2wYiB\_aOsElY1h8yNRymzvCzDWTFRU4UiaPGH4

{

"orderDetails": "2 plates",

"orderDate": "2019-12-25T10:00:00.000Z"

}

Response

HTTP/1.1 200 OK

X-RateLimit-Limit: name=ten,10;

X-RateLimit-Remaining: name=ten,9;

Accept: application/json

X-Client-IP: 10.126.64.177

X-Global-Transaction-ID: 6fc036bd5e877298000817c9

Content-Type: application/json

{

"norderId": "7275084087558144",

"norderResult": true,

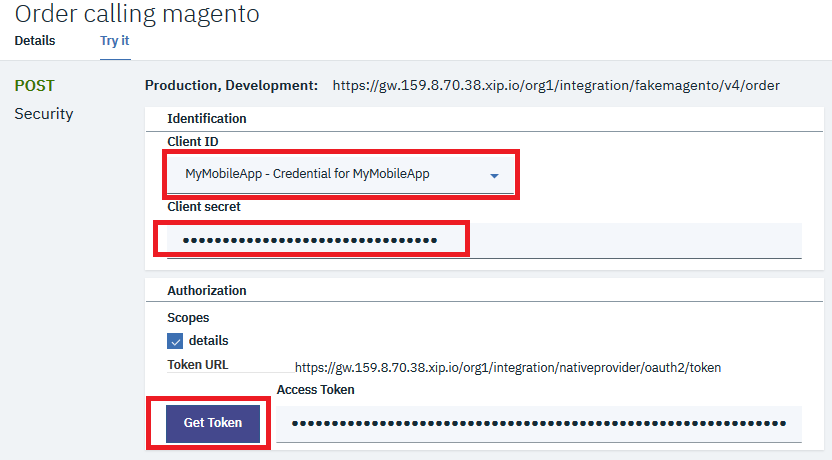
"norderDetails": "2 plates",

"norderDate": "2019-12-25T10:00:00.000Z"

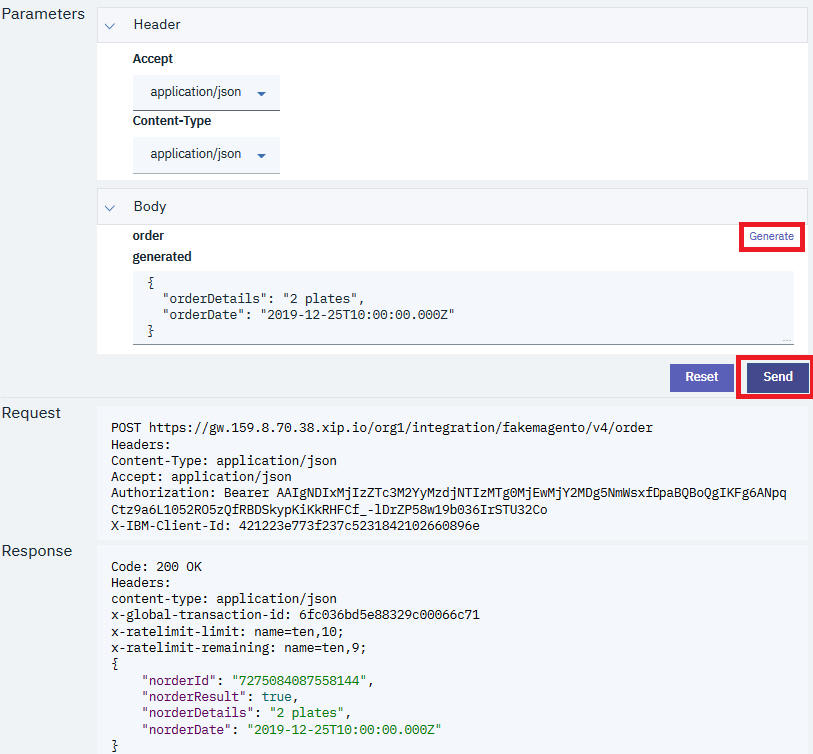
}

**Using the Developer portal**

We do not explain all the steps because it has been done in previous chapters. We sign in to the developer Portal, then select the FakeMagento V4.0.0 API and click on Try it. In the first step, we get the token. We select the MyMobileApp application, then enter the client\_secret, enter the username and password and click on the Get Token button. It returns the Access Token.

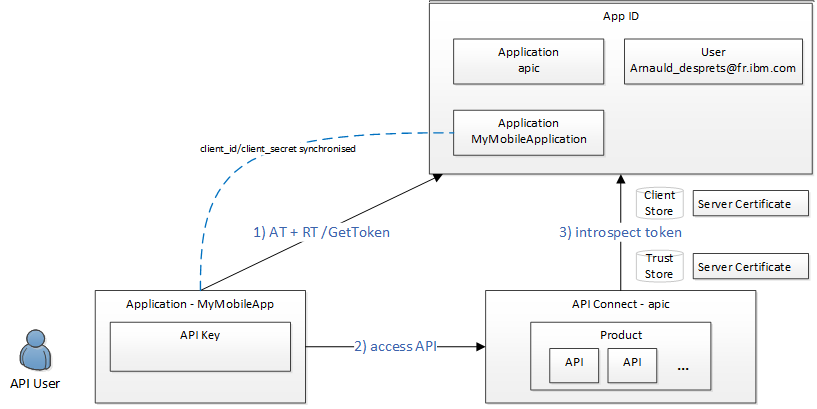
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-application-21.png)

In the second step, we click on Generate link to automatically populate the parameters needed to call the API. Then click on Send button. This invokes the FakeMagento API using the access Token as a Bearer.

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/test-application-22.png)

**Protecting an API with OAuth - External Provider**

In this scenario, we already have an OAuth Provider available. The integration with API Connect is very simple, the application first synchronise the API keys with the applications defined in the OAuth Provider. It can be done in two manners, either the API keys are created in API Connect and then added to the OAuth Provider, either the API Keys are created in the OAuth Provider and they are added to the application in API Connect. In this scenario, we will use the second option which will give us an opportunity to look at the API Connect CLI. At runtime, the application, first get the token from the OAuth provider directly, not going through API Connect (this is a more secured approach since the credentials do not flow in API Connect), then the application invoke the API with the Access Token in an Authorization header as a Bearer. API Connect then invokes the OAuth Provider with an Introspect call, if the calls return a 200 (or active=true), then the Access Token is valid and API Connect can proceed.

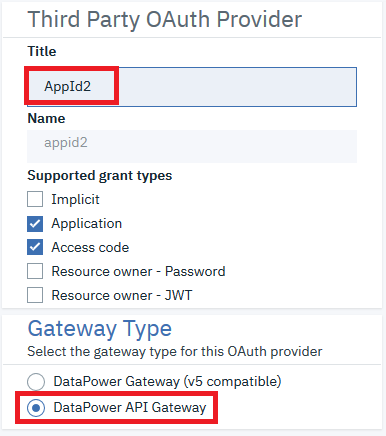
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/oauth-third-concepts.png)

**Create the OAuth Provider**

We again have decided to create the OAuth Provider in Console Manager.

