# Exercise 13

*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:
```
cd ~/servers
unzip ~/Downloads/wso2am-3.0.0.zip
unzip ~/Downloads/wso2am-analytics-3.0.0.zip
```

2) Now enable analytics:
```
code wso2am-3.0.0/repository/conf/deployment.toml
```

Go to the **Analytics** section of the TOML file on **line 79.**
**Uncomment the whole block (down to line 86).**
Change enable from **false** to **true**, then save.

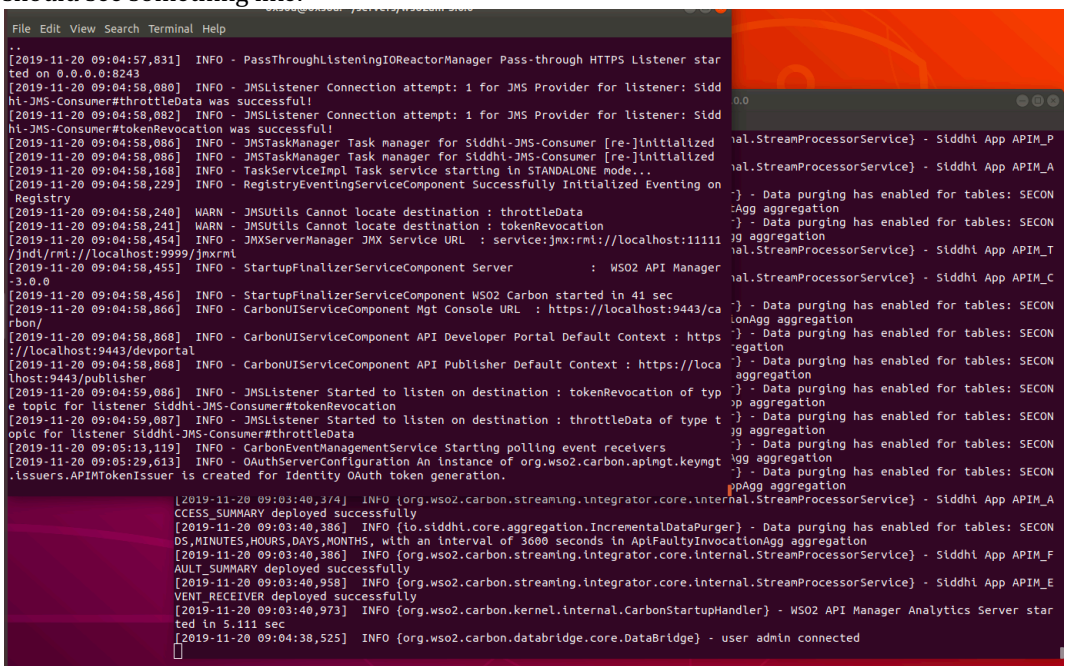
It should look like this:
```
78
79  [apim.analytics]
80  enable = true
81  store_api_url = "https://localhost:7444"
82  username = "$ref{super_admin.username}"
83  password = "$ref{super_admin.password}"
84  event_publisher_type = "default"
85  event_publisher_impl = "org.wso2.carbon.apimgt.usage.publisher.APIMgtUsageDataBridgeDataPublisher"
86  publish_response_size = true
87
88  #[[apim.analytics.url_group]]
```

3) From a fresh terminal window or tab start the WSO2 API Manager Analytics:
   a. `cd ~/servers/wso2-am-analytics-3.0.0`
   b. `bin/worker.sh --run`

4) The API Manager uses its own internal AMQP server, also on port 5762, so first stop RabbitMQ:
```
sudo service rabbitmq-server stop
```

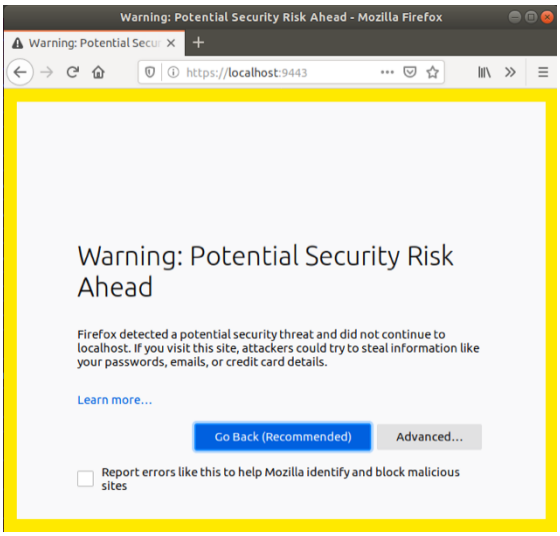
5) Then in another terminal window start the WSO2 API Manager:
   c. `cd ~/servers/wso2am-3.0.0`
   d. `bin/wso2server.sh`

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




7) Once both servers are started, check that you can access the web interface of the API Manager

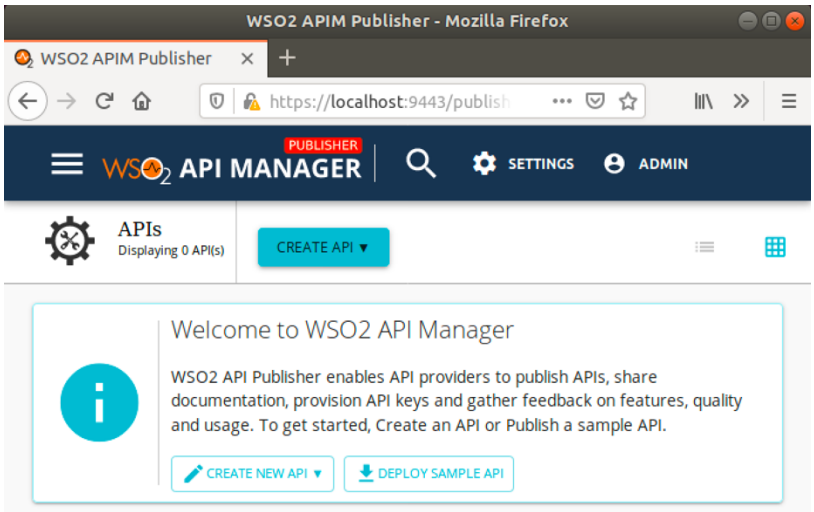
   e. https://localhost:9443/ (AM console)
   If this is the first time you try this server you may need to allow the self-signed certificate. **Advanced,** then **Accept the Risk and Continue.**



