Exercise 10

API Management and Governance including Analytics

Prior Knowledge
RESTful services

Objectives
Understand API management and key issuing.
Understand API Analytics.
Be able to configure the API Manager and Analytics, and use OAuth2 Bearer Tokens

Software Requirements
OpenJDK 1.8
WSO2 API Manager 3.0.0 (AM)
WSO2 API Manager Analytics 3.0.0
Node.js and npm (and other existing APIs)

1) Install the WSO2 API Manager and Analytics servers:

```
sudo rm -rf ~/servers
mkdir ~/servers
cd ~/servers
unzip ~/Downloads/wso2am-3.2.0.zip
unzip ~/Downloads/wso2am-analytics-3.2.0.zip
```

2) Now enable analytics:

```
code wso2am-3.2.0/repository/conf/deployment.toml
```

Go to the **Analytics** section of the TOML file on **line 93.**
Uncomment the whole block (down to line 100).
Change enable from **false** to **true**, then save.

It should look like this:

```
92
93  [apim.analytics]
94  enable = true
95  store_api_url = "https://localhost:7444"
96  username = "$ref{super_admin.username}"
97  password = "$ref{super_admin.password}"
98  event_publisher_type = "default"
99  event_publisher_impl = "org.wso2.carbon.apimgt.usage.publisher.APIMgtUsageDataBridgeDataPublisher"
100 publish_response_size = true
101
102 #[[apim.analytics.url_group]]
103 #analytics_url =["tcp://analytics1:7611","tcp://analytics2:7611"]
```

3) From a fresh terminal window or tab start the WSO2 API Manager Analytics:

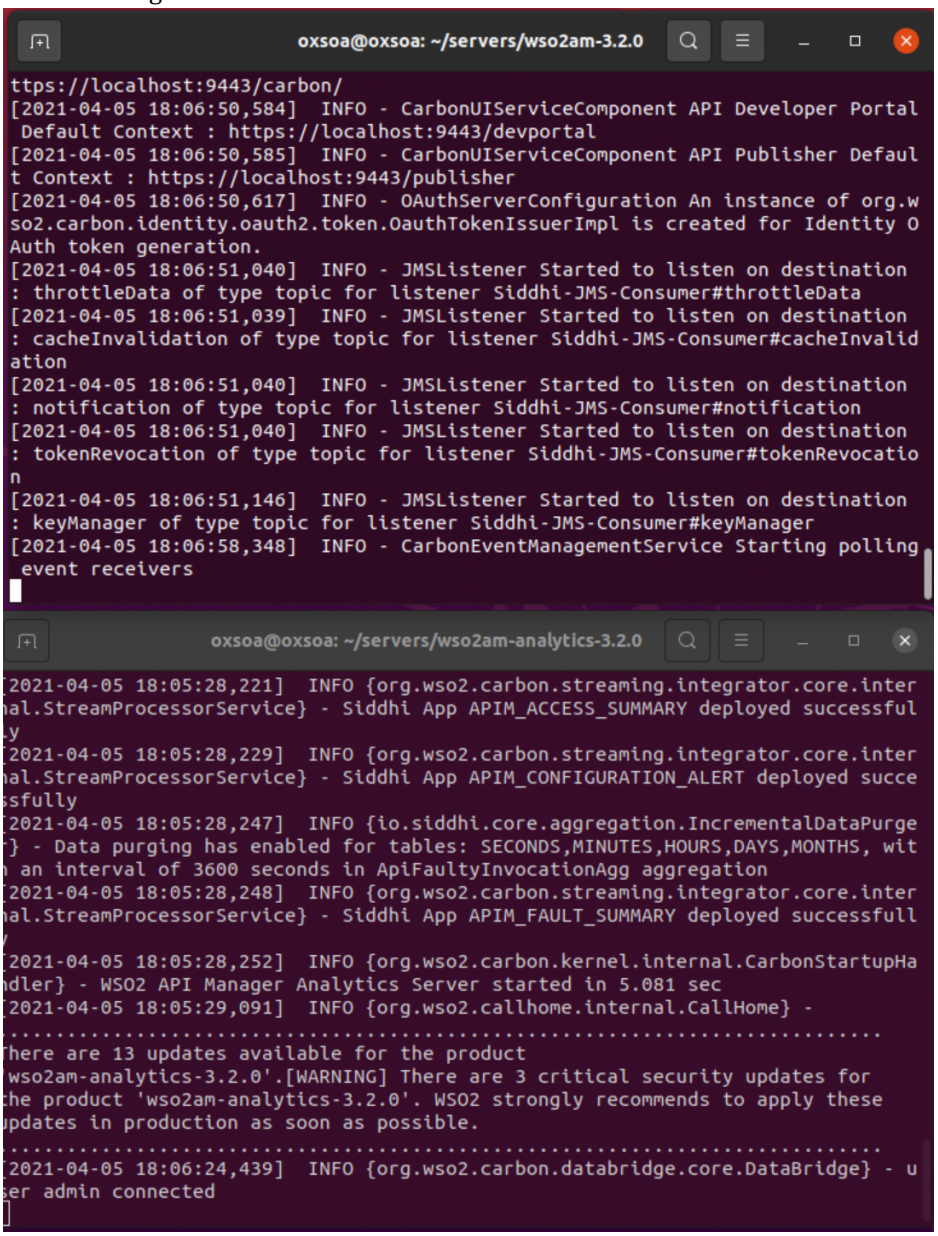
```
cd ~/servers/wso2-am-analytics-3.0.0
bin/worker.sh --run
```

4) Then in another terminal window start the WSO2 API Manager:

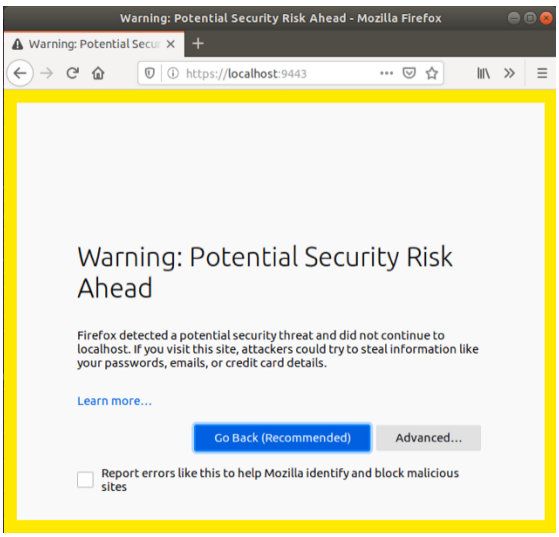
```
cd ~/servers/wso2am-3.0.0
```

```
bin/wso2server.sh
```

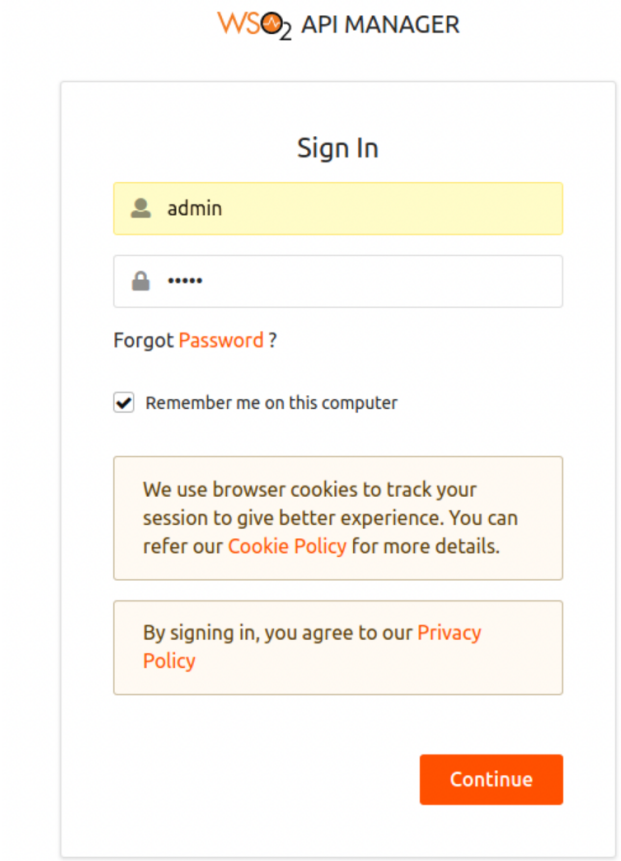
5) Wait until it has started. The first time run is a bit slower as it is performing setup. You should see something like:



6) Once both servers are started, check that you can access the web interface of the API Manager
   a. https://localhost:9443/ (AM console)
      If this is the first time you try this server you may need to allow the self-signed certificate. **Advanced,** then **Accept the Risk and Continue.**

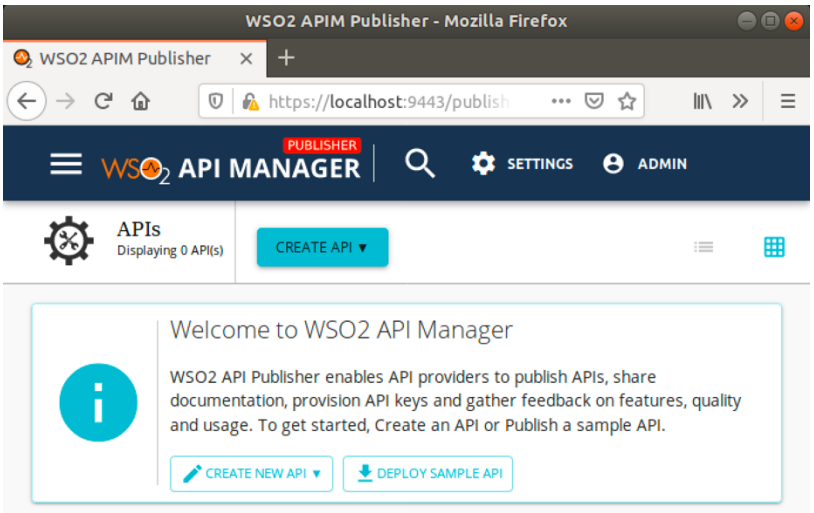


7) You should see something like this:



8) Log in as **admin/admin**

You should see:



9) An API creator uses the **API Publisher** system to create and publish APIs into the **API Store (Developer Portal)**. Firstly we will follow the tutorial to create a managed API. Then we will try it out using the API Store.

10) Let's create a simple hello API to use as the backend for this, using docker.