As previously, click on Resources, then OAuth Providers. Click Add button selecting Third Party OAuth Provider on top left. Enter:

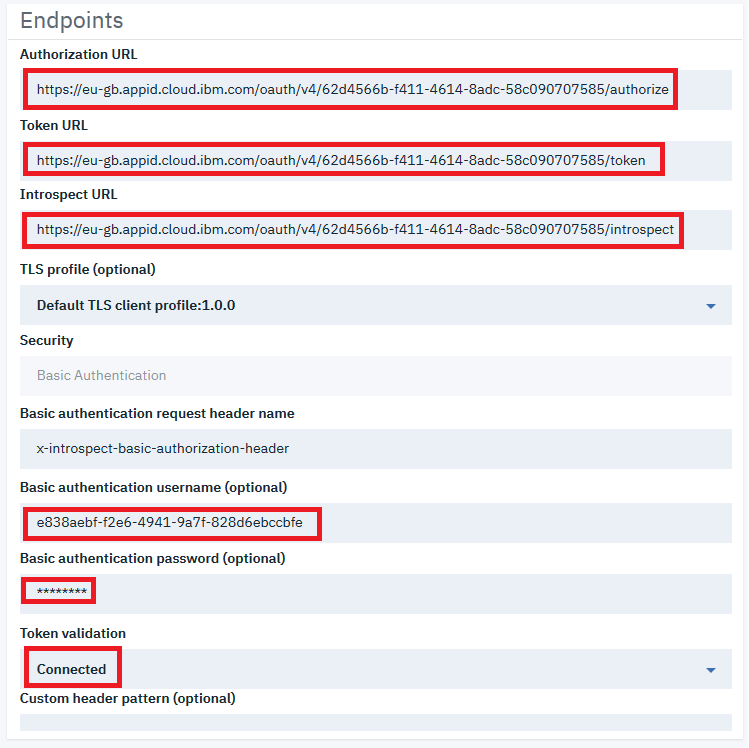
Title: AppId   
Keep Access code selected and select Application   
Select DataPower API Gateway

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

Click Next button In the Endpoints panel, enter:

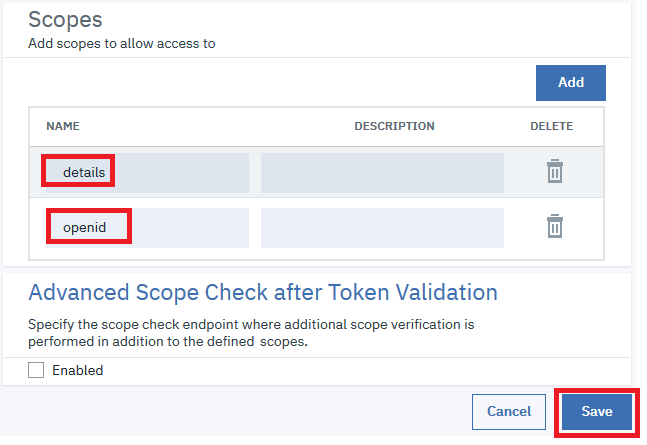
Authorization URL: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/authorize>   
Token URL: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/token>   
Introspect URL: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/introspect>   
TLS profile (optional): Default TLS client profile:1.0.0   
Basic authentication username (optional): e838aebf-f2e6-4941-9a7f-828d6ebccbfe   
Basic authentication password (optional):   
Select Connected for Token validation Click Next button

The only used endpoint is the Introspect URL. The Authorization and Token URL are used in the documentation only and publish along side the API.

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

For the scope enter:

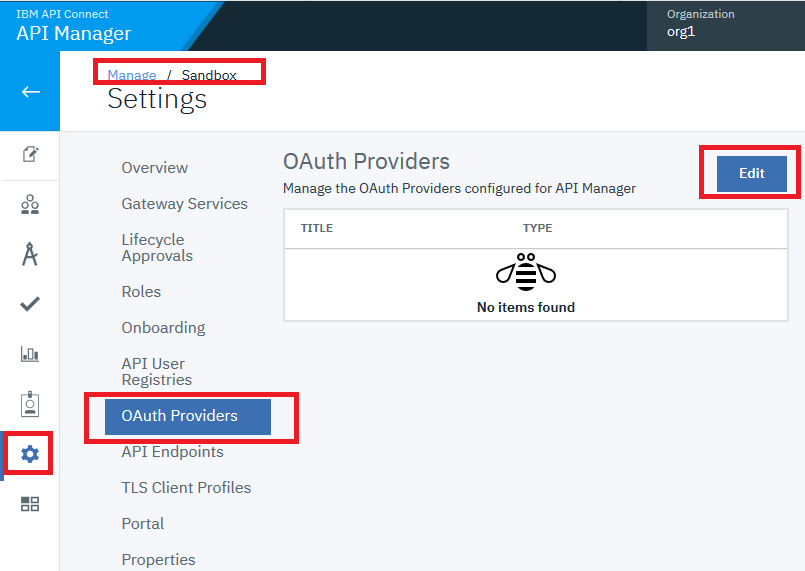
details for elevated access   
openid for OIDC support   
and then click Save button

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

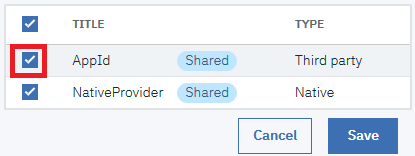
We are done with the configuration of the OAuth provider. We can now leave the Cloud Manager console.

**Associate the OAuth Provider with the different catalogs**

We need to make this OAuth Provider accessible in the various catalogs where we want to use it. In the Manager Console, click on Manage and select Sandbox, then Settings and OAuth Providers. Click on Edit button on the top left.