8) You should see something like this:


9) Log in as **admin/admin**

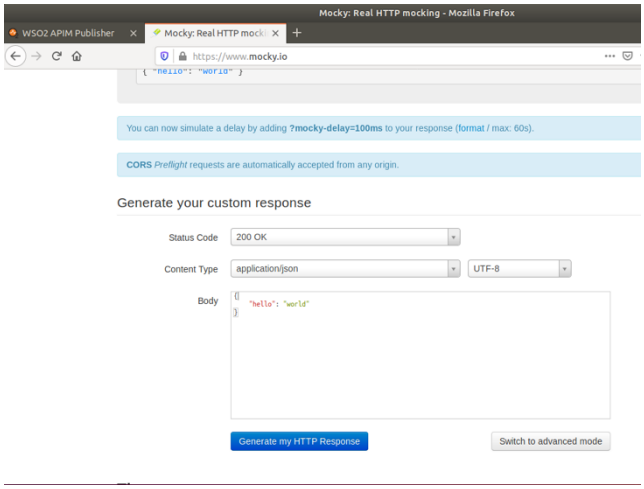
You should see:



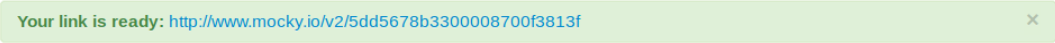

10) An API creator uses the **API Publisher** system to create and publish APIs into the **API Store (Developer Portal)**. Firstly we will follow the tutorial to create a managed API. Then we will try it out using the API Store.

11) Let's create a simple "mock" API to use as the backend for this.
Go to https://www.mocky.io

12) Edit the response Body to be: `{ "hello" : "world" }`
Like this:

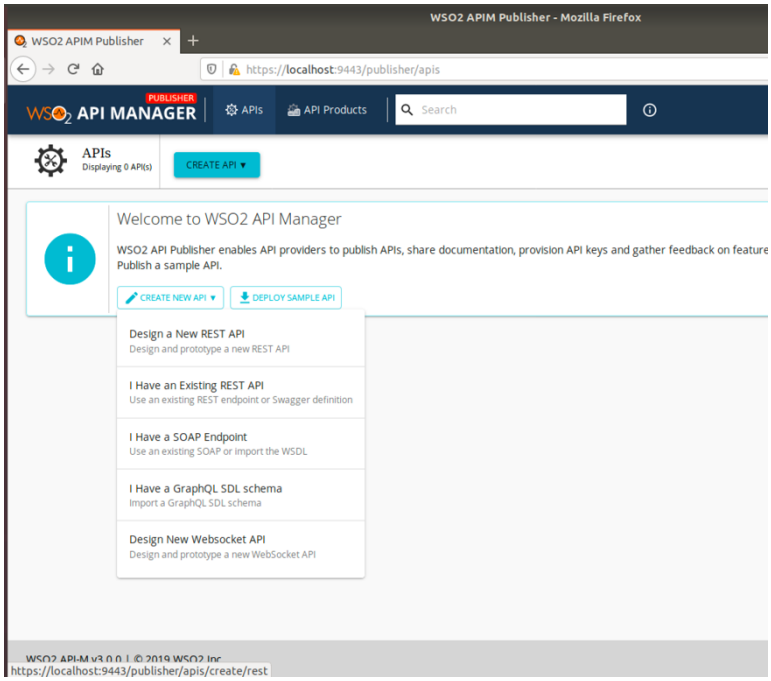



13) Click **Generate My HTTP Response**
This creates a new URI that acts as your API.
You should see something like:




Click on the link and you will see it simply returns the response you asked it to.

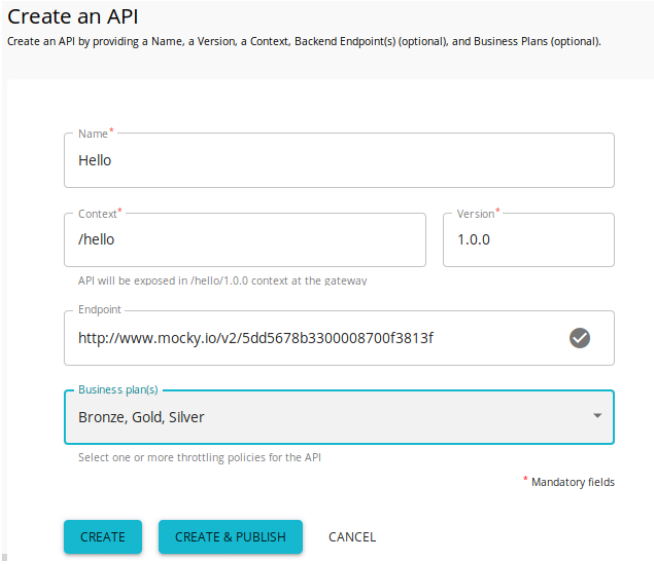
14) Back on the Publisher click the **Create New API** button. You will see:



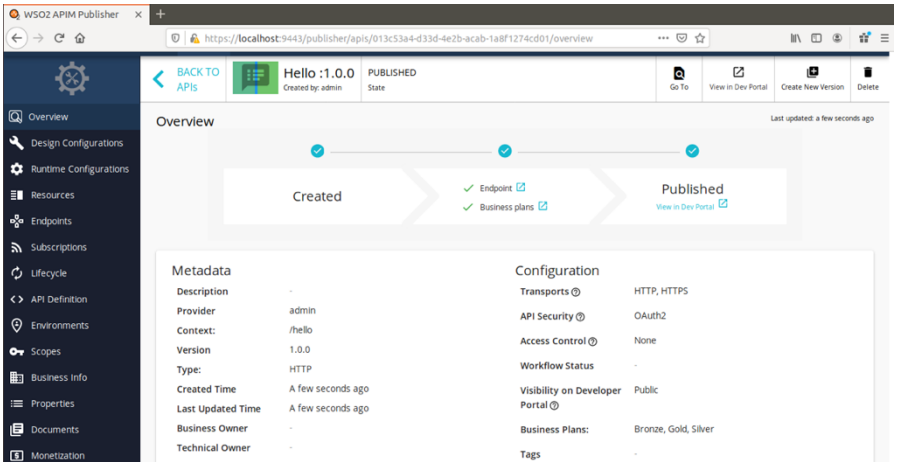
15) Click **Design a new REST API**

Fill in as follows:

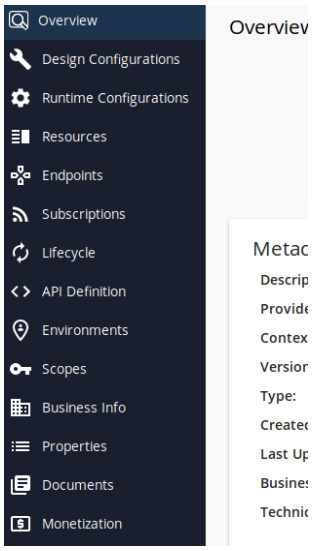
| | |
|---|---|
| Name: | Hello |
| Context: | /hello |
| Version: | 1.0.0 |
| Endpoint: | *your mocky URL* |
| Business Plans: | Choose Gold, Silver and Bronze |




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

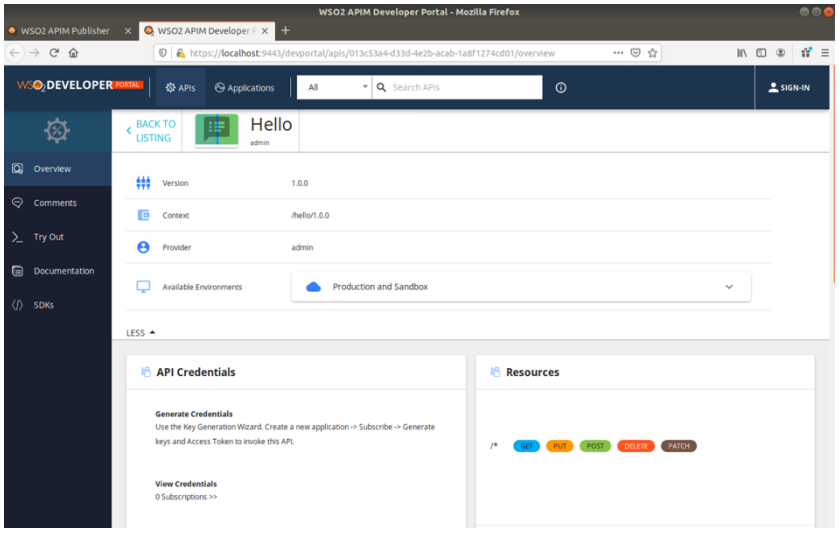


17) Notice that there is a lot more you can do with your API:




Take a look at these.

18) Now click View in Portal
You should see:




This is the view that an external/third-party developer will see. It can of course be customized.
For example, these portals are based on this technology:
https://developer.stubhub.com/store/
https://developer.wellsfargo.com/

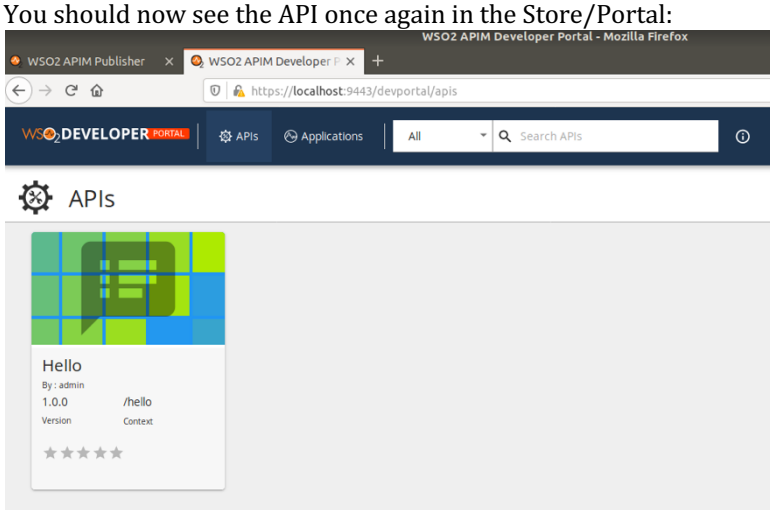
19) You can continue as admin, but it would be nice to do this "properly", where you treat the subscriber as a different user. To do this, **Sign In,** then **Register Now.**

20) Do the usual sort of thing.

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

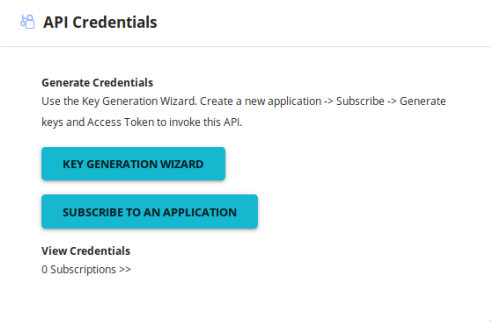
21) Now sign in using your new credentials.


