Exercise 13

API Management and Governance including Analytics

Prior Knowledge
RESTful services

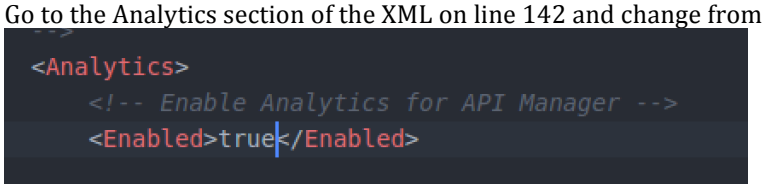
Objectives
Understand API management and key issuing.
Understand API Analytics.
Be able to configure the API Manager and Analytics, and use OAuth2 Bearer Tokens

Software Requirements
OpenJDK 1.8
WSO2 API Manager 2.6.0 (AM)
WSO2 API Manager Analytics 2.6.0 (AMA)
Node.js and npm (and other existing APIs)

1) Firstly we are going to install the WSO2 API Manager and Analytics servers:
```
cd ~/servers
unzip ~/Downloads/wso2am-2.6.0.zip
unzip ~/Downloads/wso2am-analytics-2.6.0.zip
```

2) Now enable analytics:
```
code wso2am-2.6.0/repository/conf/api-manager.xml
```

Go to the Analytics section of the XML on line 142 and change from **false** to **true**, then save.

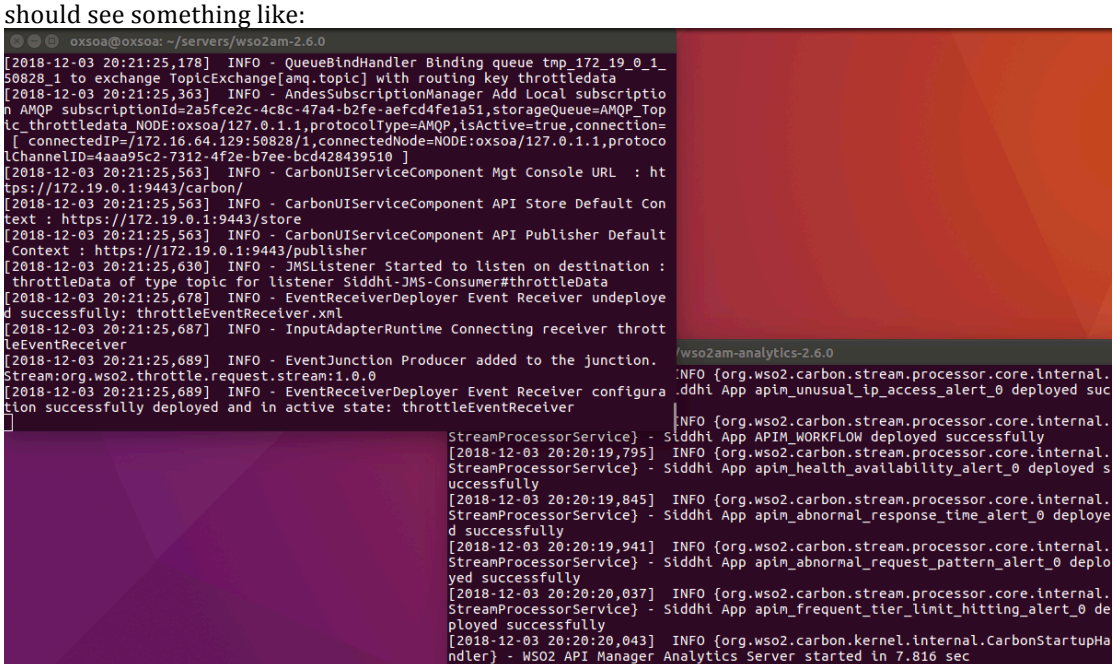


3) From a fresh terminal window or tab start the WSO2 API Manager Analytics:
   a. `cd ~/servers/wso2-am-analytics-2.6.0`
   b. `bin/worker.sh --run`

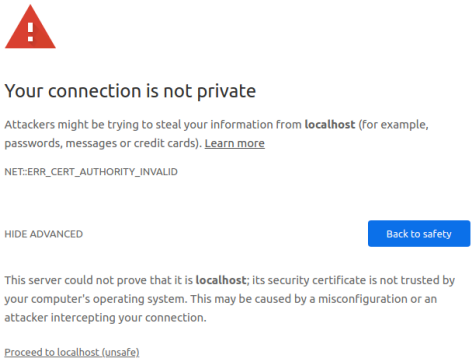
4) The API Manager uses its own internal AMQP server, also on port 5762, so first stop RabbitMQ:
```
sudo service rabbitmq-server stop
```

5) Then in another terminal window start the WSO2 API Manager:
   a. `cd ~/servers/wso2am-2.6.0`
   b. `bin/wso2server.sh`

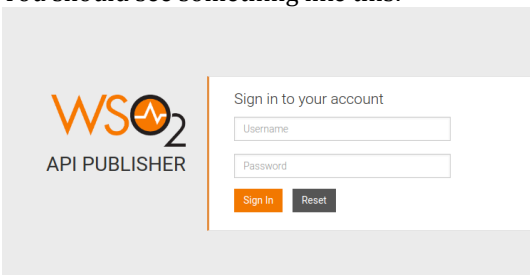
6) Wait until it has started. The first time run is a bit slower as it is performing setup. You should see something like:



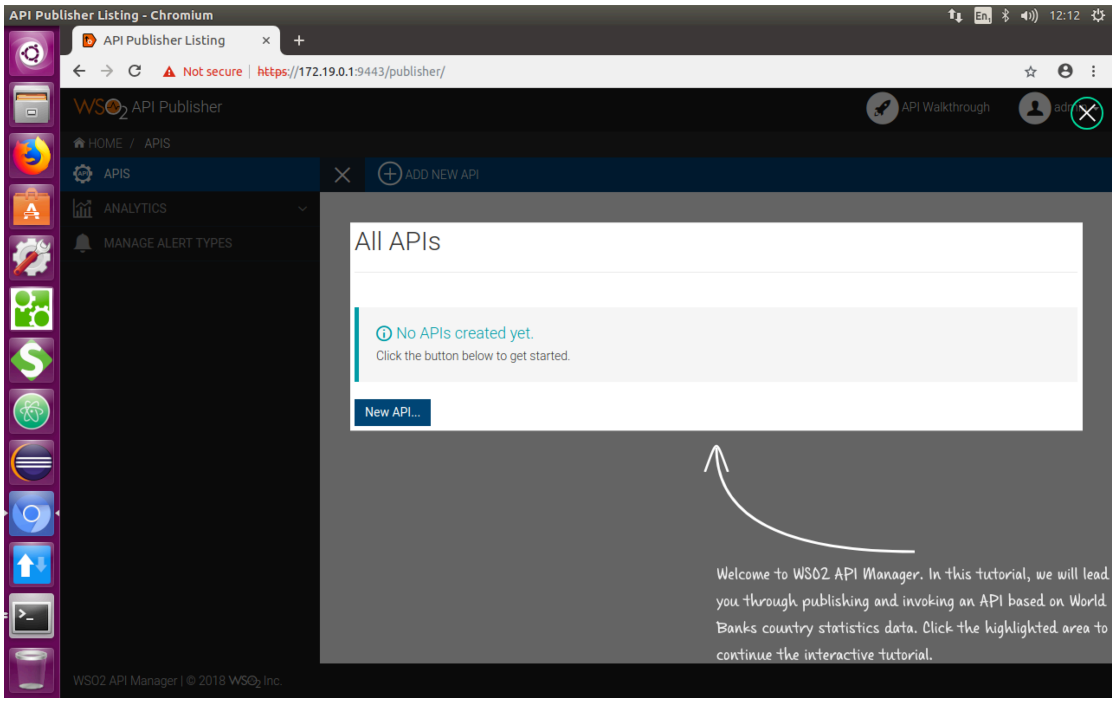
7) Once both servers are started, check that you can access the web interface of the API Manager
   a. https://localhost:9443/ (AM console)
      If this is the first time you try this server you may need to allow the self-signed certificate. **Show Advanced,** then **Proceed to localhost.** You might need to do this twice!



8) You should see something like this:

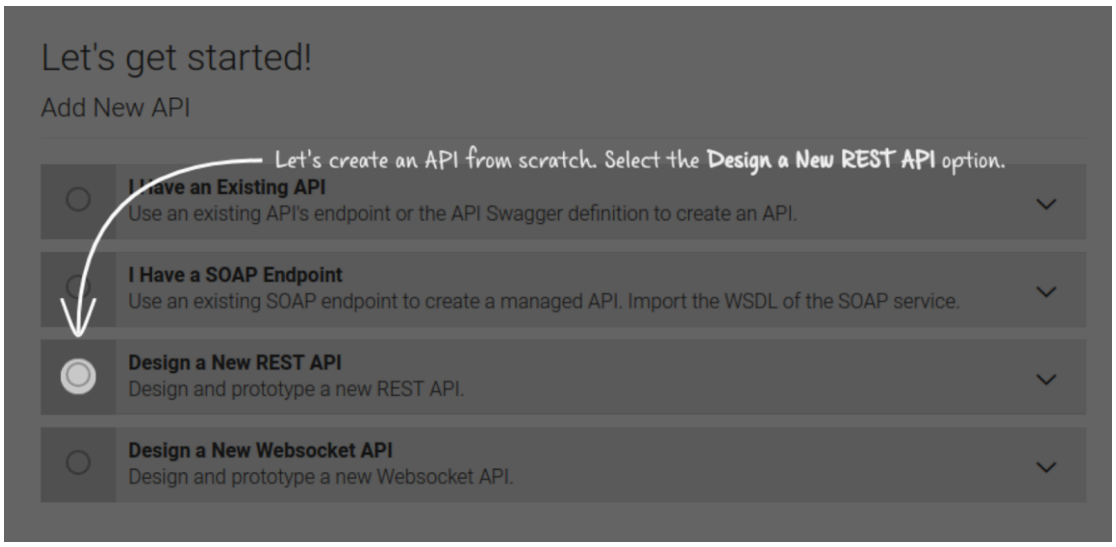


9) Log in as **admin/admin**
   You should see:



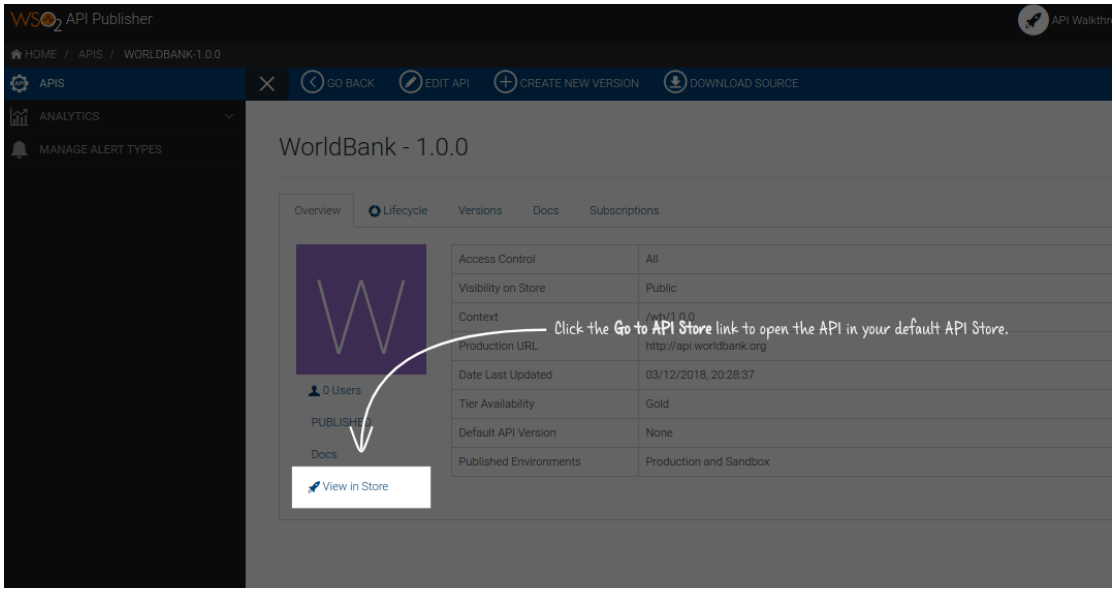
10) An API creator uses the **API Publisher** system to create and publish APIs into the **API Store**. Firstly we will follow the tutorial to create a managed API. Then we will try it out using the API Store.

Click the **New API** button. You will see:

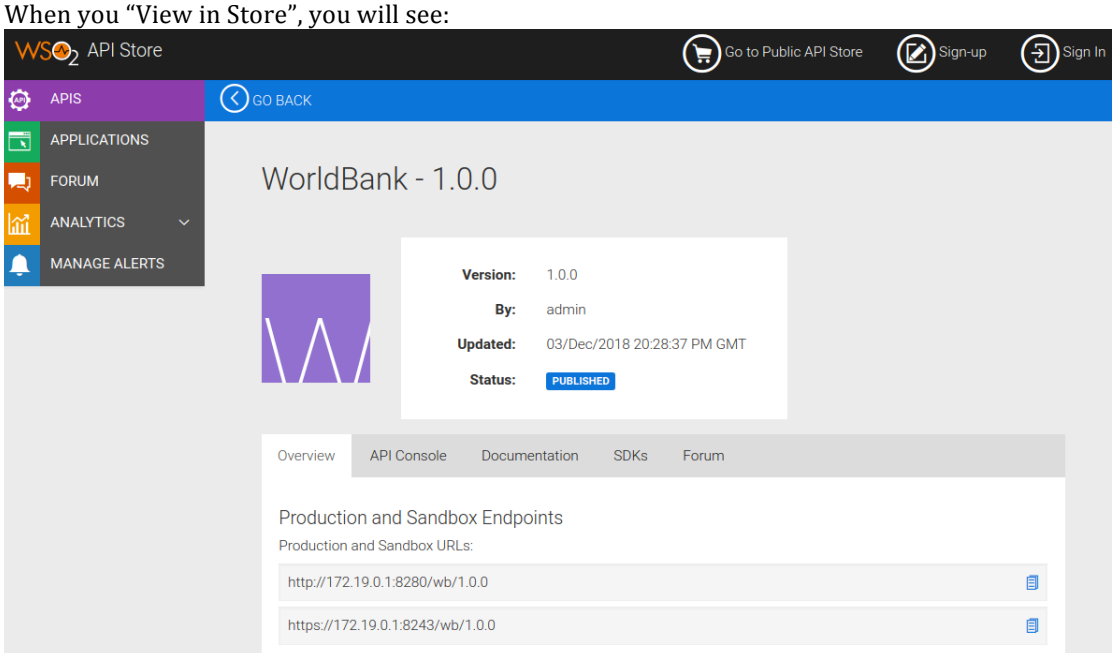


11) Follow the instructions on screen, through a number of steps.

12) You should get to the point where you have created an API, and it is published. The screen should look like:

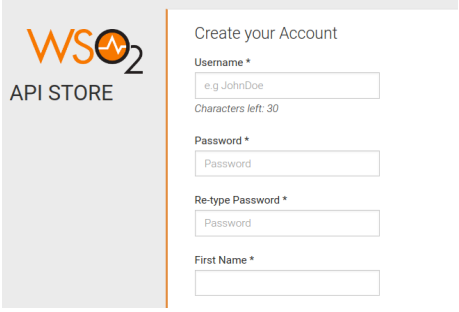


13) When you "View in Store", you will see:



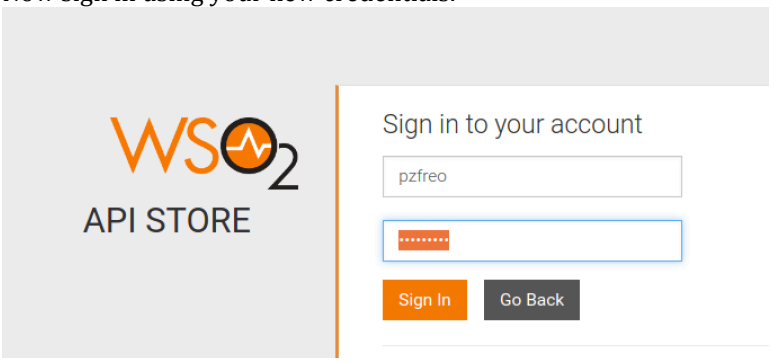
This is the view that an external/third-party developer will see. It can of course be customized (e.g. https://developer.stubhub.com/store/).