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

Click on the checkbox close to AppId

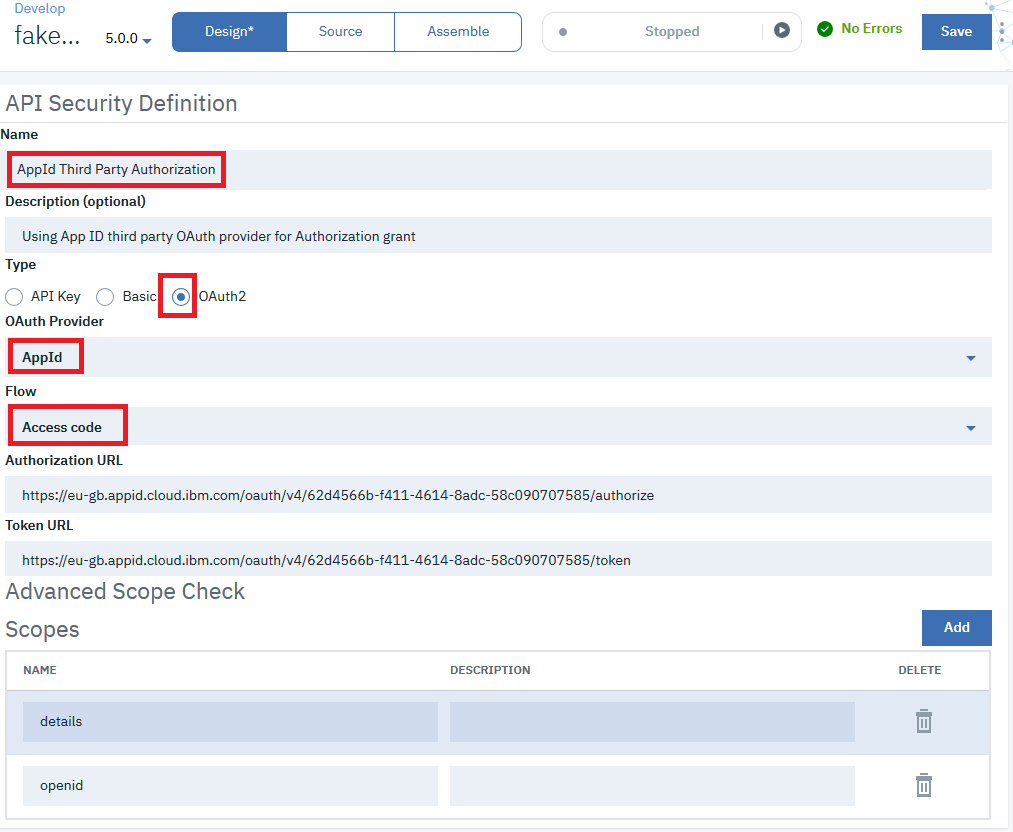
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/third-oauth-sandbox-associate-edit.png)

Repeat the same operation with the Integration Catalog. It is not yet accessible because we are not using it in any API.

**Protect the API with OAuth - Third Party**

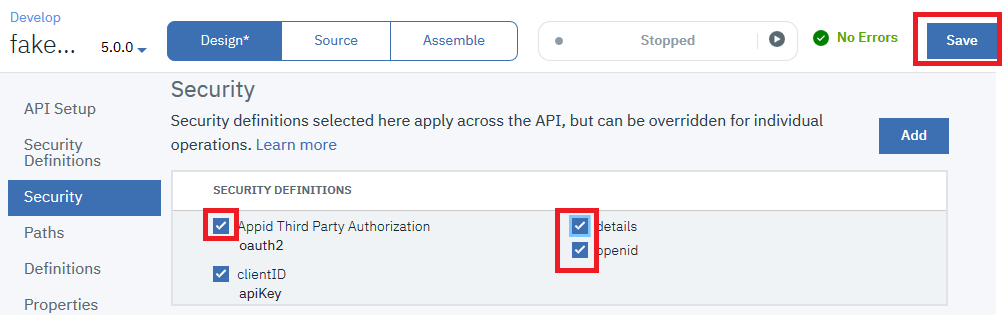
Let's protect, the FakeMagento version 5.0.0 API. Click on Develop and select the FakeMagento-5.0.0 API. Click on Security Definitions and click on Add button. Enter:

Name: AppId Third Party Authorization   
Description: Using App ID third party OAuth provider for Authorization grant   
Select OAuth2   
Select AppId for the OAuth Provider   
Select Access code for the Flow

[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/third-party-access-secure-API-sec-def.png)

Click Save button.

In the Security section, select AppId Third Party Authorization and the two scope details and openid, then click Save button.

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

**Synchronize the client credentials from AppId into API Connect Application definition**

What we do here is get the API keys defined in App ID for the application and then use the CLI to add this set of credentials to the MyMobileApp application. The only way to do this is using the CLI or the REST API. As said previously, an alternative would have been to update the credentials from API Connect to AppId with the AppId API.

* Log to the manager

apic.exe login -s <manager-endpoint> -u <user> -p <password> -r provider/default-idp-2

**Note:** From now on, I'm going to use a short cut/alias in order to simplify what I type. And notice that I'm using Windows so you may have to slightly modify those command if on Linux.

set apic2018=<path-to-apic> -s <manager-endpoint> -o <organisation>

Typing %apic2018% provides an extensive help. To update the application credentials, we need to find the name of the consumer org in the integration catalog and the name of the application.

* Get the name of the consumer organisation

%apic2018% consumer-orgs:list -c integration

orgdev1 [state: enabled] https://manager.159.8.70.38.xip.io/api/consumer-orgs/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda

* Get the name of the application

%apic2018% apps:list -c integration --scope catalog

mymobileapp [state: enabled] https://manager.159.8.70.38.xip.io/api/apps/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda/8ec77979-0877-443f-bc6d-17ba55f48b5a

* We need to specify the credentials information in a json file. Create a file called appidcredentialSet.json for example.

{

"type": "credential",

"api\_version": "2.0.0",

"name": "appid-credentials",

"title": " AppId Credentials for MyMobileApp",

"summary": " AppId Credentials for MyMobileApp",

"client\_id": "9ce0b78a-289c-4d2b-8968-165b5f9c726d",

"client\_secret": "NzNiMGJmYmUtMjNiYi00YmExLWIwY2UtZDg5MDAxZjFjZDI3"

}

* We can now update API connect:

%apic2018% -c integration --consumer-org orgdev1 -a mymobileapp credentials:create C:\temp\appidcredentialSet.json

* Let's check that the credentials have been correctly updated

%apic2018% -c integration --consumer-org orgdev1 -a mymobileapp credentials:list --format json

{

"total\_results": 2,

"results": [

{

"type": "credential",

"api\_version": "2.0.0",

"id": "43a84659-28e4-4bb9-8dbe-998cd478c6c0",

"name": "appid-credentials",

"title": " AppId Credentials for MyMobileApp",

"summary": " AppId Credentials for MyMobileApp",

"client\_id": "9ce0b78a-289c-4d2b-8968-165b5f9c726d",

"client\_secret\_hashed": "uSCwG4YW8Fg+PRf1U0UXYDkgQw5BLMDn4G5vOL+RERI=",

"created\_at": "2020-04-01T14:05:43.592Z",

"updated\_at": "2020-04-01T14:05:43.592Z",

"org\_url": "https://manager.159.8.70.38.xip.io/api/orgs/3f015cc4-9cb5-4d72-a202-008473d14a11",

"catalog\_url": "https://manager.159.8.70.38.xip.io/api/catalogs/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026",

"consumer\_org\_url": "https://manager.159.8.70.38.xip.io/api/consumer-orgs/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda",

"app\_url": "https://manager.159.8.70.38.xip.io/api/apps/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda/8ec77979-0877-443f-bc6d-17ba55f48b5a",

"url": "https://manager.159.8.70.38.xip.io/api/apps/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda/8ec77979-0877-443f-bc6d-17ba55f48b5a/credentials/43a84659-28e4-4bb9-8dbe-998cd478c6c0"