In a terminal window type:

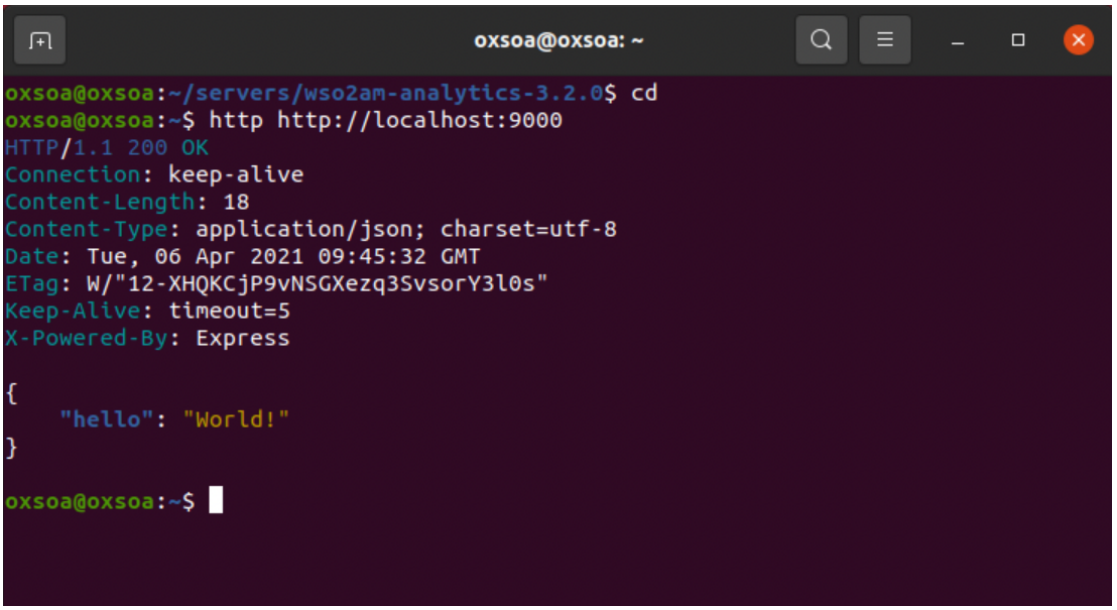
```
docker run -d -p 9000:8000 pzfreo/hello-api:1.0.0
```

(Source code is here: https://github.com/pzfreo/hello-api.git)

Notice we are exposing the port on 9000

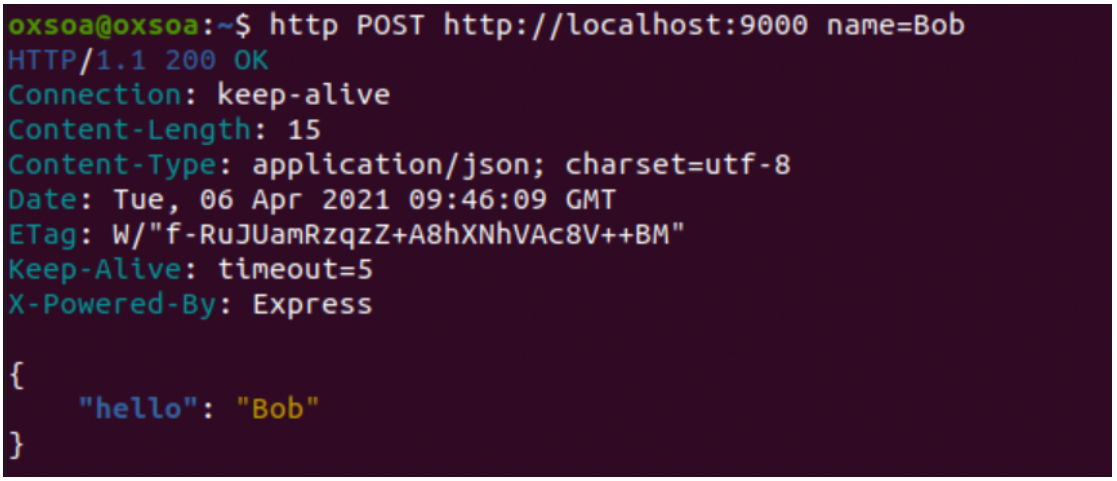
11) Test it out to see how it works:

```
http http://localhost:9000
```

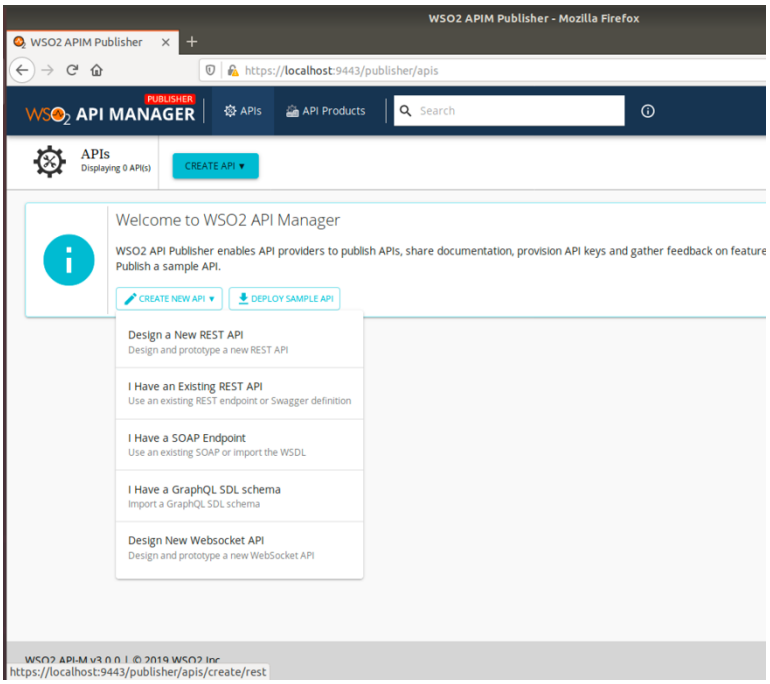


And

```
http POST http://localhost:9000 name=Bob
```



12) Back on the Publisher click the **Create New API** button. You will see:

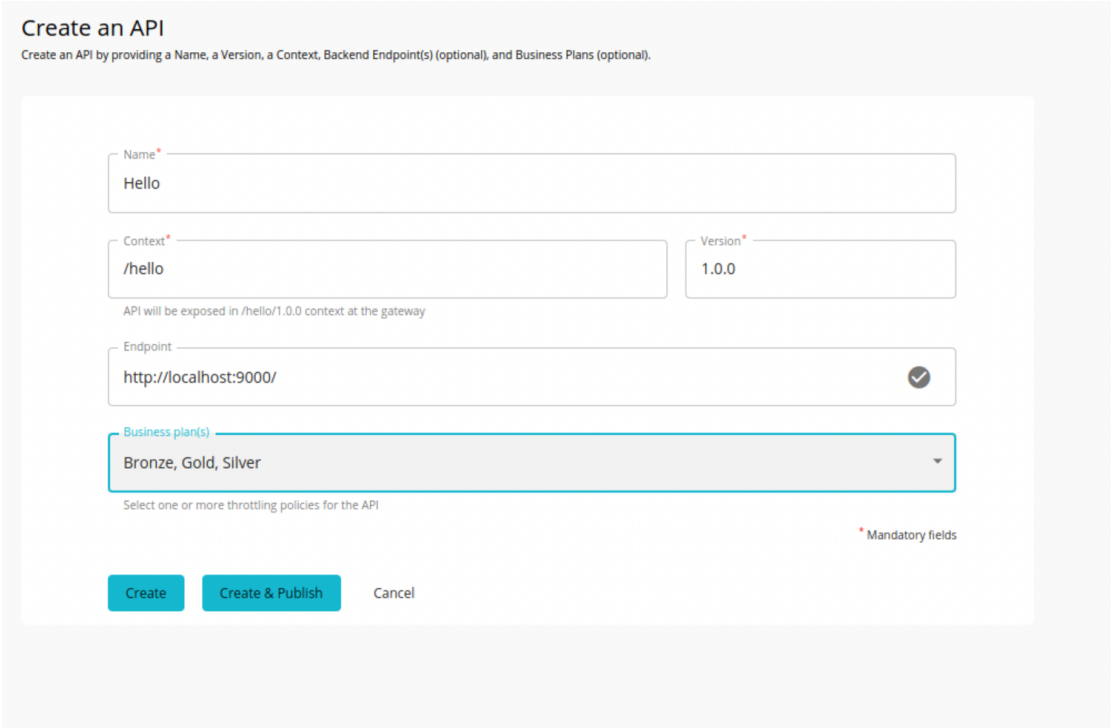


13) Click **Design a new REST API**

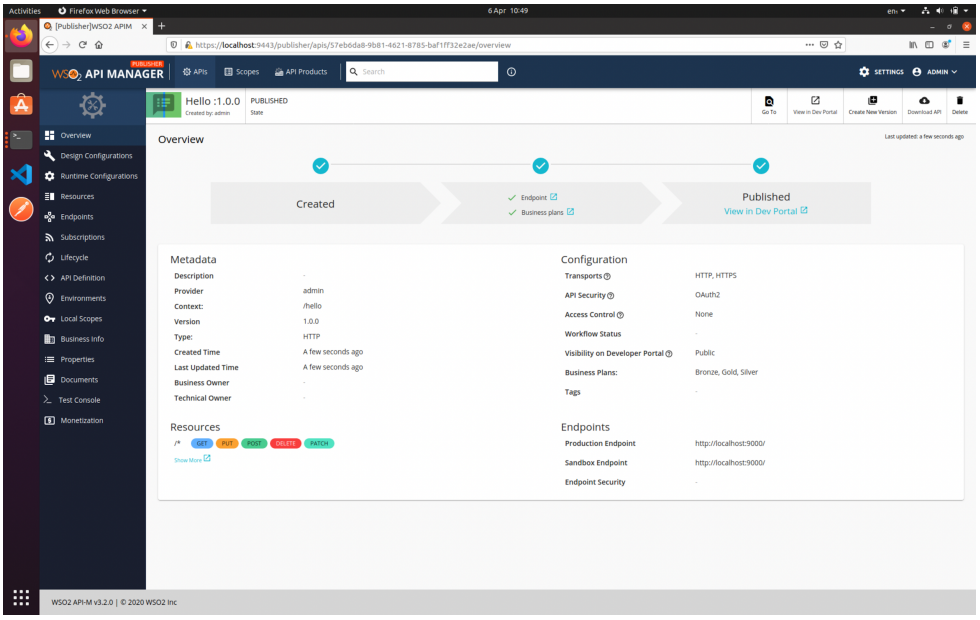
Fill in as follows:

Name:	Hello
Context:	/hello
Version:	1.0.0
Endpoint:	http://localhost:9000/
Business Plans:	Choose Gold, Silver and Bronze

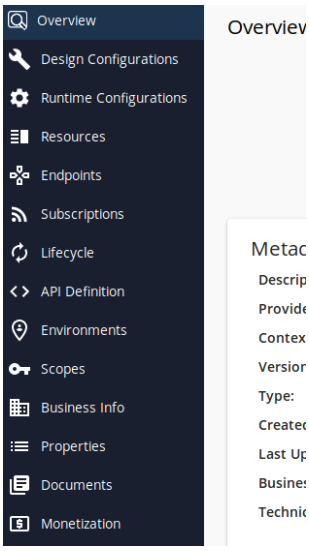
Please make sure you have a / at the END of the Endpoint URL.



14) Now click **Create and Publish**
Your screen should look like:

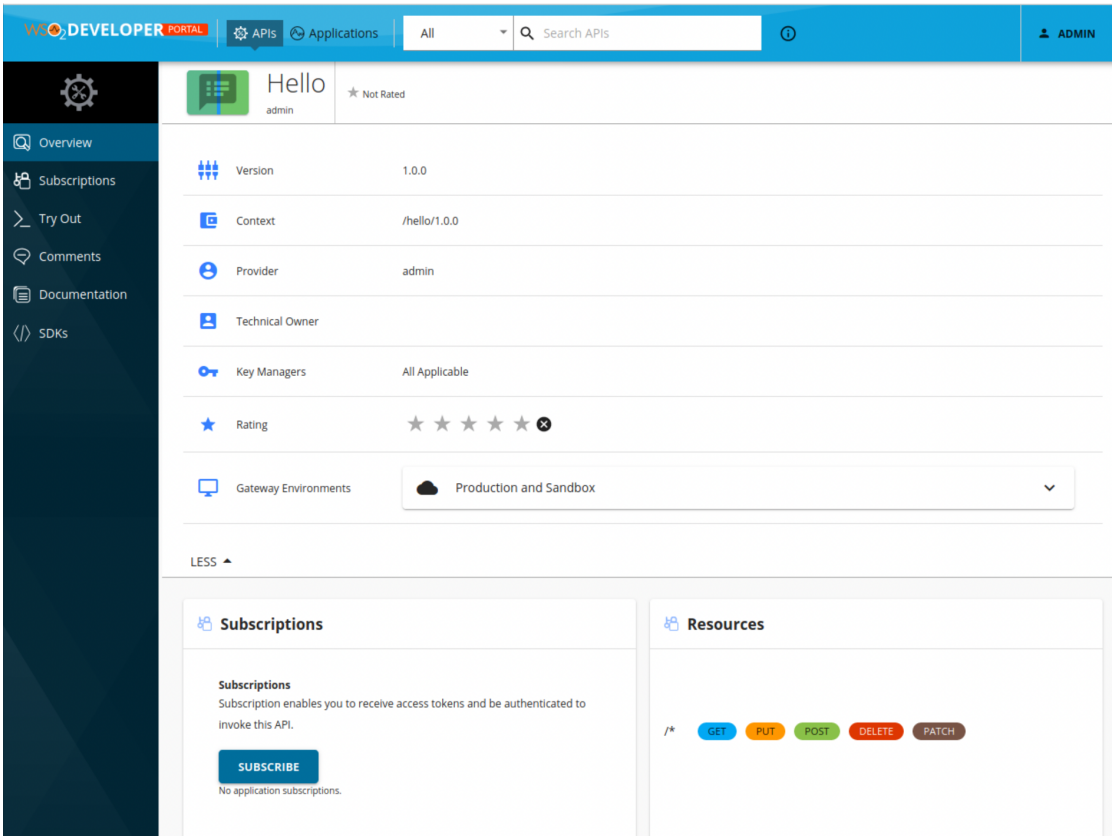


Notice that there is a lot more you can do with your API:



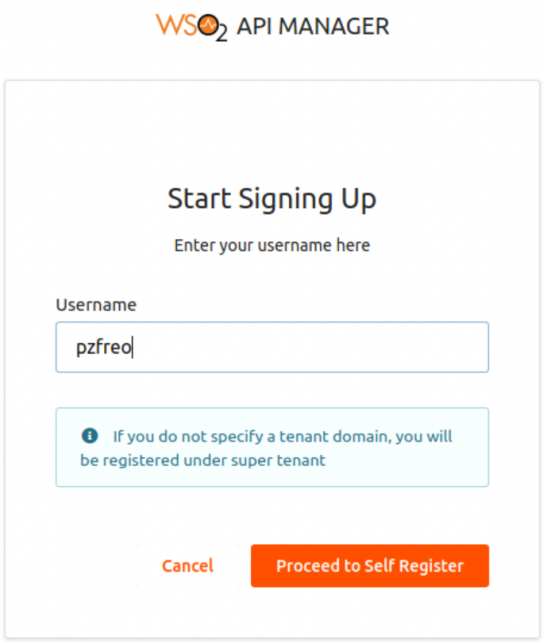
Take a look at these, but don't change anything just yet!

15) Now click View in Dev Portal (on the Overview Tab)
You should see:

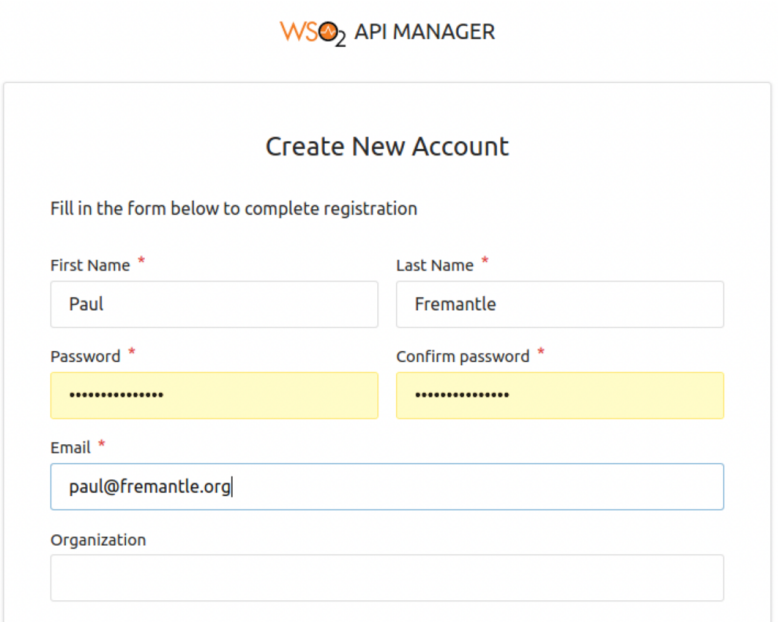


This is the view that an external/third-party developer will see. It can of course be customized.

16) You can continue as admin, but it would be nice to do this "properly", where you treat the subscriber as a different user. To do this, **Logout, Sign In,** then **Create Account.**



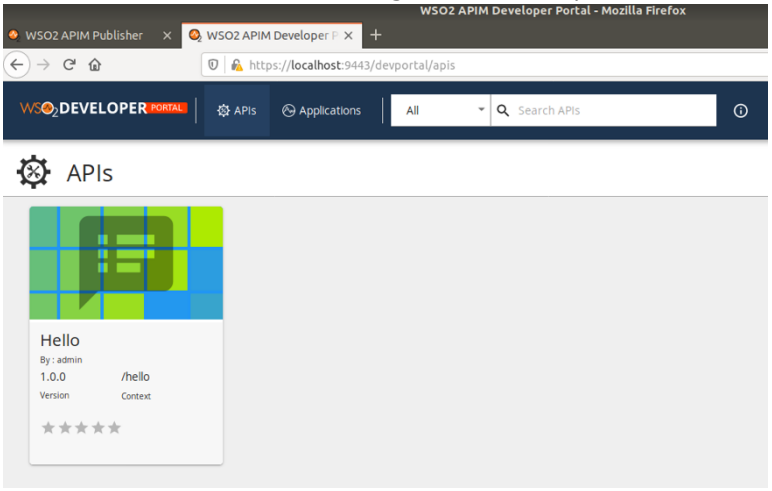
Choose a username (don't use an email address as that is based on the multi-tenancy support). Then fill in your details.



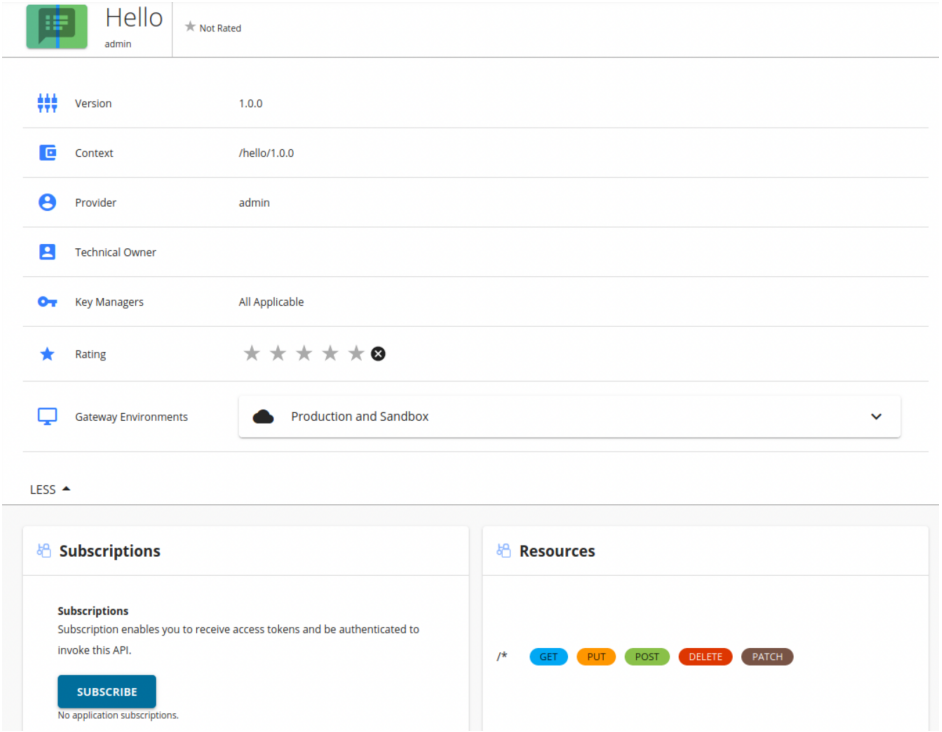
PS the system can also be configured to use Github/Twitter/Google credentials using an OAuth2 flow.

17) Now sign in using your new credentials.

18) You should now see the API once again in the Store/Portal:

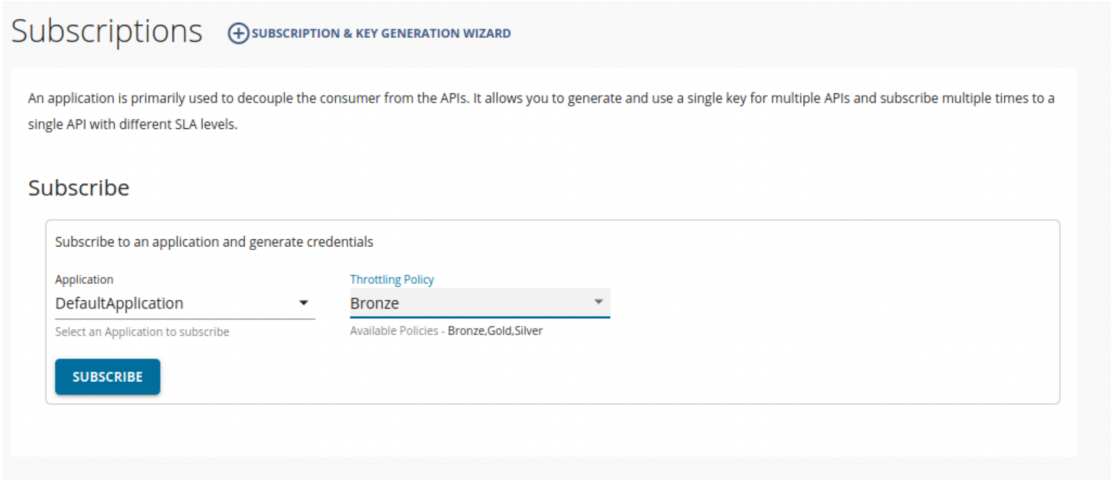


19) Click on the **Hello** API.



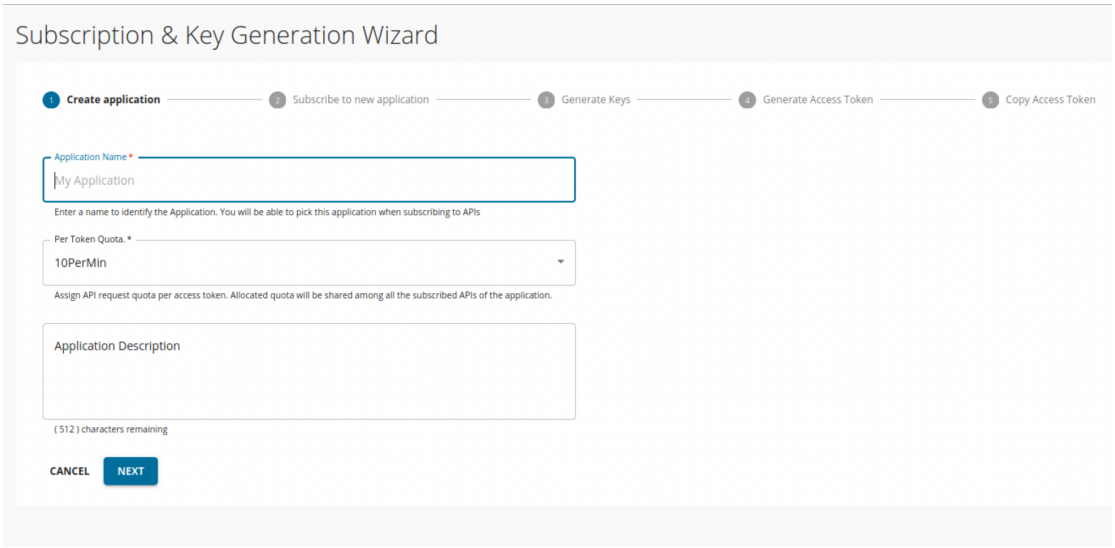
20) Now we need to subscribe to this API. As a developer you are going to be creating an application that calls this API. Logically speaking it is the application that is subscribed – not the developer.