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



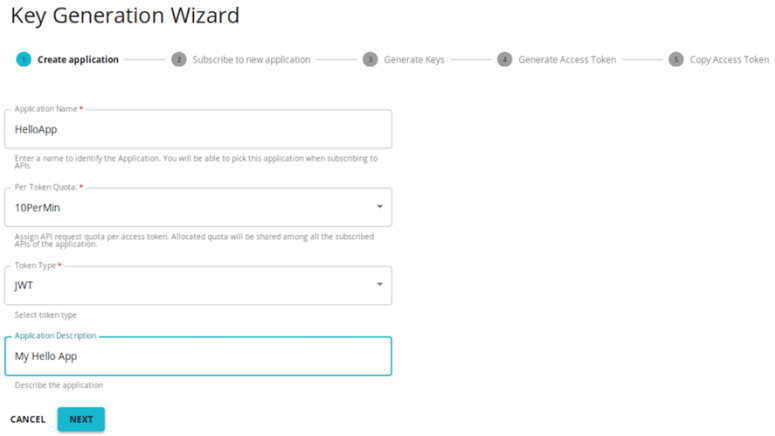

23) Click on the **Hello** API.

24) 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. So we will "register" our application to the portal and then subscribe it.



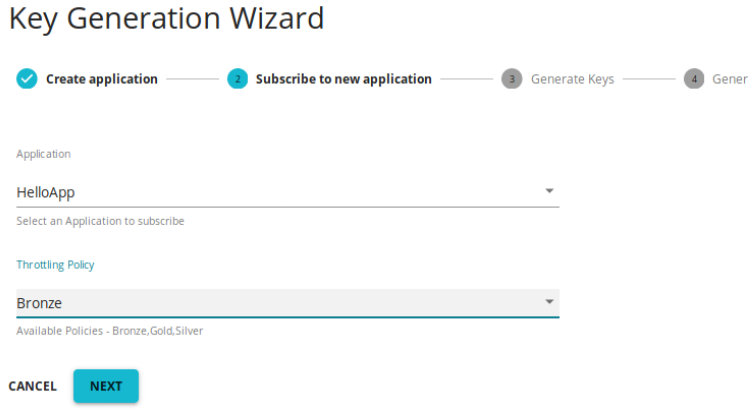

Click on **Key Generation Wizard**

25) Fill in the Form a bit like this:



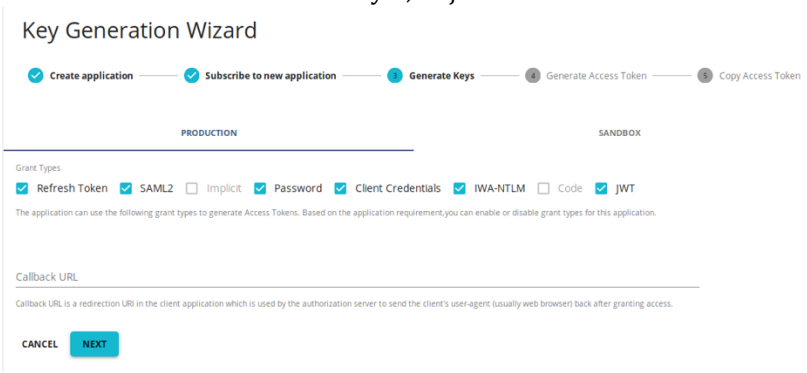

**Click Next**

26) Now you can choose a subscription level

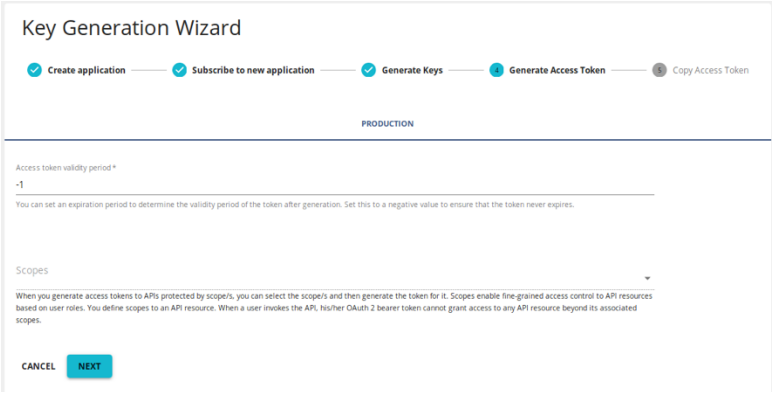



**Click Next**

27) Now you can choose to generate keys for either the Production or Sandbox. We haven't worried about a Sandbox server yet, so just click **Next.**

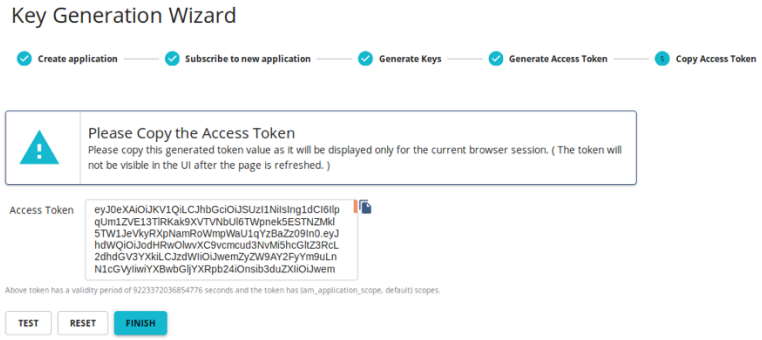


28) Set the access time validity period to -1 so they don't time out (since this is just for testing).



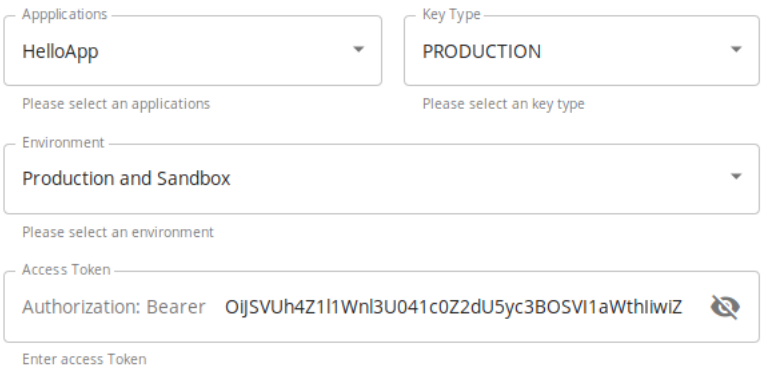

Click **Next**

29) Now you have a token to access the API gateway. Click on the little copy icon to copy it to your clipboard.