},

{

"type": "credential",

"api\_version": "2.0.0",

"id": "d8d6b731-5a46-41dc-a27f-a073c76c72dc",

"name": "credential-for-mymobileapp",

"title": "Credential for MyMobileApp",

"summary": "Credential for MyMobileApp",

"client\_id": "421223e773f237c5231842102660896e",

"client\_secret\_hashed": "91IcbIJ+mL/oC0EnPrroU7mzRGMrwROoja8KhT8s4RQ=",

"created\_at": "2020-03-05T13:28:03.994Z",

"updated\_at": "2020-03-05T13:28:03.994Z",

"org\_url": "https://manager.159.8.70.38.xip.io/api/orgs/3f015cc4-9cb5-4d72-a202-008473d14a11",

"catalog\_url": "https://manager.159.8.70.38.xip.io/api/catalogs/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026",

"consumer\_org\_url": "https://manager.159.8.70.38.xip.io/api/consumer-orgs/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda",

"app\_url": "https://manager.159.8.70.38.xip.io/api/apps/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda/8ec77979-0877-443f-bc6d-17ba55f48b5a",

"url": "https://manager.159.8.70.38.xip.io/api/apps/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda/8ec77979-0877-443f-bc6d-17ba55f48b5a/credentials/d8d6b731-5a46-41dc-a27f-a073c76c72dc"

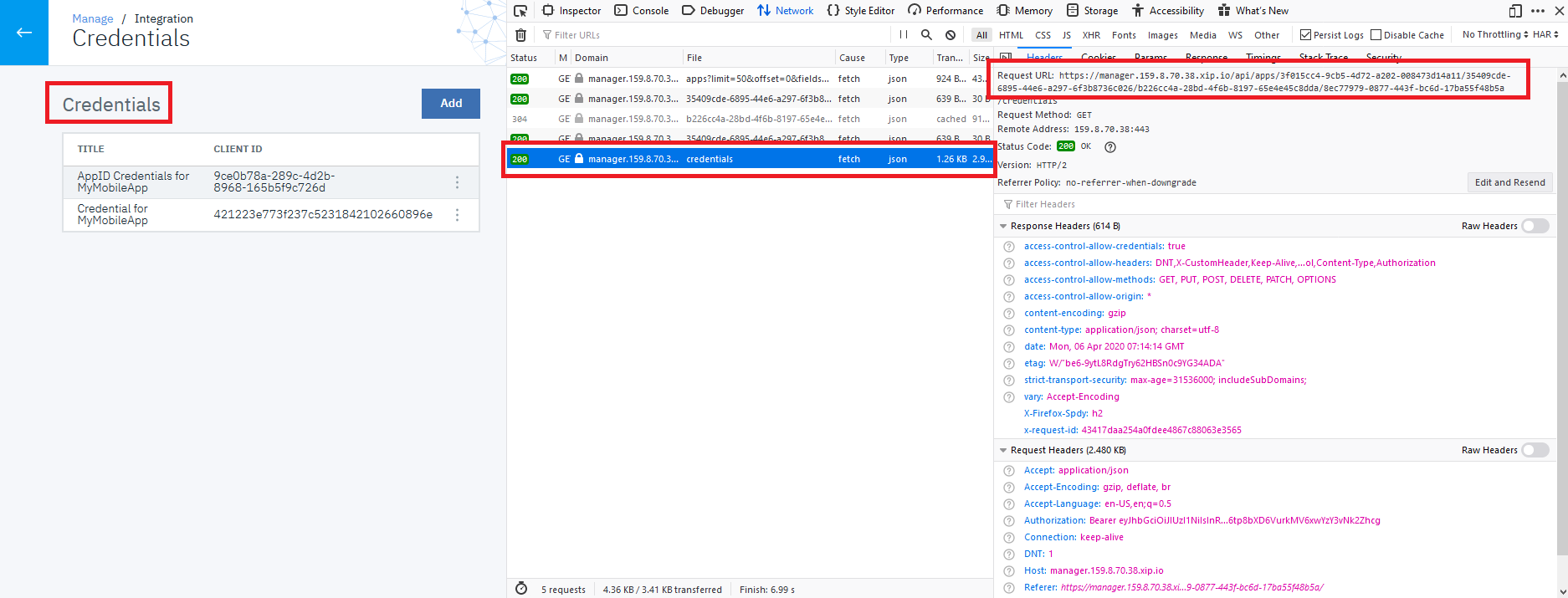
}

]

}

**Hint:** using the --debug on apic gives the equivalent curl requests. The it is possible to directly invoke object with REST API. You can get the URL of each object using the web-dev-tools in your browser. In the command below, you can get the Access Token, client\_id and client\_secret using the --debug option on the login call.

curl -k -H "accept: application/json" -H "Authorization: Bearer " -H "X-Ibm-Client-Id: <client\_id>" -H "X-Ibm-Client-Secret: <client\_secret>" "<https://manager.159.8.70.38.xip.io/api/apps/3f015cc4-9cb5-4d72-a202-008473d14a11/35409cde-6895-44e6-a297-6f3b8736c026/b226cc4a-28bd-4f6b-8197-65e4e45c8dda/8ec77979-0877-443f-bc6d-17ba55f48b5a/credentials>"

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

TO BE COMPLETED

**Tests**

Not explained in detail here, but we publish the product (or use versioning with the publish capability), we are using the Integration environment. Then we subscribe to the Product with the Gold Plan and approve the subscription. The API is published and ready to use.

**Using POSTMAN**

As f or the Authorization grant with the native OAuth Provider we use Postman to directly get the access token.

Using the "1 -Access Token Authorization V5" request. Click on the Authorization link, then click on Get New Access Token