21) Click **Subscribe**

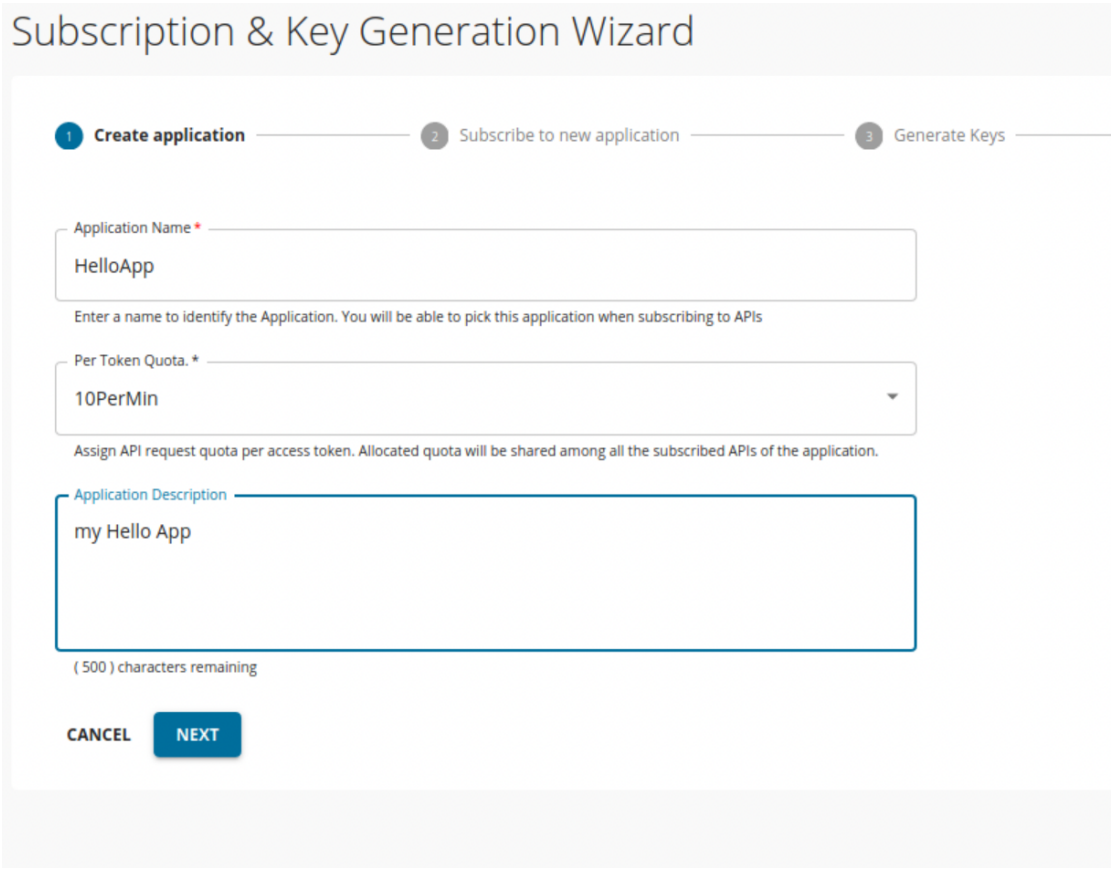


22) Click **Subscription & Key Generation Wizard**

23) You should see:

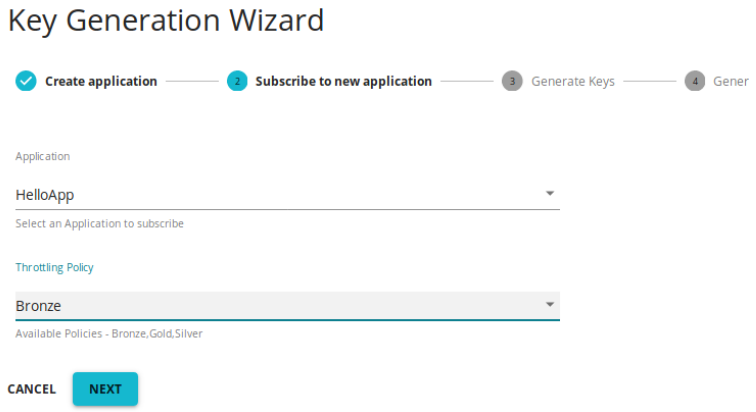


24) Fill in the Form a bit like this:



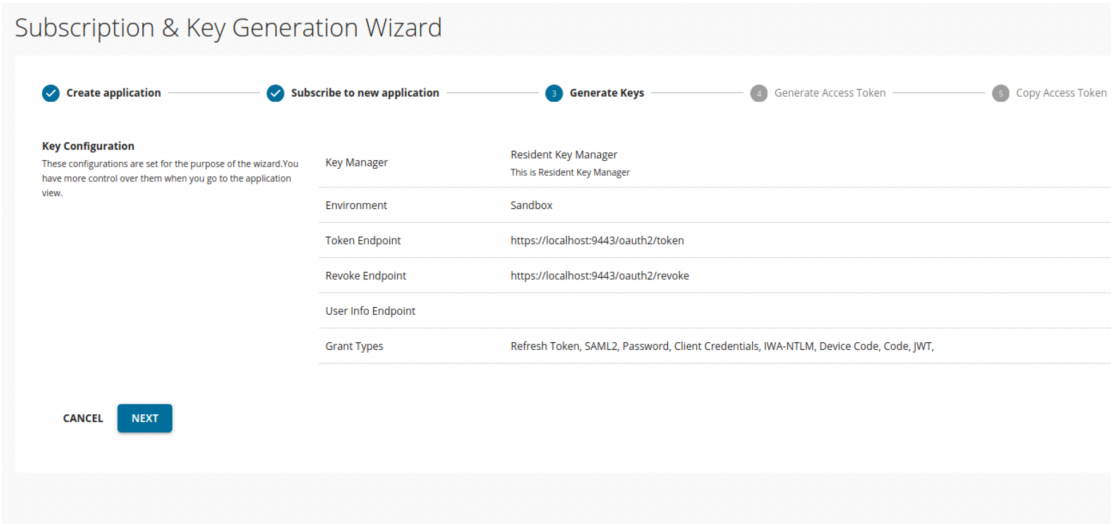
Click Next

25) Now you can choose a subscription level



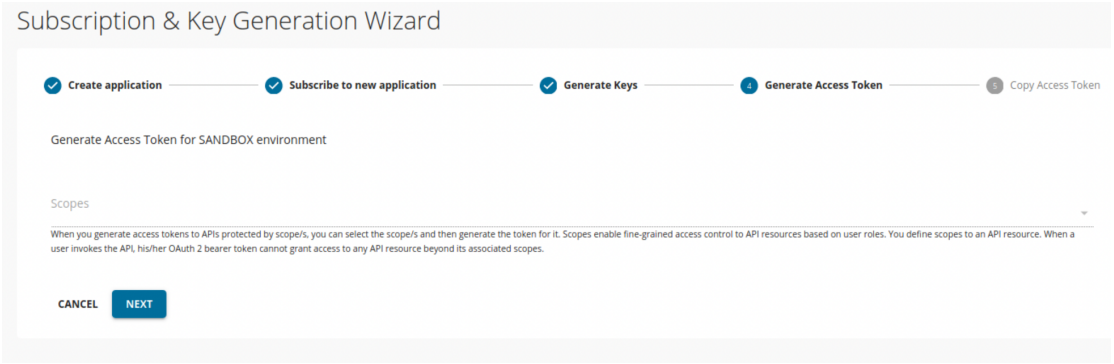
Click Next

26) Now you can choose to generate keys.



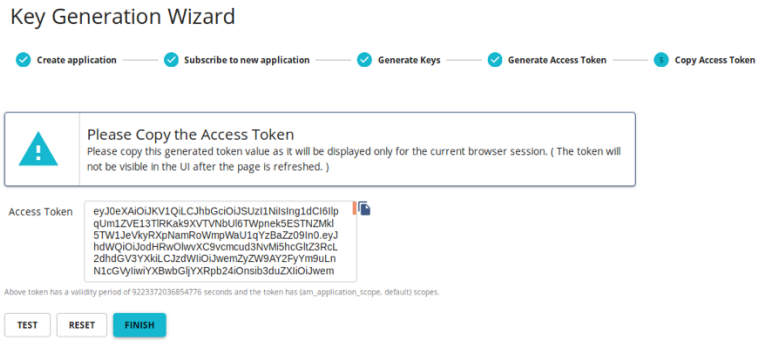
Click **Next**

27) You should see



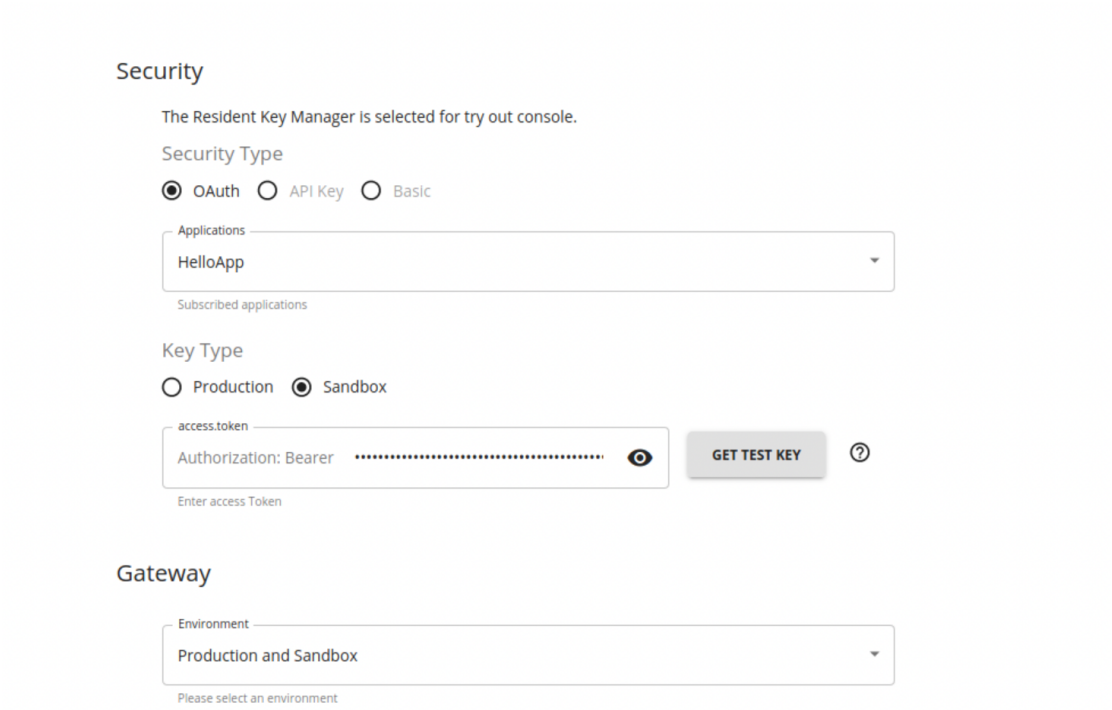
Click **Next**

28) Now you have a token to access the API gateway. For some reason, I found the little "copy" icon did not work for me on Ubuntu, so manually copy the key to your clipboard.



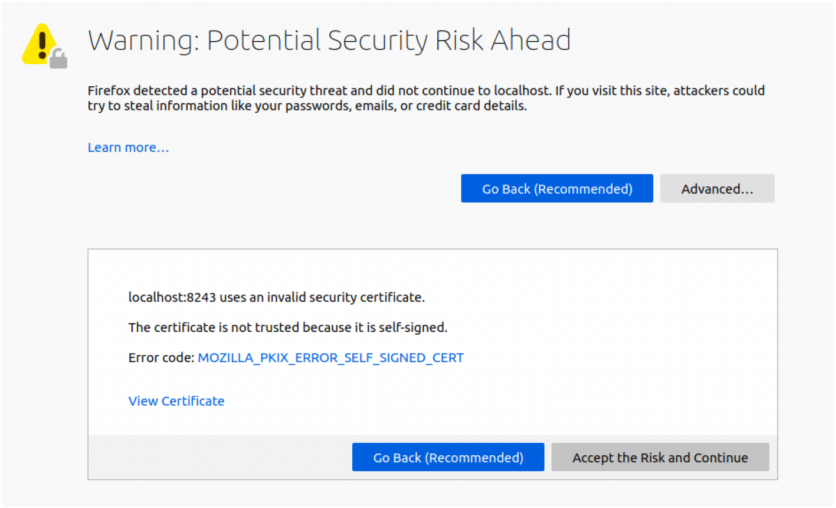
Click **Test**

29) Paste the access token into the right place:



30) Before we test it, we need to sort out a potential problem. The Gateway for this service is running on port 8243 with TLS. Because we haven't yet accessed that port, Firefox doesn't know about the self-signed certificates and won't let us access it. In production these would be replaced by real SSL certs.