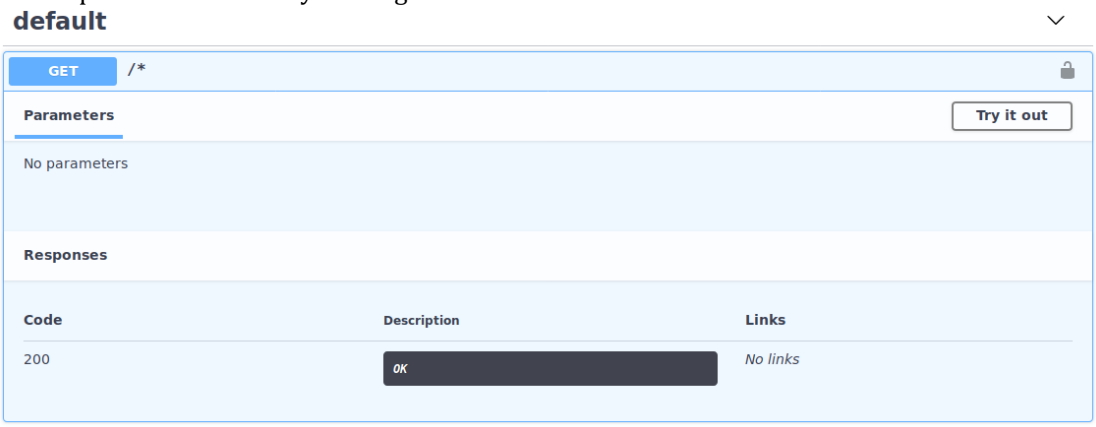



Click **Test**

30) Paste the access token into the right place:

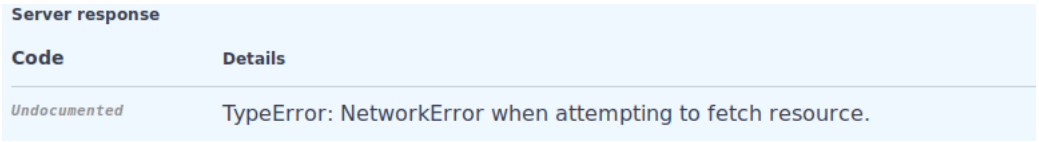



31) Now expand the GET box by clicking on the **GET** button.




Click **Try it Out**

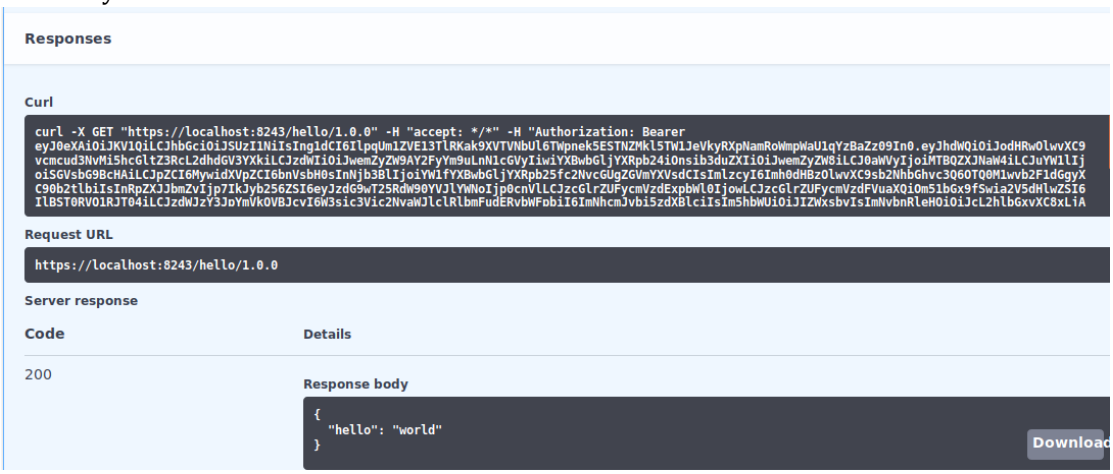
32) Click **Execute**




33) Unfortunately it will fail! Basically this is accessing a new SSL endpoint (https://localhost:8243) and the certificate is not trusted. Let's fix that.

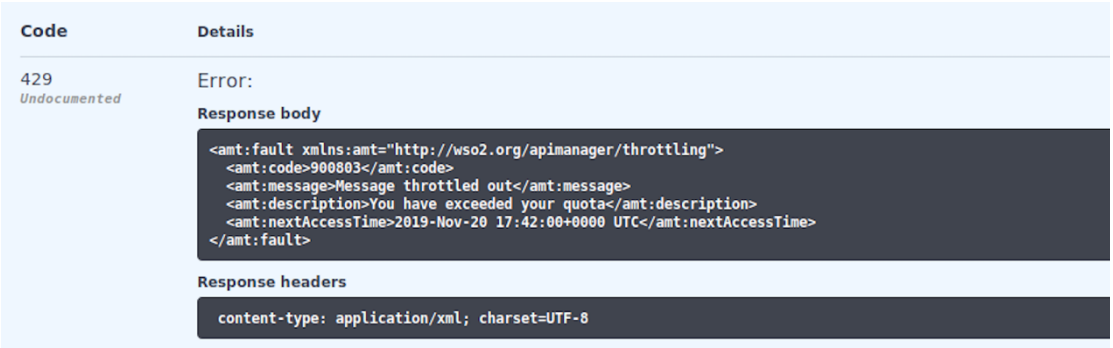
34) In a new window or tab, browse to https://localhost:8243
Once again you have the warning. Click **Advanced** and then **Accept the Risk**
It should say: *Welcome to APIM*

35) Now retry **Execute**. It should work now!