14) We now need to sign up as an external developer. Click **Sign Up**

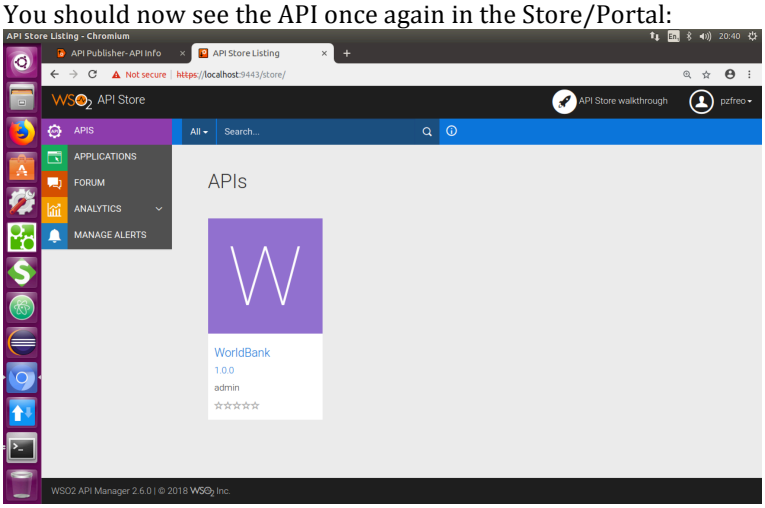


15) Do the usual sort of thing.
(PS the system can also be configured to use Github/Twitter/Google credentials using an OAuth2 flow).

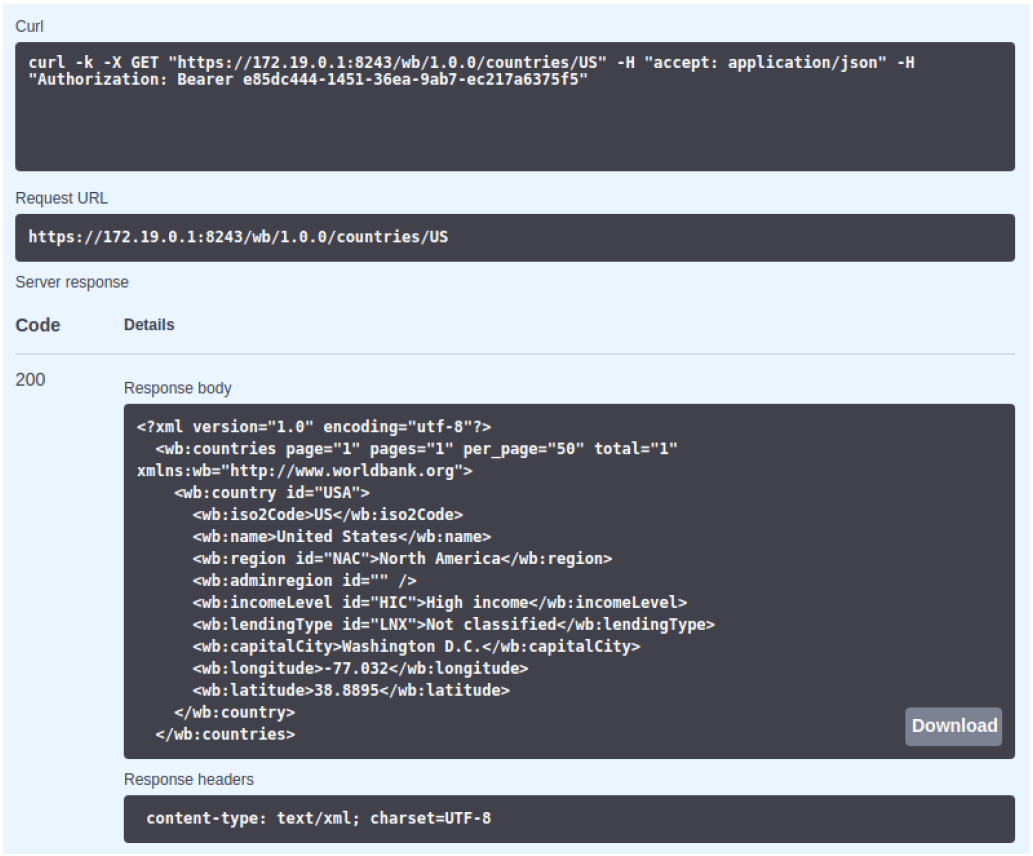
16) Now sign in using your new credentials.