**Using curl**

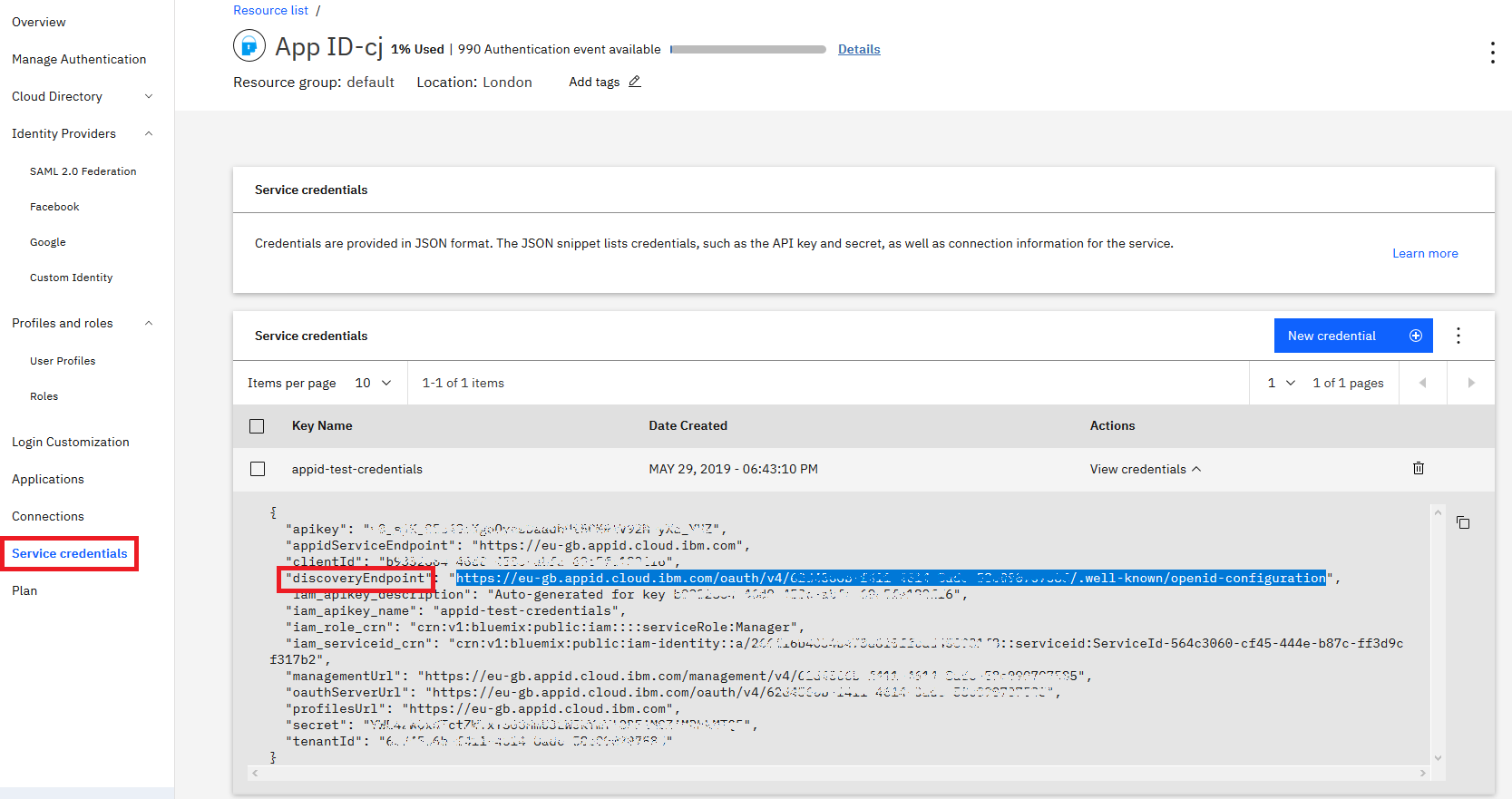
**Using an OIDC Registry to protect the platform**

**Gather the various information needed**

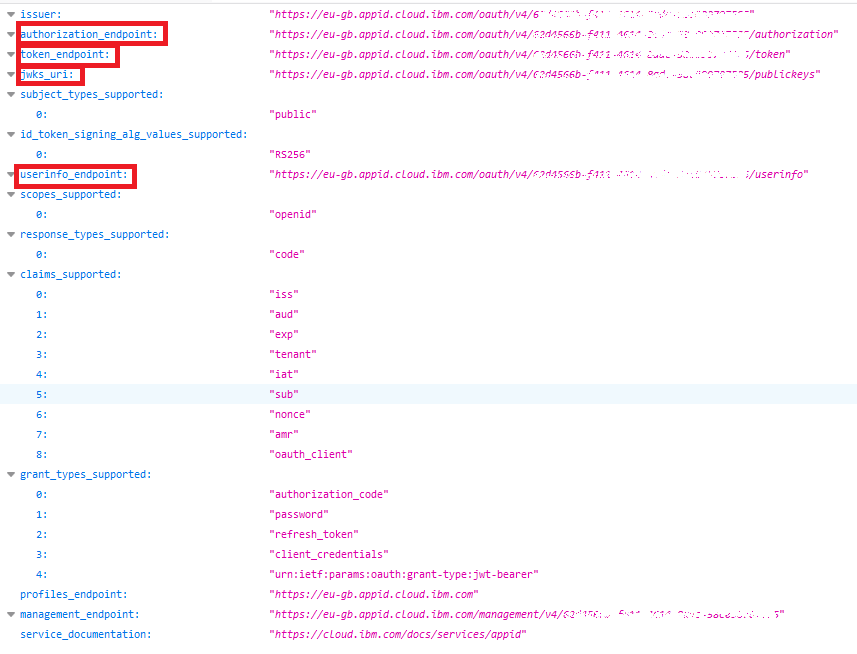
So far, we have looked mostly as the API Security, now let's see the protection of the platform (Cloud Manager et API Manager). The most common and simple way to protect the platform is to use one or more LDAP servers. A very simple way is also to use the Local User Registry, but if it is ok to include the user in it fore the CLoud Manager, because we may say that the number of users may be small (the platform administrators), it is probably not suitable for large number of users for the organizations and for the Portal. The concern here being the management side and thefact that it is not integrated with the entreprise IAM solution. With the adoption of OIDC, API Connect offers the possibility to also use an OIDC provider to rotect the platform (Cloud Manager/API Manager and Portal). As seen previously, you can combine the various authentication mechanisms as well and have a different ways of authenticating for two organizations andalso various ways for two portals.

In this first section, we are going to configure an OIDC provider to protect a portal.