In another browser window or tab, go to https://localhost:8243
As before, accept the risk:

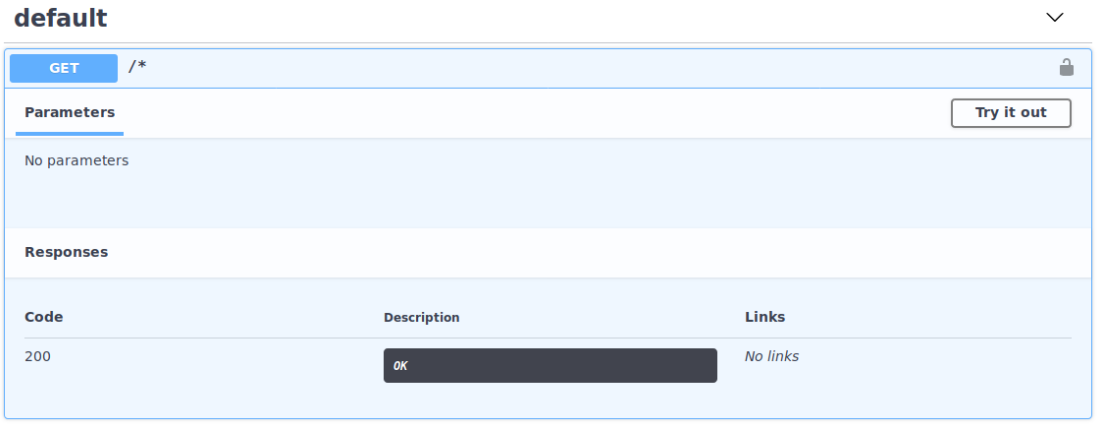


You should see:

Welcome to APIM

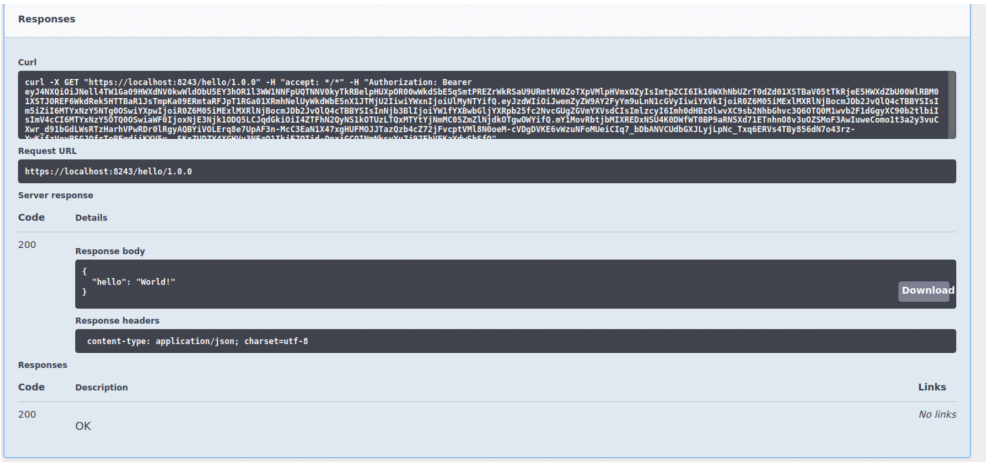
31) Close that window or tab.

32) Now expand the GET box by clicking on the **GET** button.



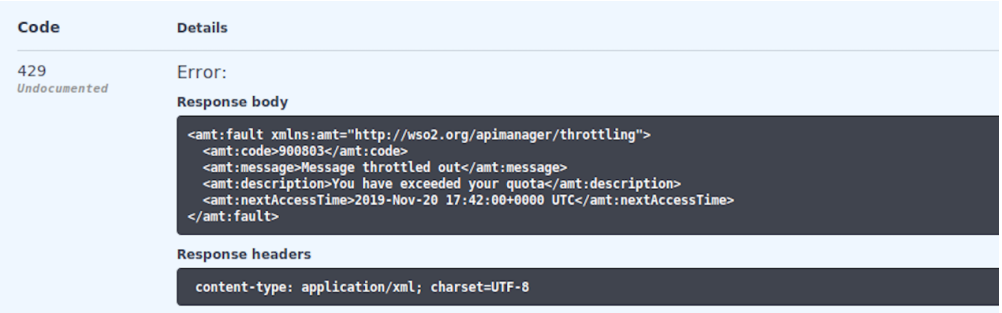
Click **Try it Out**

33) Click **Execute**



34) Click on **Execute** a few more times to generate a little bit of data for the analytics.

35) Keep clicking and eventually you should see the throttling happen:



36) To summarize what you have done is to create a "managed API" that is being controlled by the API gateway, and published in the API store. We will shortly create another, but first, let's explore this a bit more.

37) As the user of the API you can see some analytics about your own usage. We've been collecting the data, but before we can look at it, we need to start up the **Dashboard** server.

38) Start a new terminal window:

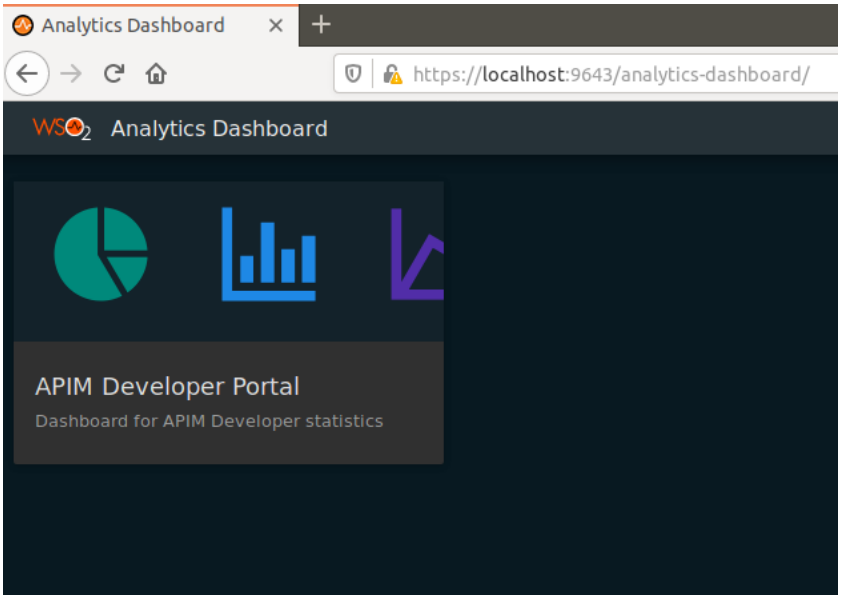
```
cd ~/servers/wso2am-analytics-3.2.0/
bin/dashboard.sh -run
```

39) Let's look at the dashboard page: https://localhost:9643/analytics-dashboard

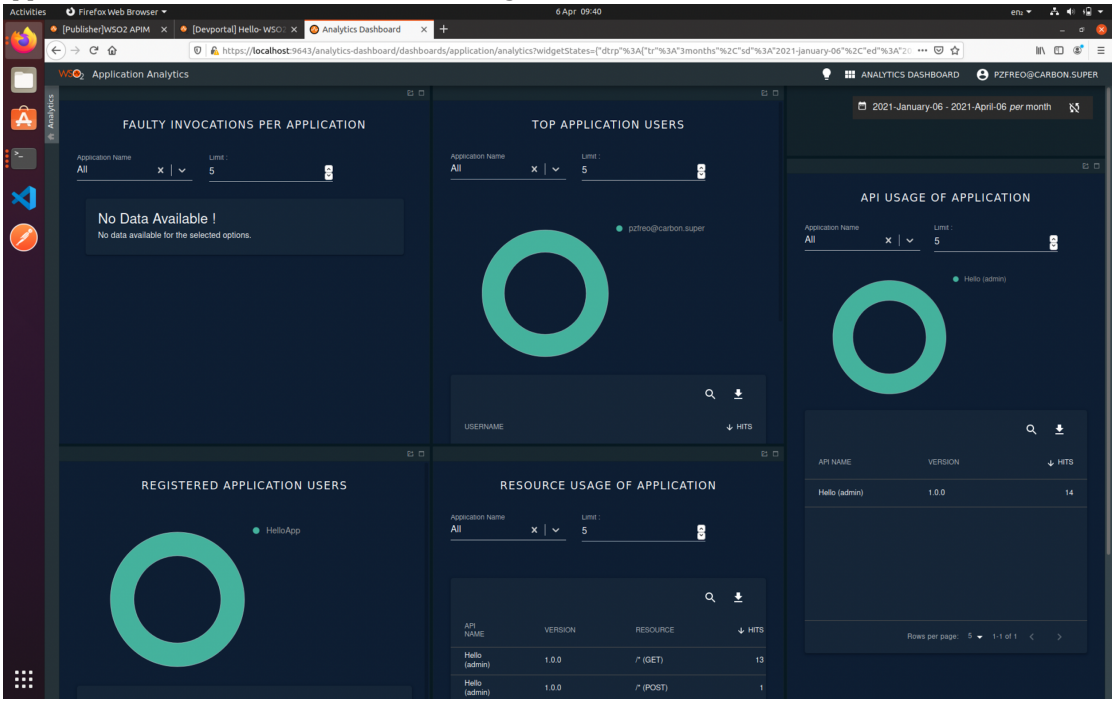
Go through the certificate approval again.
You will be prompted to login. You can login as either your subscriber or your publisher (admin).

Login as the subscriber (not admin) first.

You should see something like:

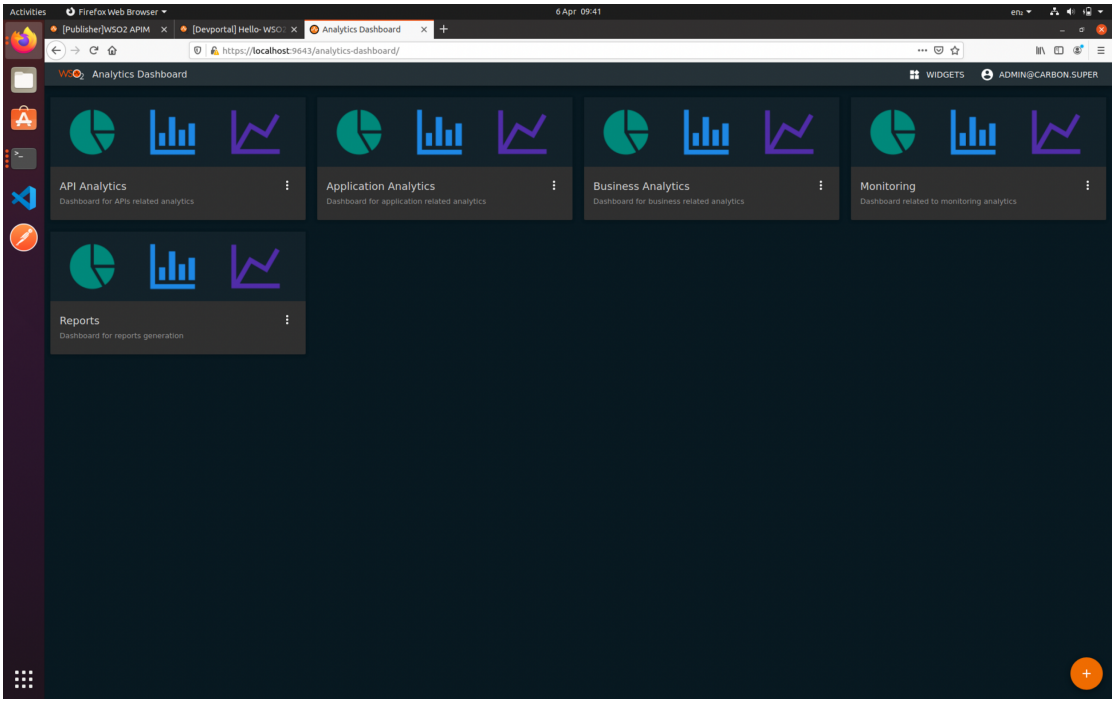


These are dashboards aimed at developers – allowing them to see the usage of APIs by their applications. Take a look at the available widgets.