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



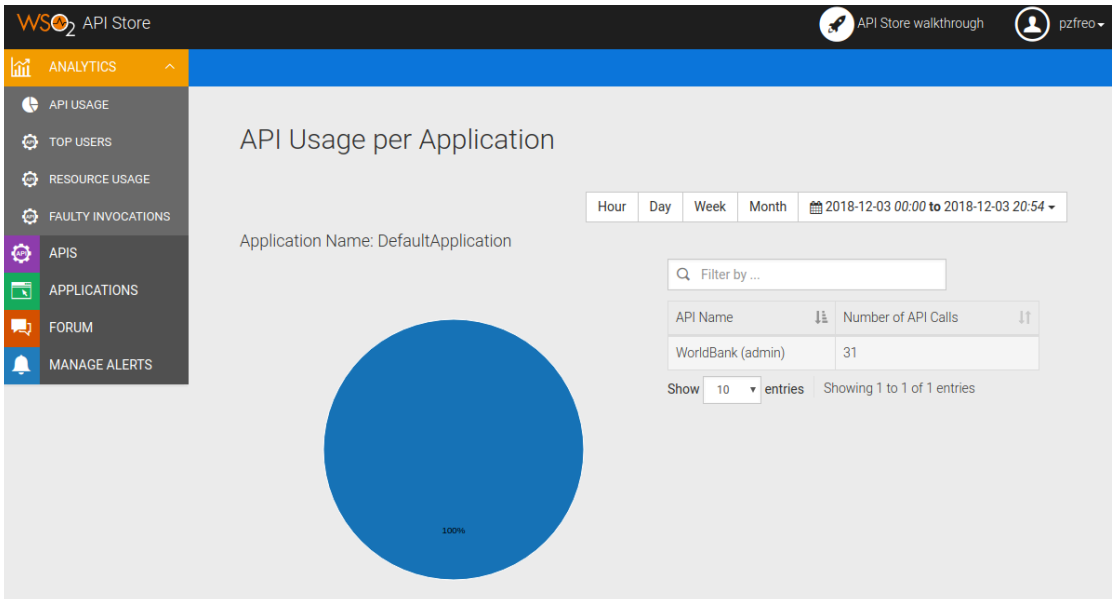
18) Click on **API Store Walkthrough** and then follow the instructions. You should end up with having called the WorldBank API via the API Manager:



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

20) 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, lets explore this a bit more.

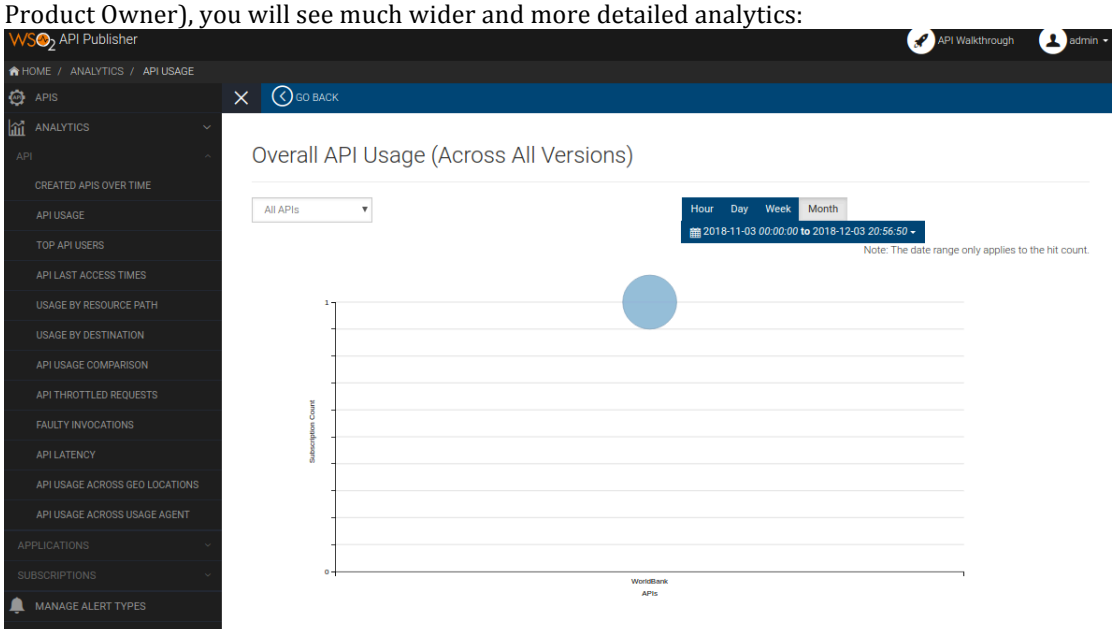
21) As the user of the API you can see some analytics about your own usage. This is particularly useful if the API is charged by usage. In the top left corner of the page, click on **Analytics.** You should see something like:



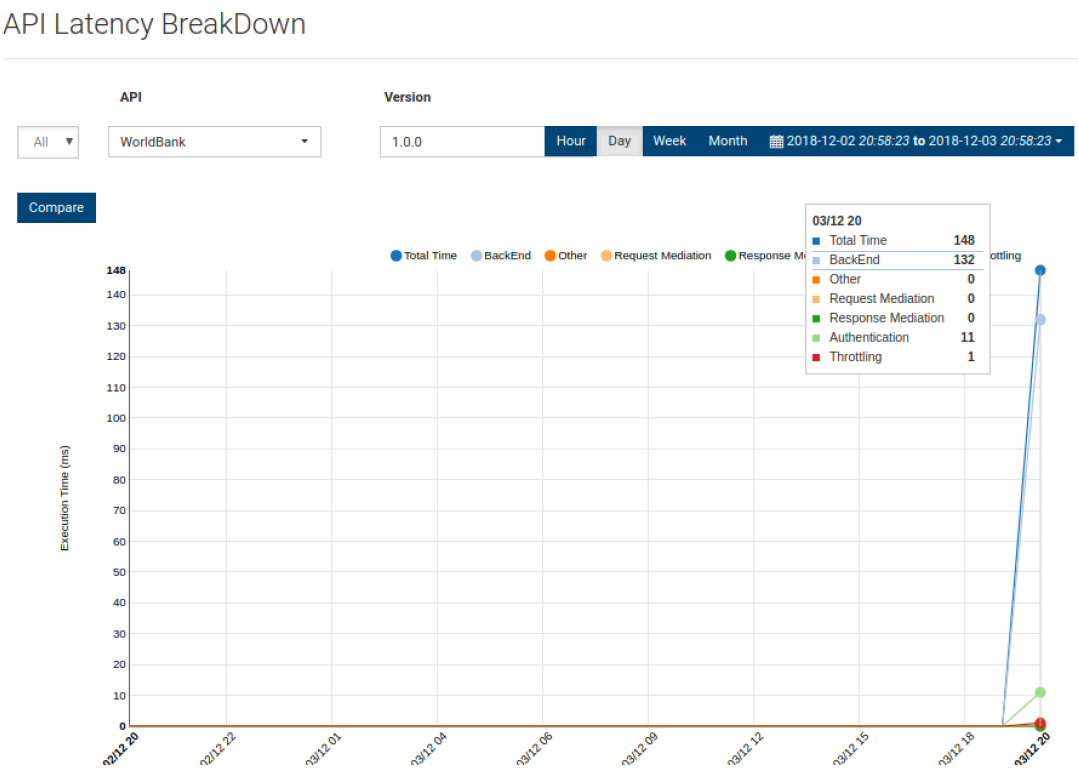
22) Go back to the **API Publisher** (its probably still in the previous tab).

23) Close down the guide (the little green and white cross in the top right corner).

24) Now look at **Analytics** in this system. As the publisher of APIs (sometimes known as a Product Owner), you will see much wider and more detailed analytics:

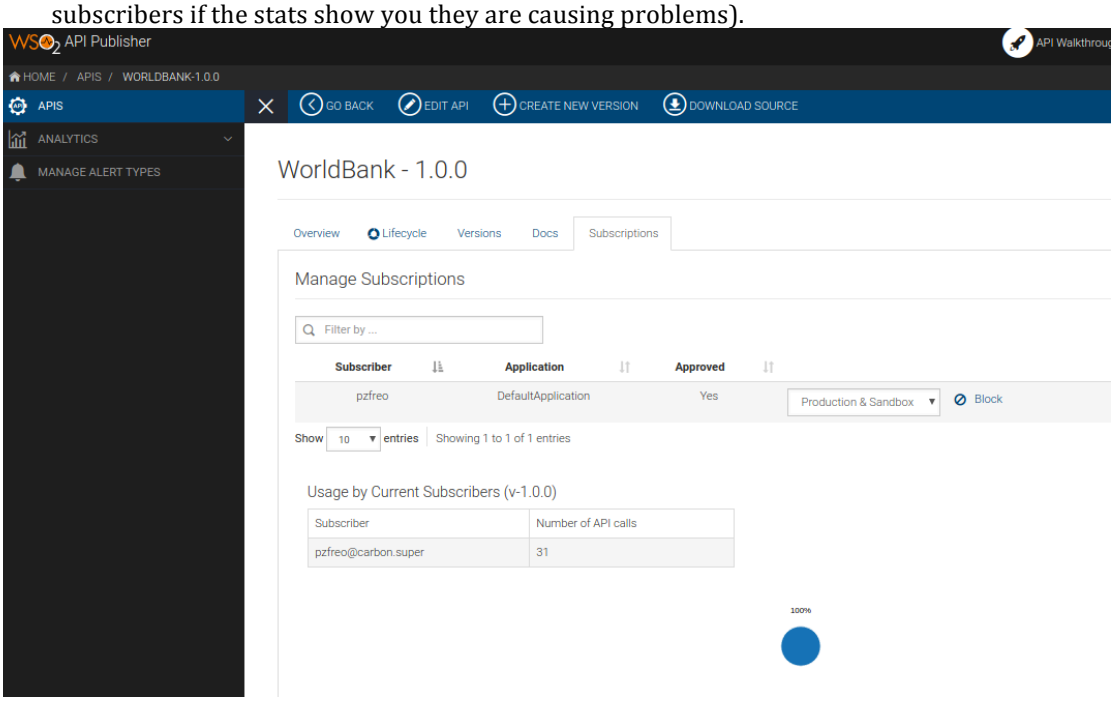


25) For example, take a look at the latency data:



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

27) You can also browse back to the API itself and now look at the subscriptions (and even block subscribers if the stats show you they are causing problems).