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

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




38) 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.

39) 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.

40) Start a new terminal window:

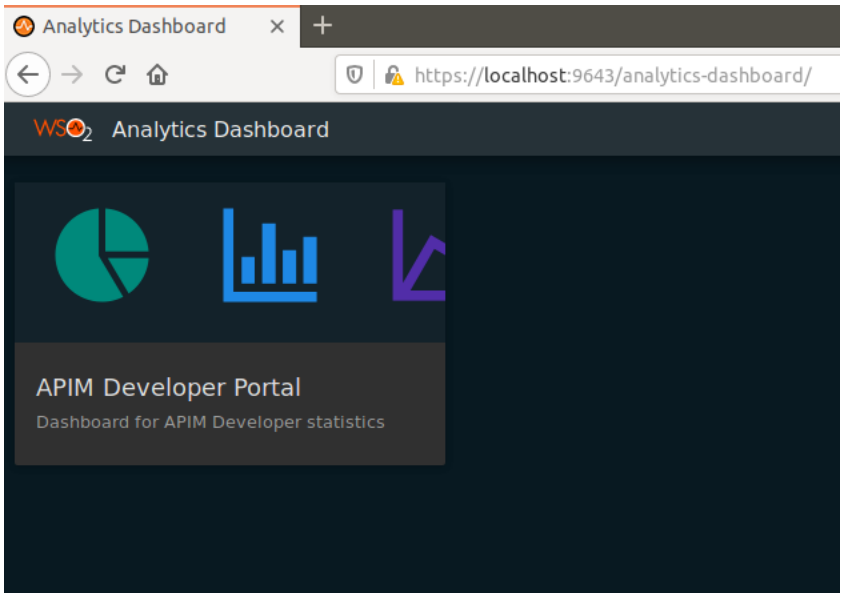
```
cd servers/wso2am-analytics-3.0.0/
bin/dashboard.sh -run
```

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

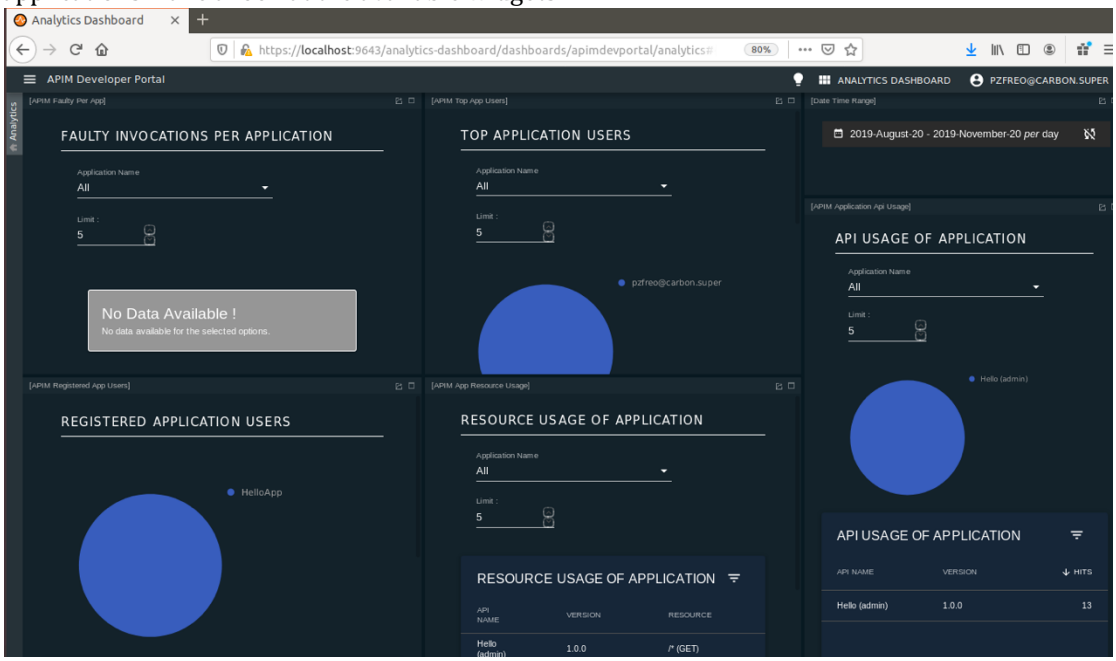
Go through the certificate approval again.
You will be prompted to login. You can login as either your subscriber or your publisher (admin).

Login as the subscriber (not admin) first.

You should see something like:

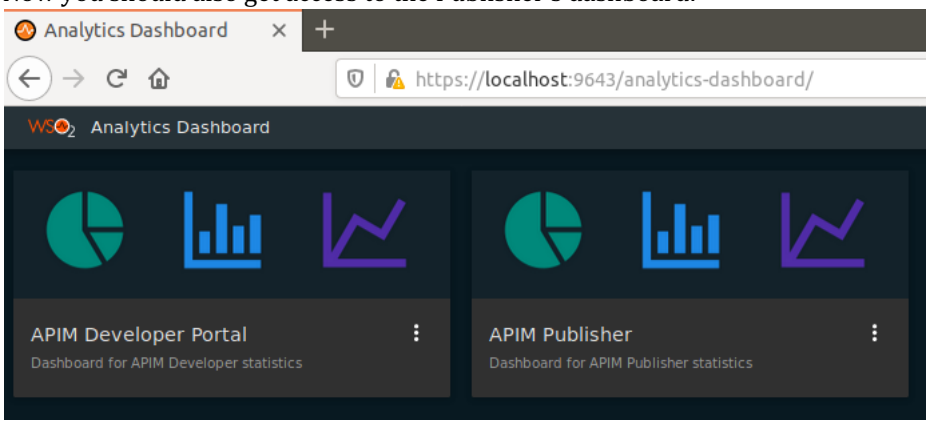



These are dashboards aimed at developers – allowing them to see the usage of APIs by their applications. Take a look at the available widgets.