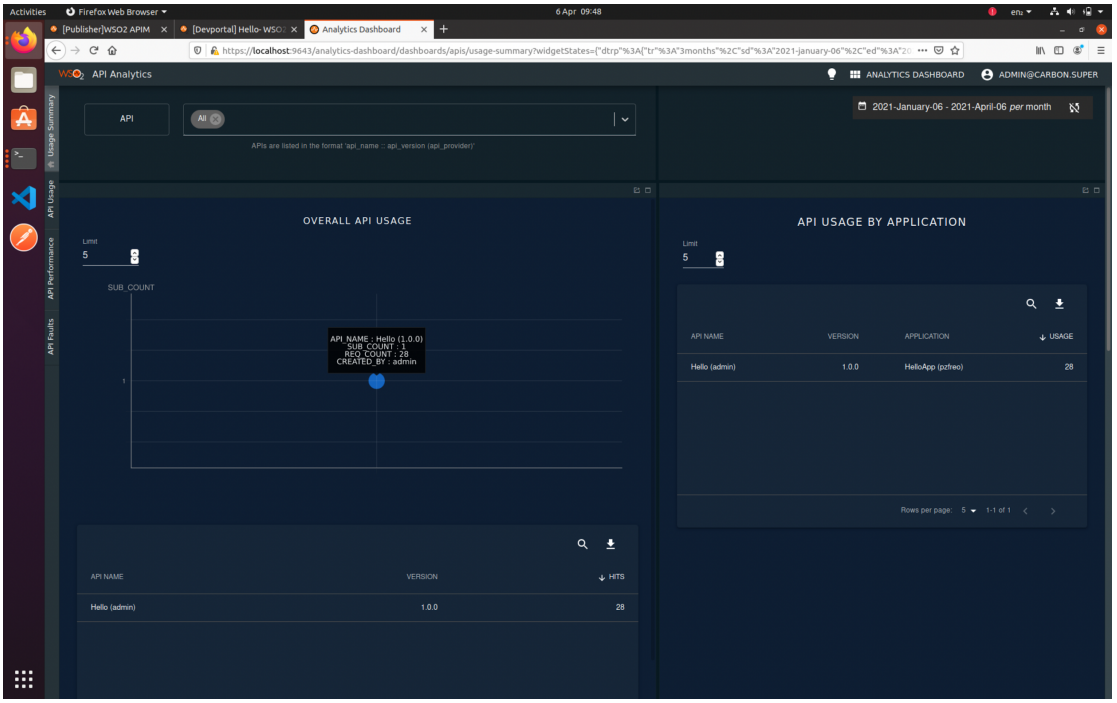


40) Now logout and log back in as **admin**.

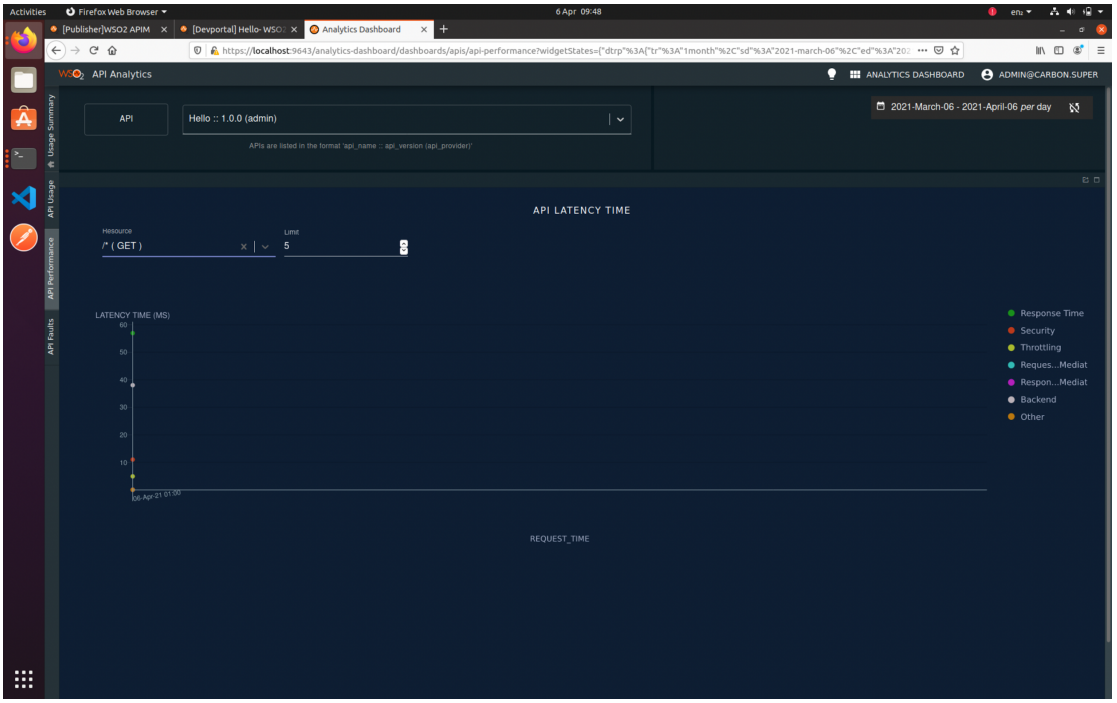
41) Now you should also get access to the Publisher's dashboard:



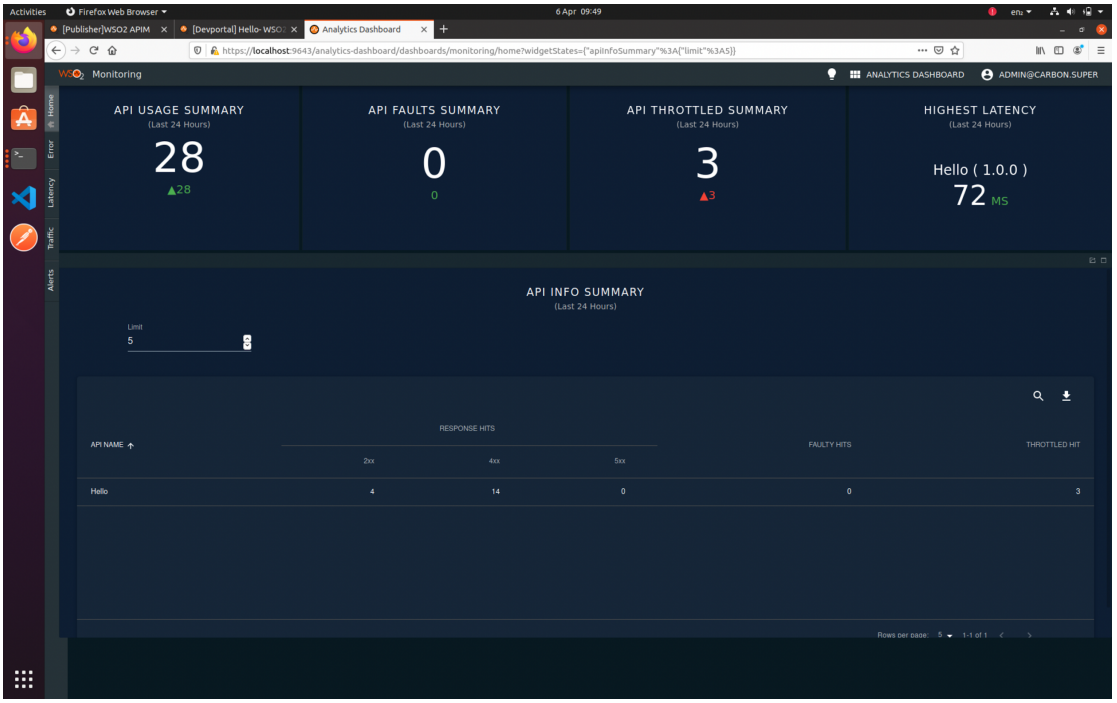
42) Explore the various analytics pages. To be honest there isn't enough data to be very exciting!



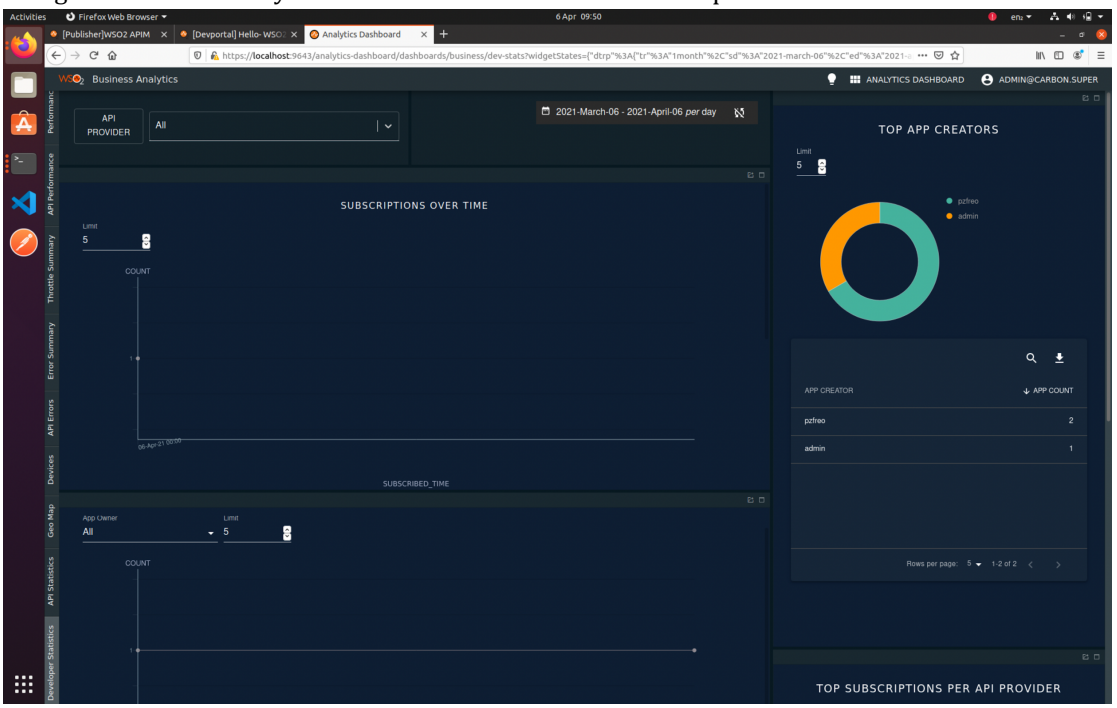
43) For example, you can view the latency of different aspects of the API:



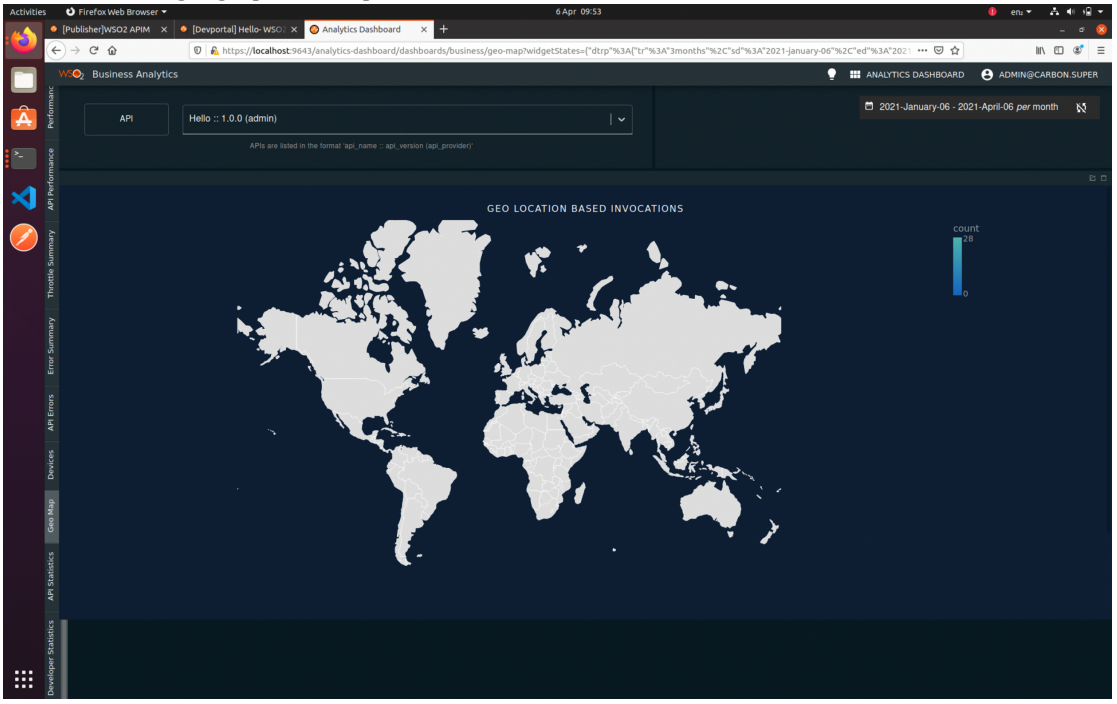
Also take a look at the monitoring page:



Also go to Business Analytics and at the bottom tab find Developer Statistics:



There is even a geographical map of invocations:



In the interests of saving memory I suggest you kill the dashboard server before continuing.