Part B - Managing our Purchase API

28) 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
```

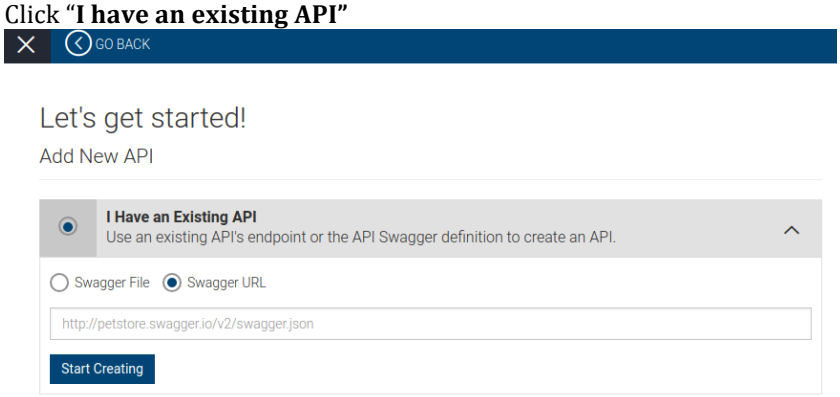
Check it is running:

```
curl http://localhost/purchase
```

29) Go back to the API Publisher tab.

30) Click on **APIs**, then **Add new API**

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



32) Use **Swagger URL**
http://localhost/openapi.yaml
then click **Start Creating**

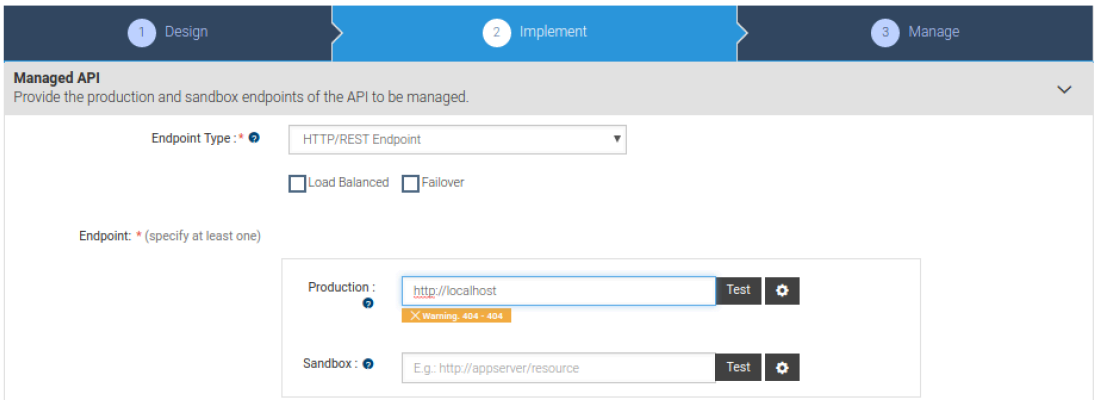
33) You will see that much of this is already filled in (especially the API definition down below). We could have added more data in the annotations in the previous exercise and it would be used here as well.

34) Set the context to be **purchase**

35) Click **Next - Implement**

36) Click **Managed API**

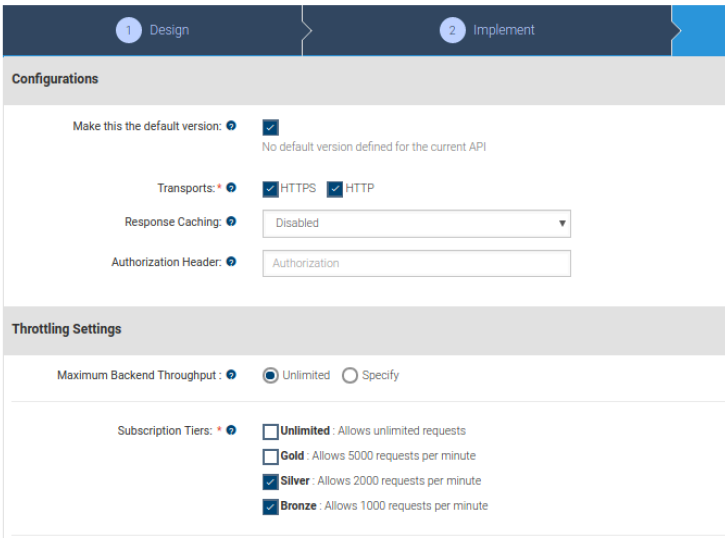
37) Use the purchase server's URL for the Production endpoint. Use:
http://localhost/



Ignore the 404 warning if you test it.

38) Click **Next Manage**

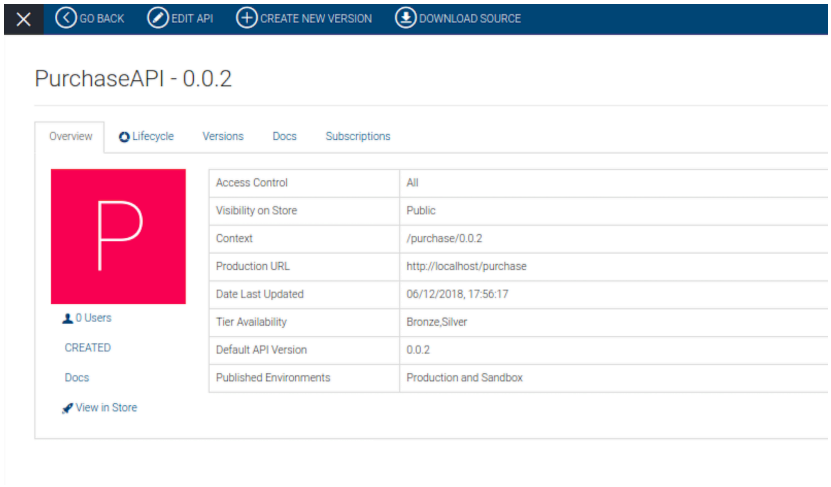
39) Click **Make this the default version.**