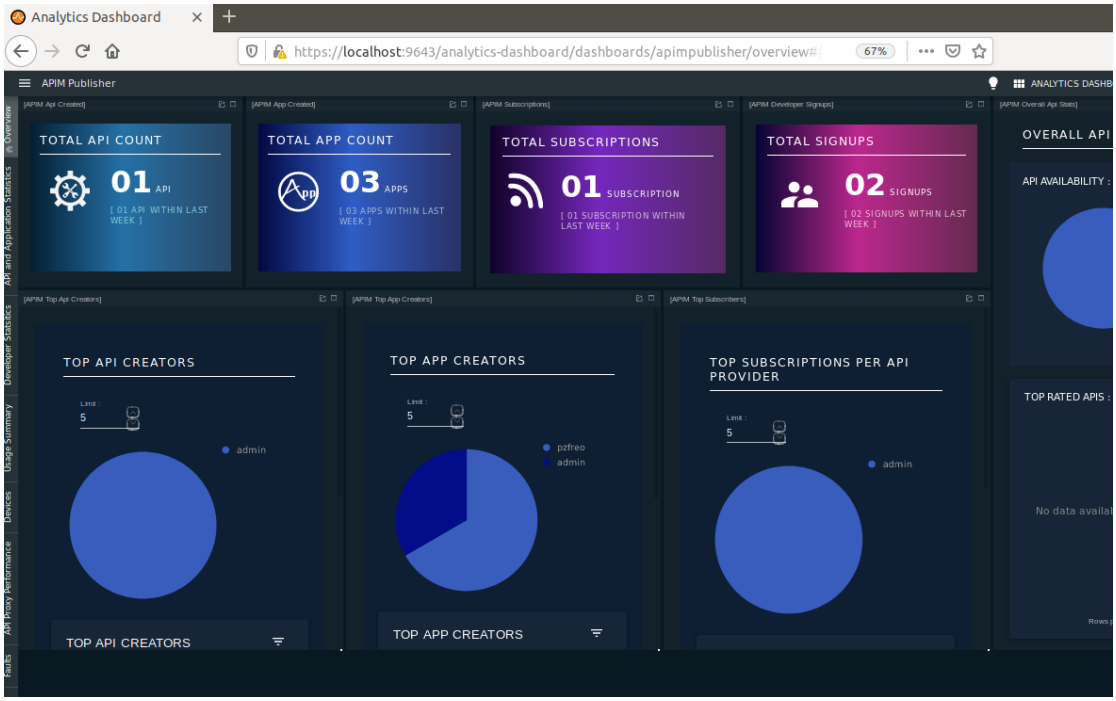

42) Now logout and log back in as **admin**.

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

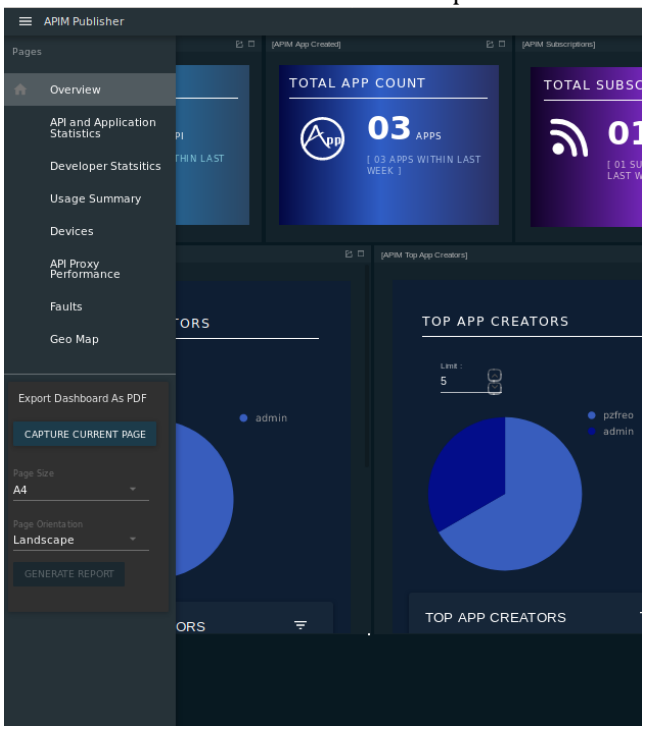



44) Click on APIM Publisher

45) It should look like this:



46) Notice that there is a menu with more options:




Have a good look at the various widgets and different views.

You can see how successful your API system is, including signup data, applications created, subscriptions, etc.

**Part B - Managing our Purchase API**

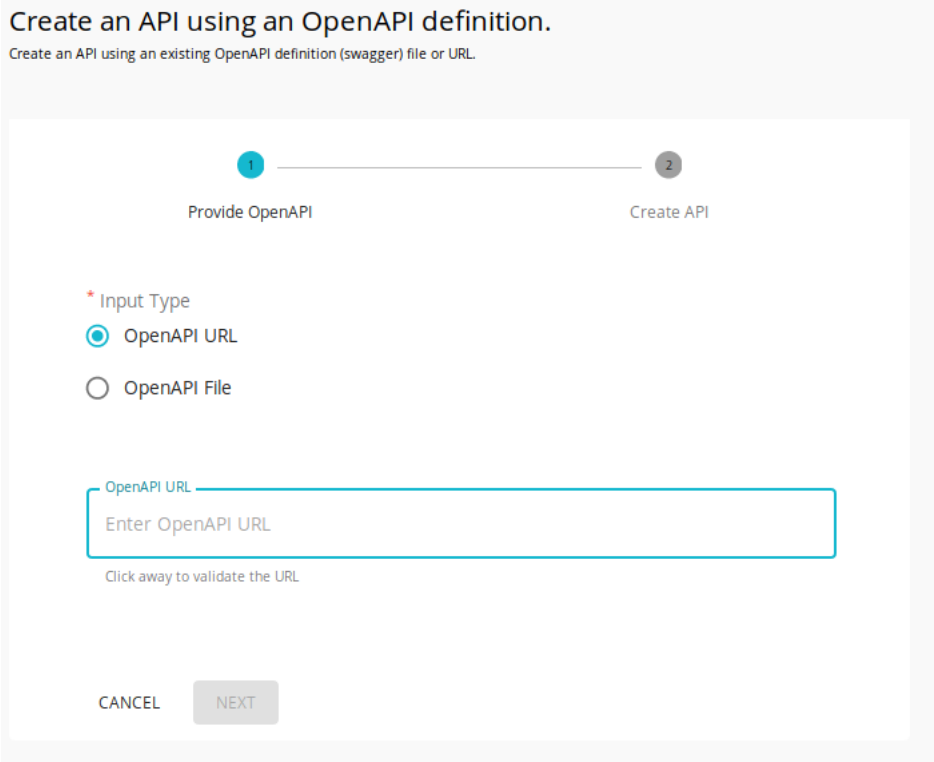
47) Firstly, make sure you have the Purchase API running from Exercise 8.