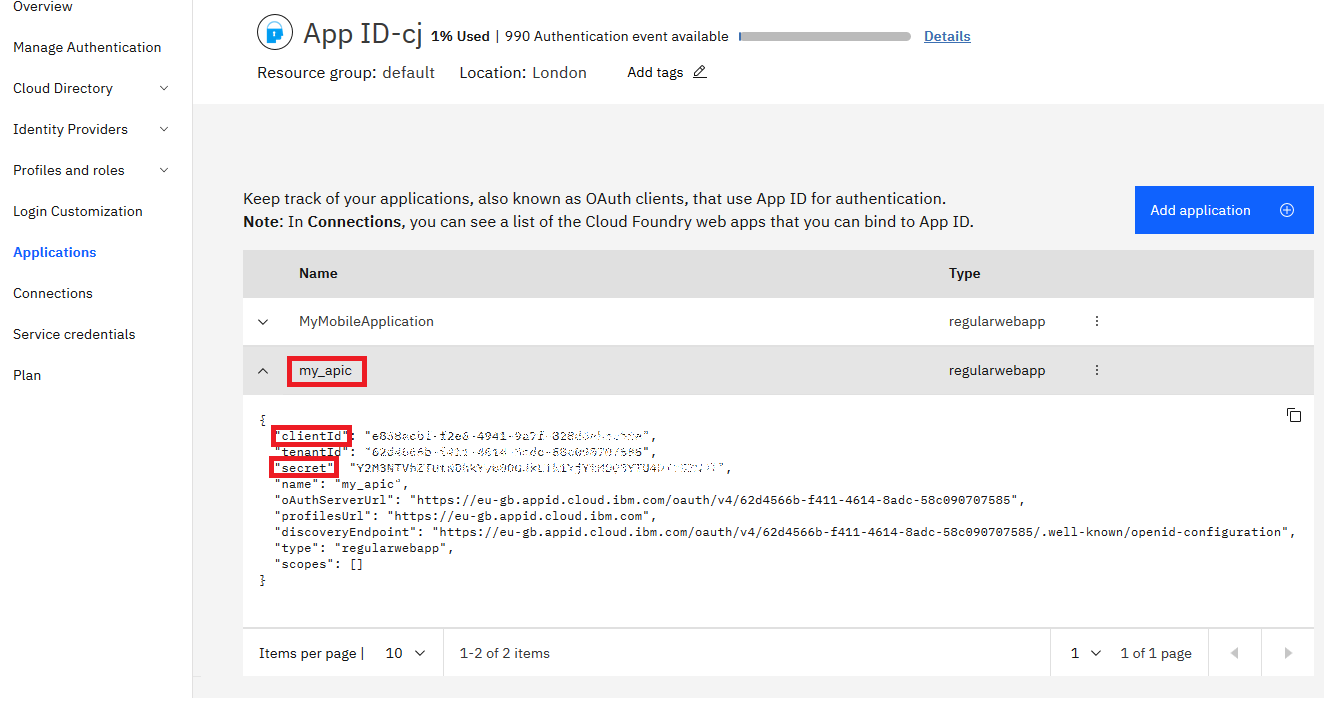
We need to find some information from the OIDC provider, in our case AppId? This information will be used to complete the definitions in API Connect. The first things is to get the various endpoints, there is a very easy to get them. Go to the Service Credentials sections in AppId. Copy the discoveryEndpoint URL. For me, this is <https://eu-gb.appid.cloud.ibm.com/oauth/v4/tenant_id_value/.well-known/openid-configuration> (where tenant\_id\_value is my tenant id).

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

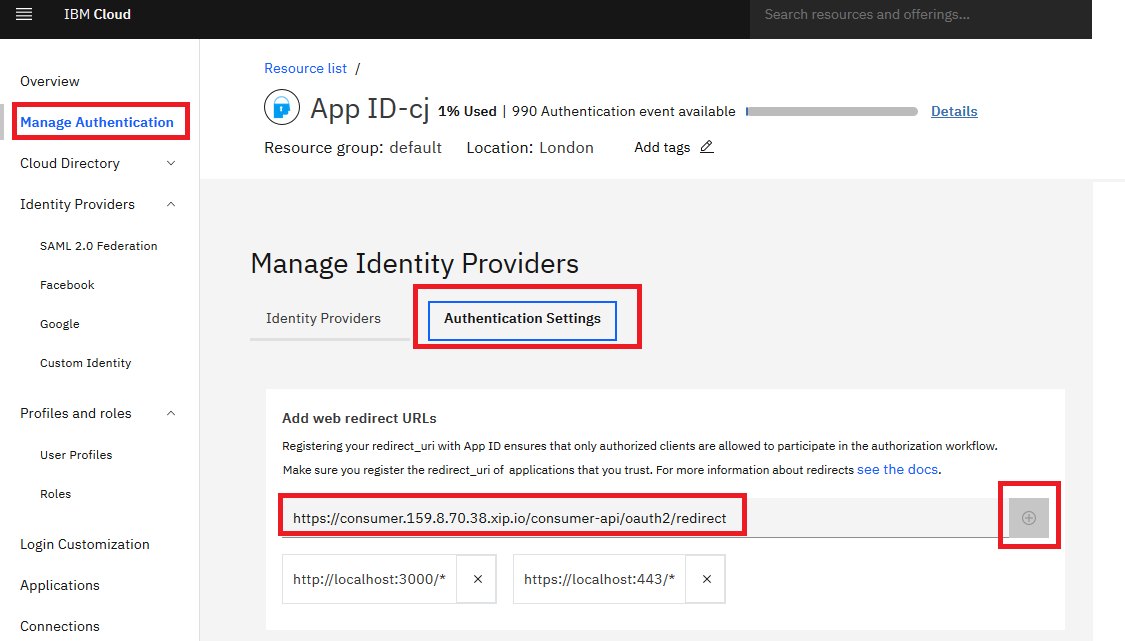
Then in a browser, past it, it gives you all the endpoints.

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

You also need to get the client\_id and client\_secret to access AppId from API Connect. In my case, I use the application called my\_apic.

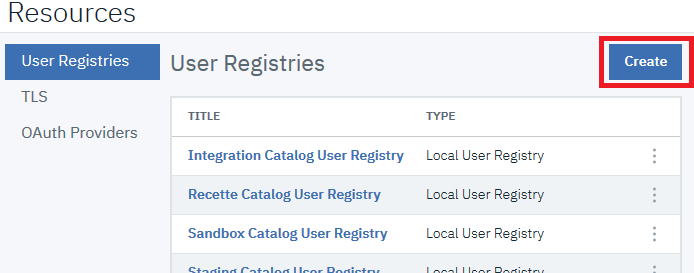
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/appid-application.png)

We need to update one aspect of the configuration, the redirect URL. So far, we used a dummy values for this values in the previous test. This time we need to use the proper value. The value is <consumer\_endpoint>/consumer-api/oauth2/redirect, in my case: <https://consumer.159.8.70.38.xip.io/consumer-api/oauth2/redirect> In AppId, click on Managage Authentication, then Authentication Settings, then enter the web redirect URL, select the + sign.

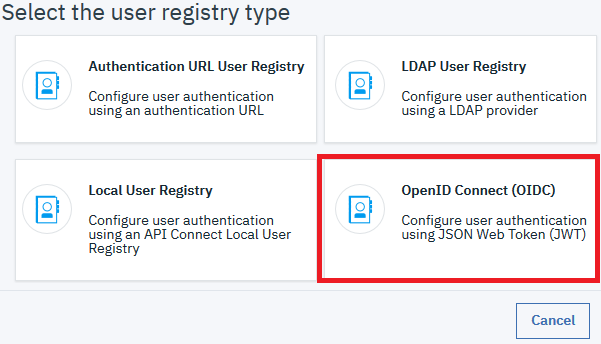
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/appid-redirect-url.png)

**Create the OIDC User Registry**

We are ready to configure API Connect. In the API Manager, click on Resources, then User Registries and click on the Create button