Part B - Managing our Purchase API

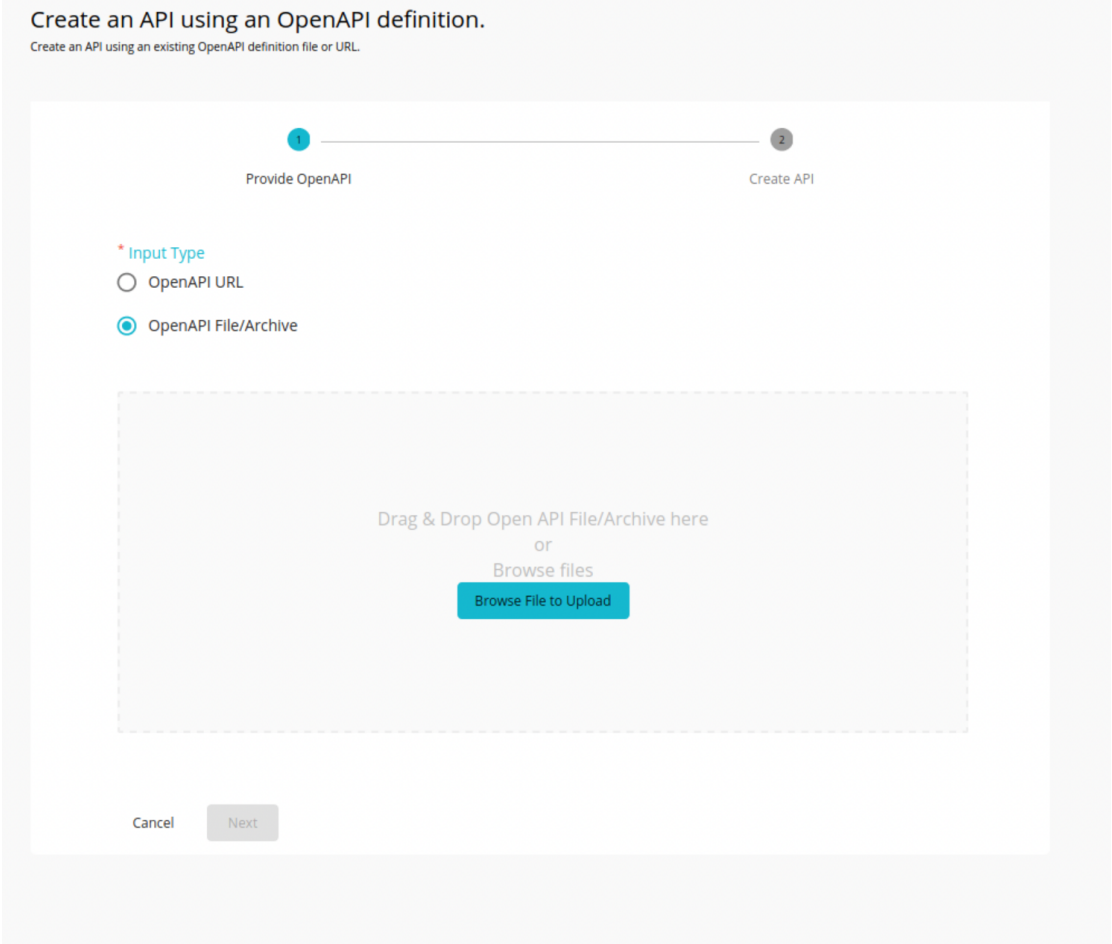
44) Firstly, make sure you have the Purchase API running from the Swagger/OpenAPI exercise.

    We DO NOT want the version from the OAuth2 exercise as that is expecting the OAuth2 token in the backend whereas we are going to handle this in the gateway. If necessary, you can comment out the lines relating to the @pzfeo/express-introspect package.

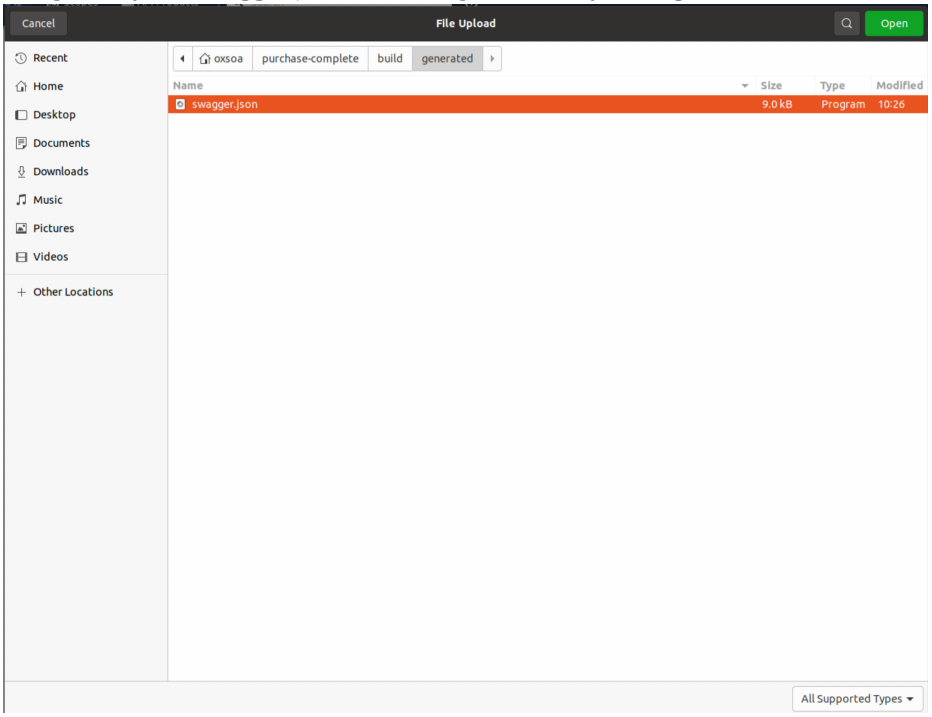
45) Go back to the API Publisher tab.

46) Click on **APIs**, then **Create API**

47) Click "**I have an existing REST API**"



48) Use **OpenAPI File**
Navigate to your **swagger.json** that was generated by tsoa spec-and-routes



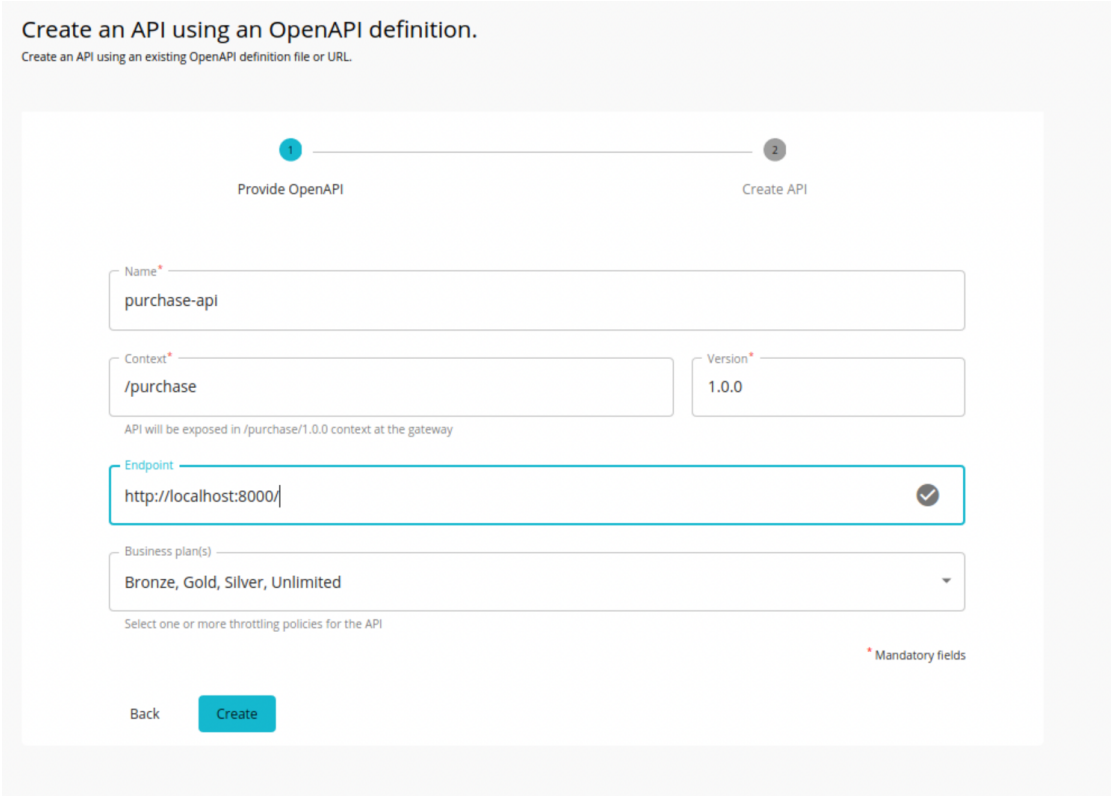
49) Click **Next**

50) change the name to Purchase-API

51) Set the context to be **/purchase**

52) Use the purchase server's URL for the Production endpoint. Use:
http://localhost:8000/

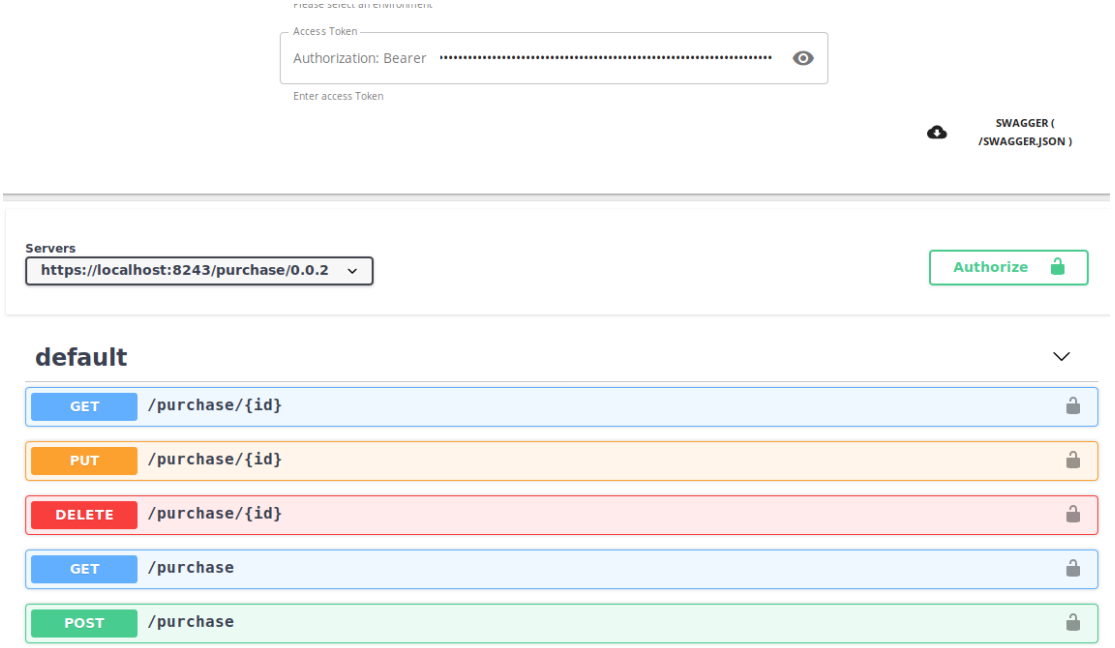
53) Enable all the potential business plans.