> *If you do not or you have problems:*
>
> ```
> sudo service redis-server stop
> git clone https://github.com/pzfreo/PSBComplete.git
> cd PSBComplete
> gradle clean build -x test
> docker-compose up --build
> ```
>
> Check it is running:
> ```
> curl http://localhost/purchase
> ```

48) Go back to the API Publisher tab.

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

50) Click "**I have an existing API**"



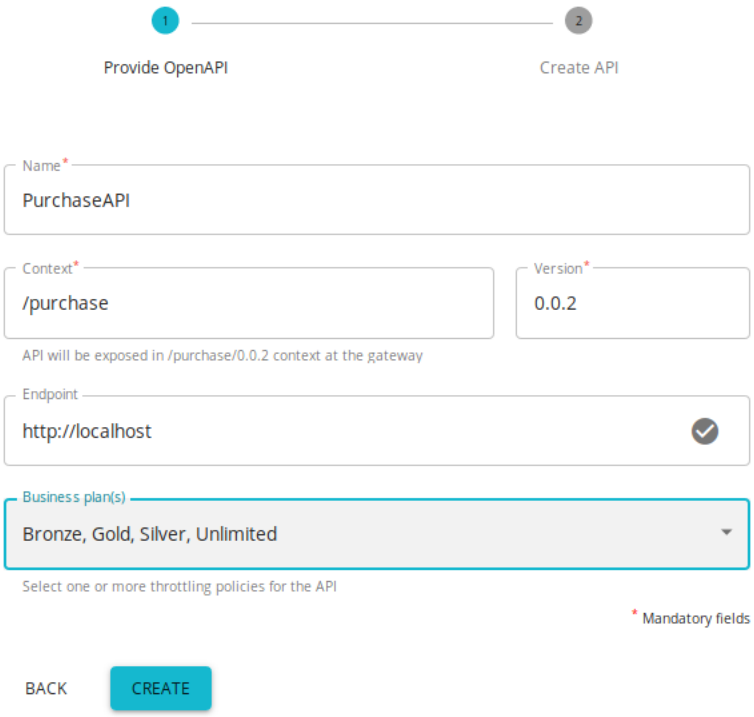

51) Use **OpenAPI URL**
**http://localhost/openapi.yaml**
then tab away from the box to validate the URL
You should see a nice tick:


52) Click **Next**

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

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

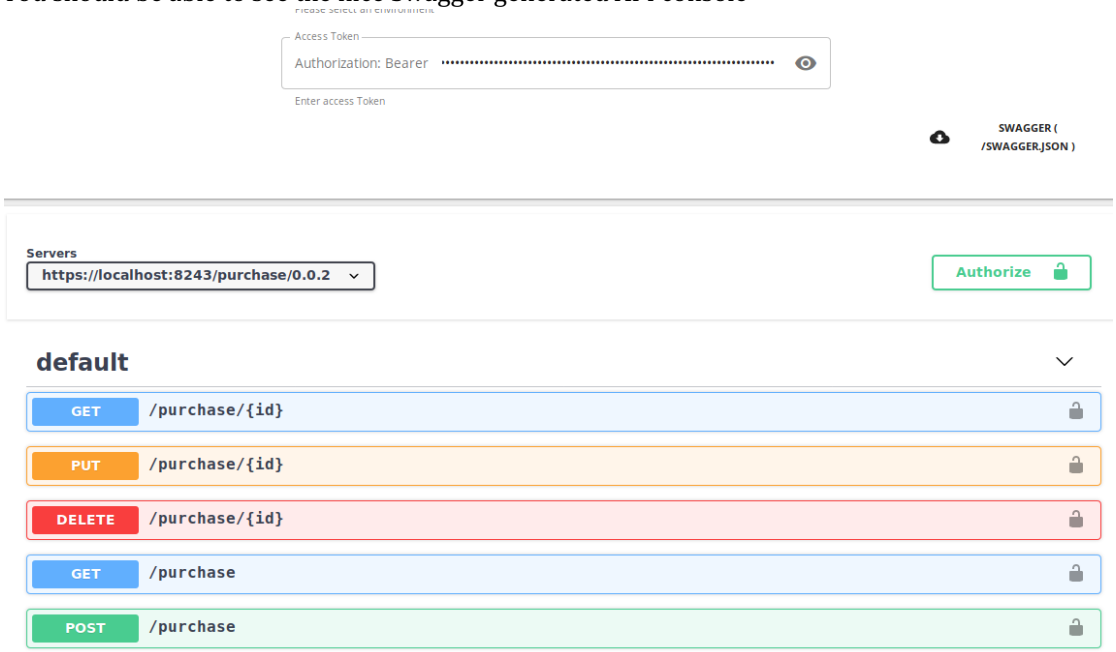
55) Enable all the potential business plans.




56) Click **Create,** then **Publish**

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

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




59) Try it out. That's the main lab finished.

**Extensions**

**1.** Try calling the Purchase API from the Advanced REST Client. 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 Copy your existing API into a new version and publish that.

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

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