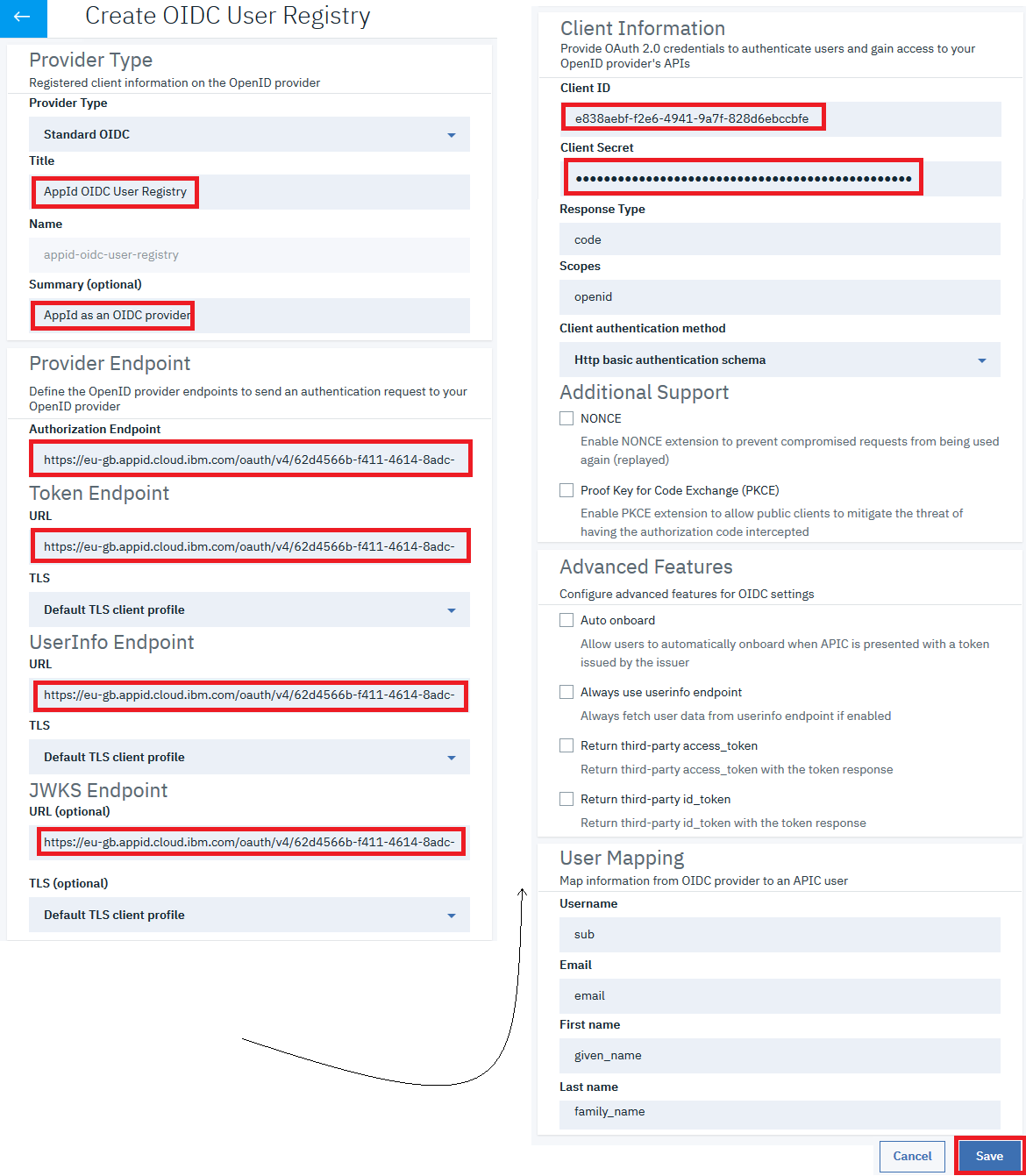
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/oidc-ur-button.png)

Select OpenID Connect (OIDC)

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

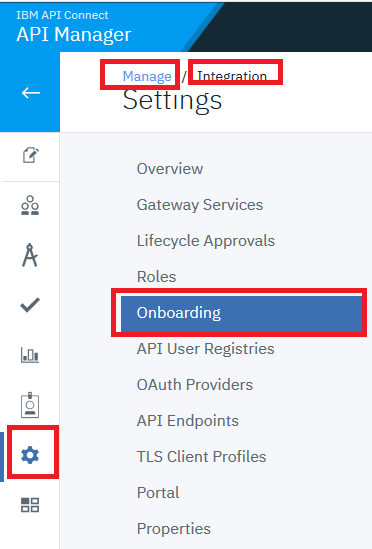
Enter the following:

Title: AppId   
Summary: AppId as an OIDC provider   
Authorization Endpoint: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/authorization>   
Token Endpoint: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/token>   
UserInfo Endpoint: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/userinfo>   
JWKS Endpoint: <https://eu-gb.appid.cloud.ibm.com/oauth/v4/62d4566b-f411-4614-8adc-58c090707585/publickeys>   
Client\_id: client\_id   
Client\_secret: client\_secret

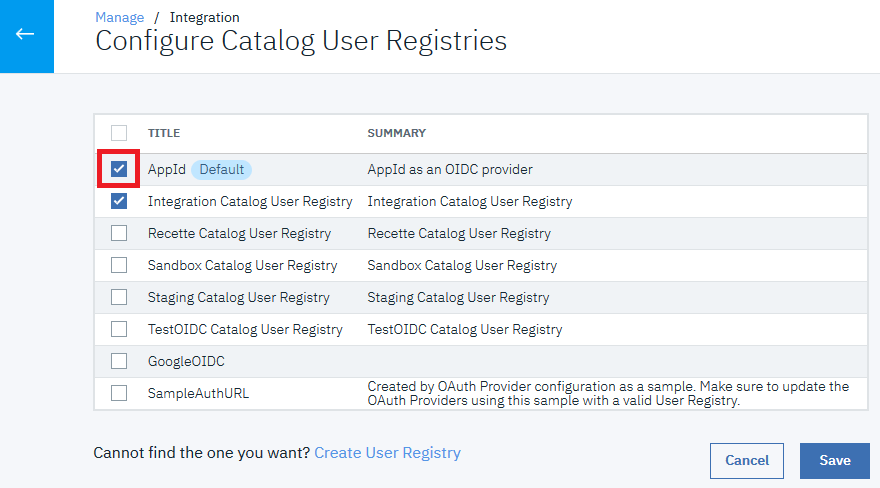
[](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/oidc-ur-create.png)

**Associate the OIDC User Registry with a Catalog**

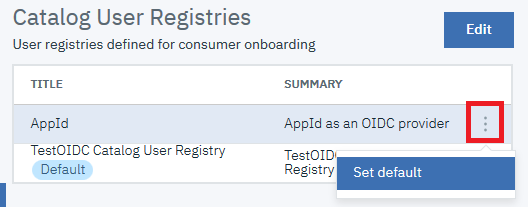
We are going to associate it with the Integration Catalog. Click on Manage, then on Integration Catalog, then on Settings, then on Onboarding.

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

Click on the Edit button, close to Catalog User Registries, and select AppId.