54) Click **Create,** then **Publish**

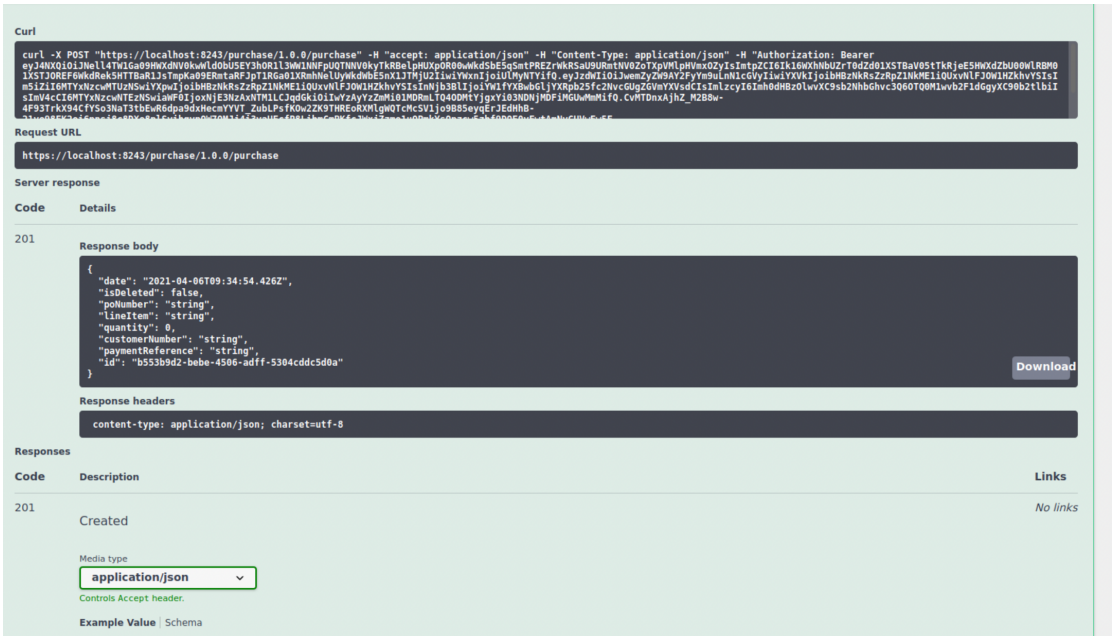
55) Once again subscribe to this API as before, but with the Gold subscription.

56) You should be able to see the nice Swagger generated API console

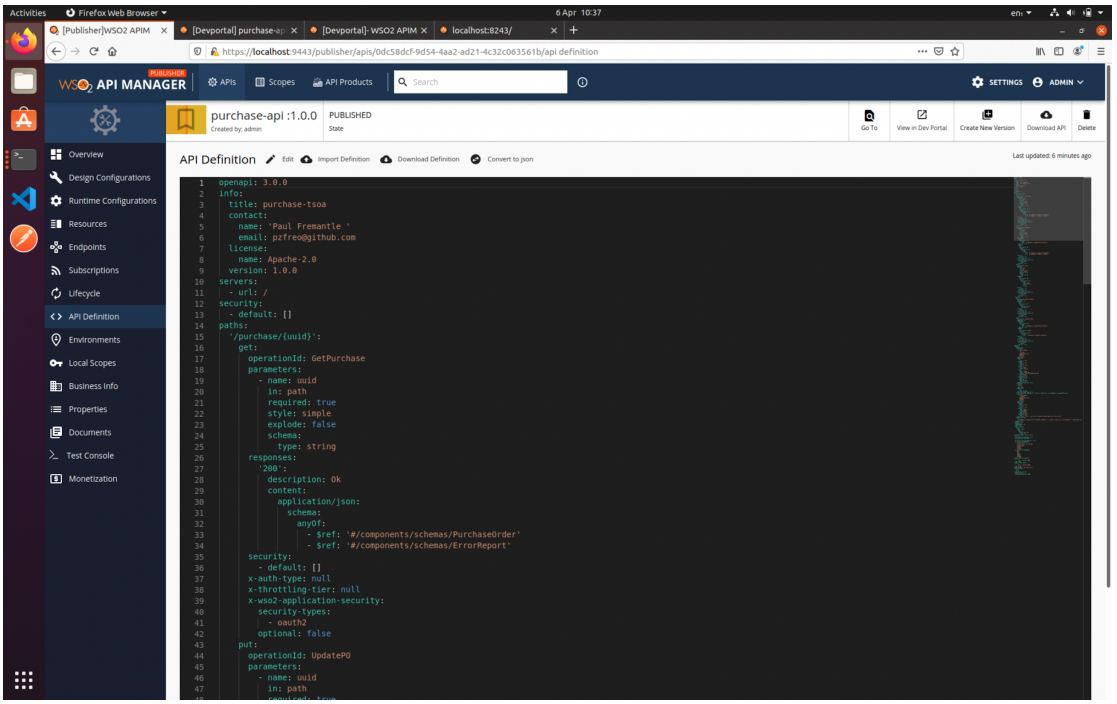


57) Try it out.

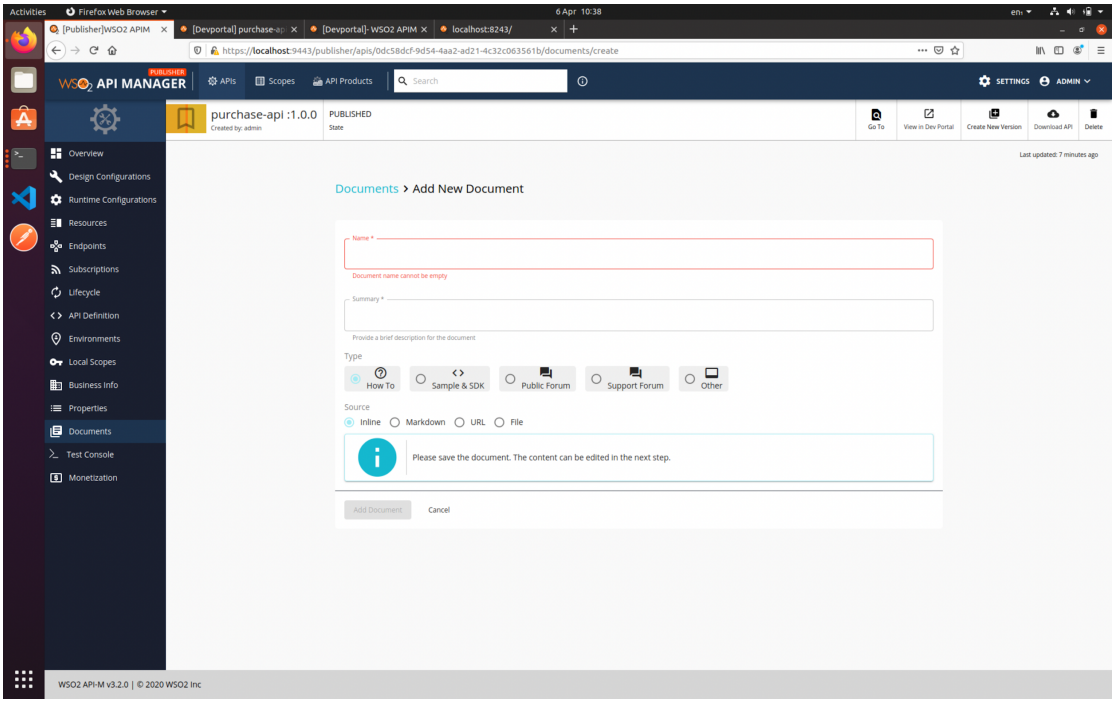
Notice how the Swagger user interface has once again helped us so that developers can see what the interface is like.



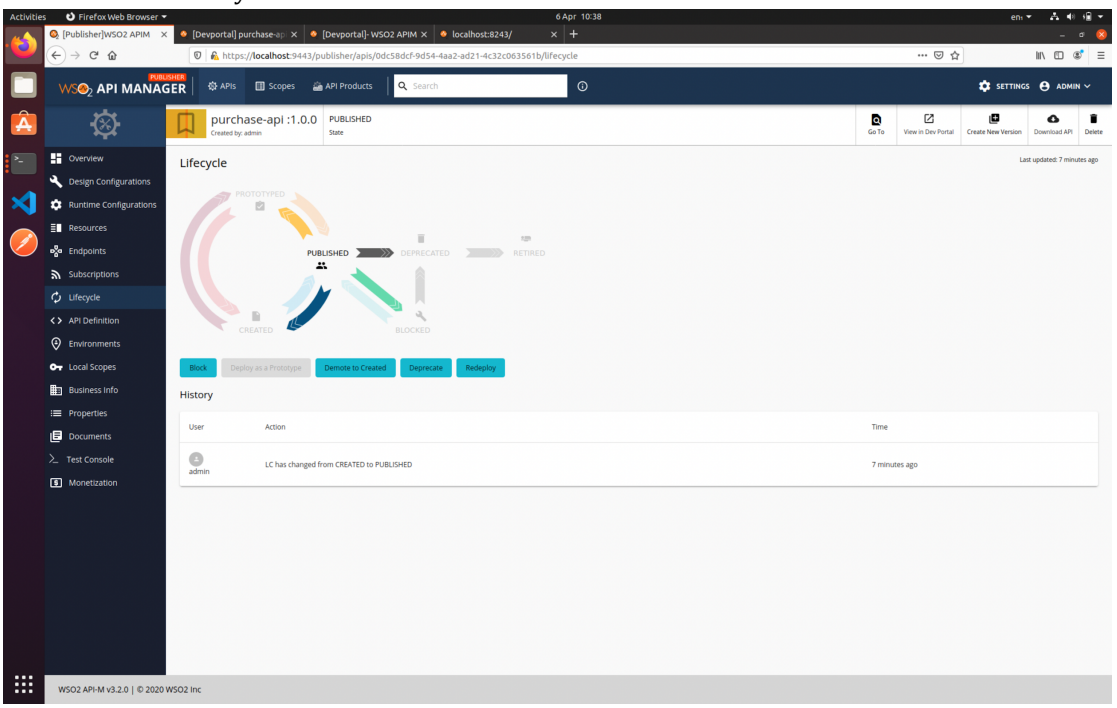
58) Take a look at the API in the published. Notice that the APIM has converted the JSON Swagger into YAML as well:



59) You can also add additional documentation:



60) Also look at the Lifecycle:



61) That's the main lab finished.

62) Close down all the servers to free up memory (unless you are doing extensions)

Extensions

1. Try calling the Purchase API from Postman. Copy the URL from the API Store and remember to add the Authorization header.

2. There are lots more things to try. For example, see if you can Create New Version.

3. Use wrk to generate enough traffic to kick in the throttling

4. Check out the analytics once you've done that.