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

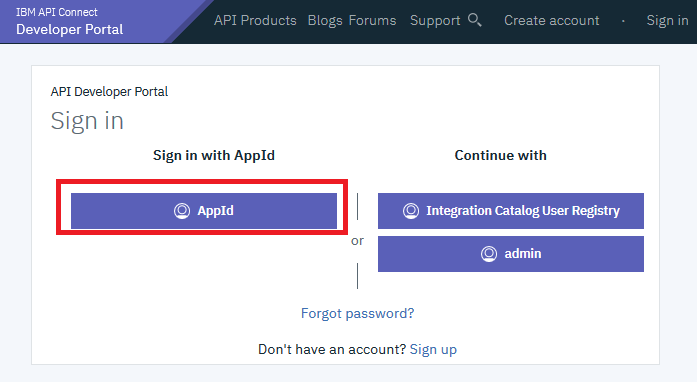
Set as default user registry, click on the three vertical dots, closed to AppId and select Set default.

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

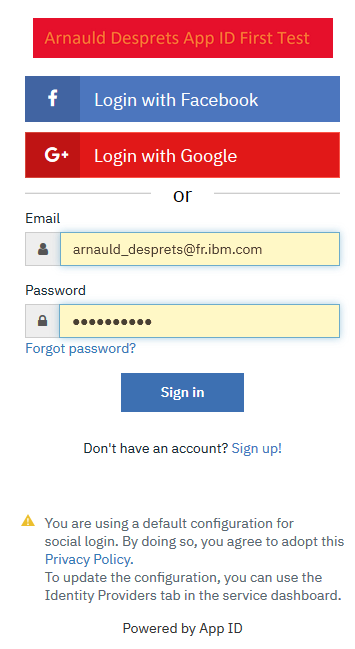
We are ready to test the configuration

**Test the new configuration**

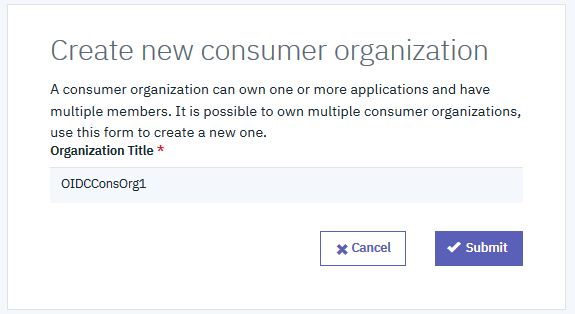
Let's go to the Integration Developer Portal. In a browser enter the URL of your portal, in my case, <https://portal.159.8.70.38.xip.io/org1/integration/user/login>. The page has changed, you can now see the a new button to authenticate to AppId. Click on the AppId button.

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

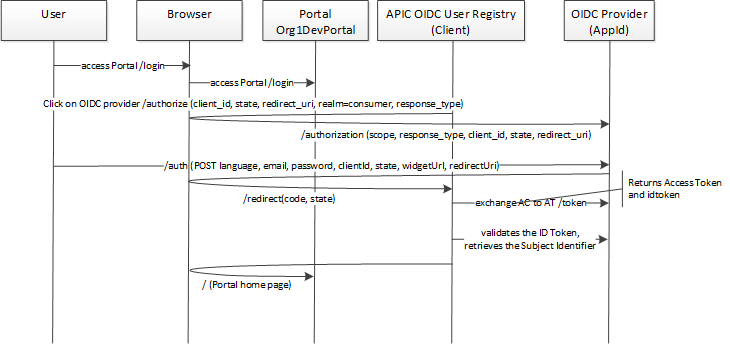
You are redicted to the login page of the OIDC Provider. Notice that in my case, it is configured to also provide additional ways of authentication.

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

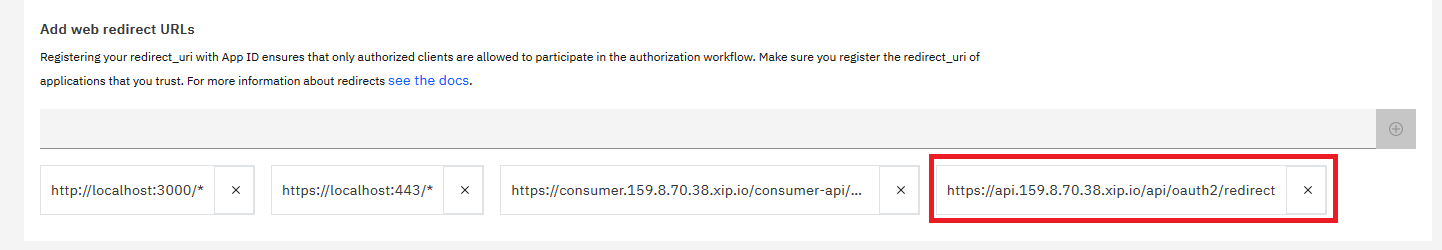
The user is logged in. Because the user has never logged in to the Portal, it needs to create the consumer organization. Enter OIDCConsOrg1 for example.

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

Below a sequence diagram explaining the various interactions including the several redirection that have happened under the cover. The OIDC implementation isq based on [Hybrid Flow as definied in OpenID Connect Core 1.0](https://openid.net/specs/openid-connect-core-1_0.html#HybridFlowAuth)

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

**OIDC Proivder for an organisation**

In this chapter, we dont detail everything because it is very similar to the protection of the Portal with an OIDC provider. We need to add a redirect URI for the Organisation to come back from the OIDC Provider to API Connect [](https://github.com/ADesprets/bluemix-labs/blob/master/Lab%20API%20-%20Manage%20your%20APIs%20with%20API%20Connect/images/oidc-mgr-redirect.png)

**Step 13 - Testing the quality of your API**

In this step we are going to see quickly how to use IBM API Connect Test and Monitor. The documentation associated with IBM API Connect Test and Monitor is available at <https://ibm-apiconnect.github.io/test-and-monitor/gettingstarted>. This tool addresses in a no code approach testing the API with a fine granularity in the test and either on demand or with scheduled testing. This provides capbilities to create tests that are very fined grained, in the sense that you are able to validate API (or group of API) either onDemand or on a regular schedule. This then allows to display functional and performance dashboards in order to better understand how the API are behaving. This is not for performance testing but quality testing. If an error occur you can receive an email, and deep dive to understand why it failed. It is also a mechanism to validate that all your API are sill working if a change was performed to the platform, API, etc ...

**Creating a test**

There are several ways to ceate a test using: a call, a Postman collection, from scratch.

**Running a test**

**Scheduling a test**

**Using the dashboards**

Congratulations. You have completed this Lab!

**Additional Resources**

For additional resources please look at the to the following:

* [API Connect Developer Center](https://developer.ibm.com/apiconnect)
* [API Connect v2018 Knowledge Center](https://www.ibm.com/support/knowledgecenter/SSMNED_2018/com.ibm.apic.overview.doc/api_management_overview.html)
* [Follow us @ibmapiconnect](https://twitter.com/ibmapiconnect)
* [POT API Connect Customization](https://ibm-apiconnect.github.io/faststart/)
* [Royal mail portal](https://developer.royalmail.net/node/2757)