Set the throttling to enable **bronze, silver** and **gold**



40) Click Save

41) On the left hand menu click **APIs**, then click on the **PurchaseAPI**

42) Have a look at the API



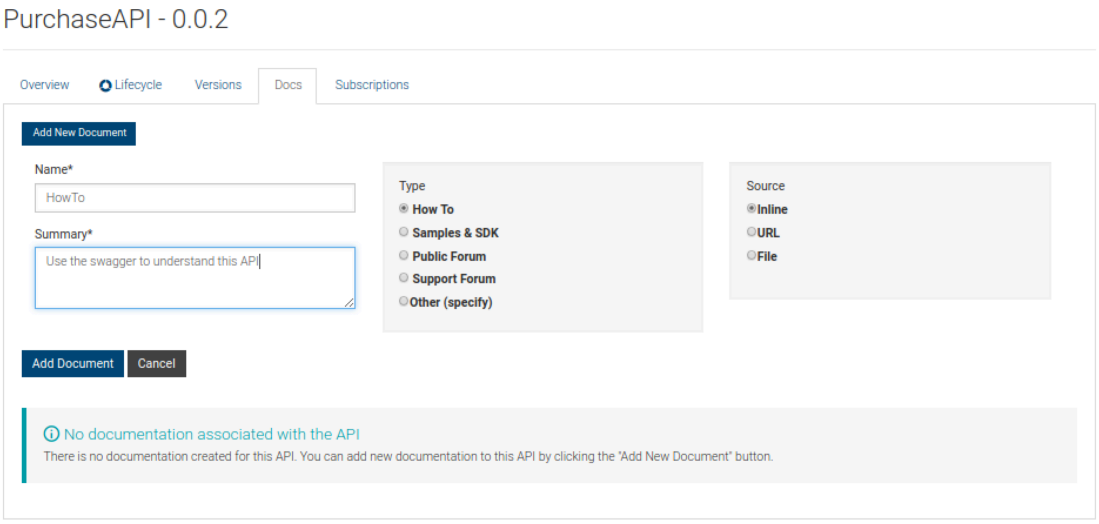
43) **Adding Documentation:** Select the Docs tab.

44) Click **Add New Document** link.

Documentation can be provided inline, via a URL or as a file. For inline documentation, you can edit the content directly from the API publisher interface. You get several documents types:

1. Swagger documents
2. How To
3. Samples and SDK
4. Public forum / Support forum (external link only)
5. API message formats
6. Other

45) Select the **How To** type, a name for the document and a short description, which will appear in the API Store. Select inline or provide a URL.

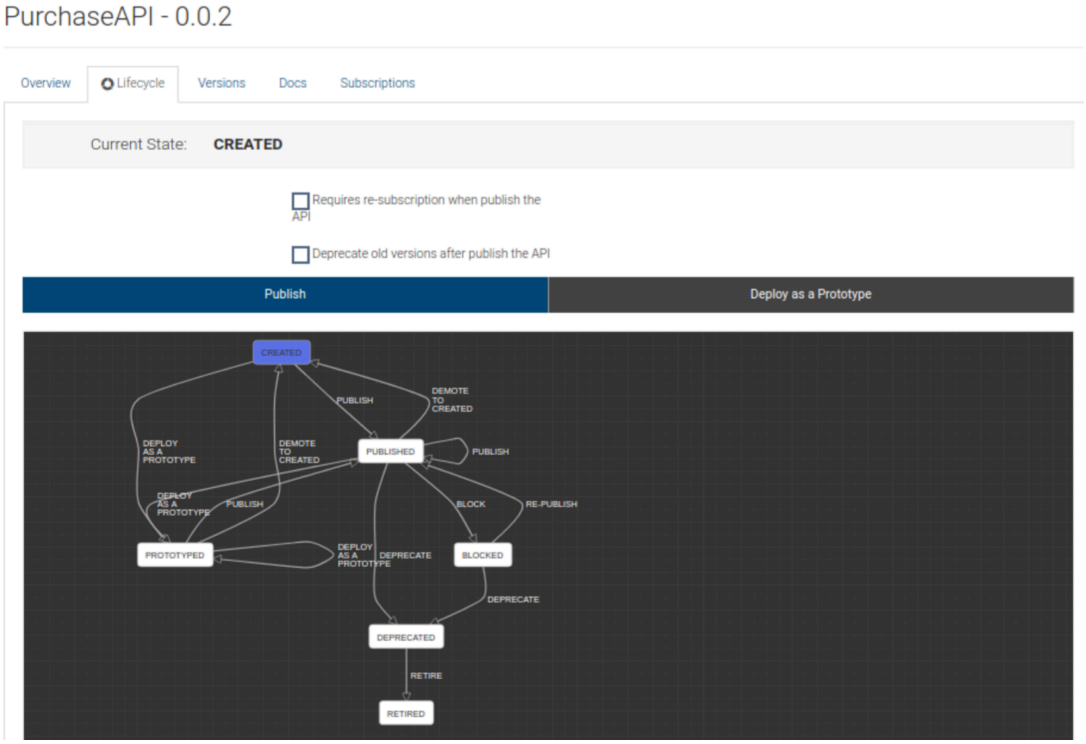


46) Click **Add Document**.

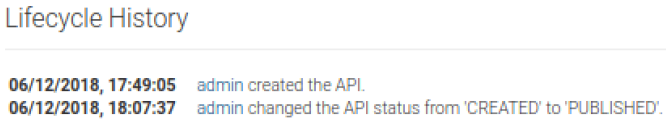
47) There is also an inline rich text editor you can use instead.

Publishing!

48) Choose the **Lifecycle** tab. Look at the lifecycle diagram. Now click **Publish**



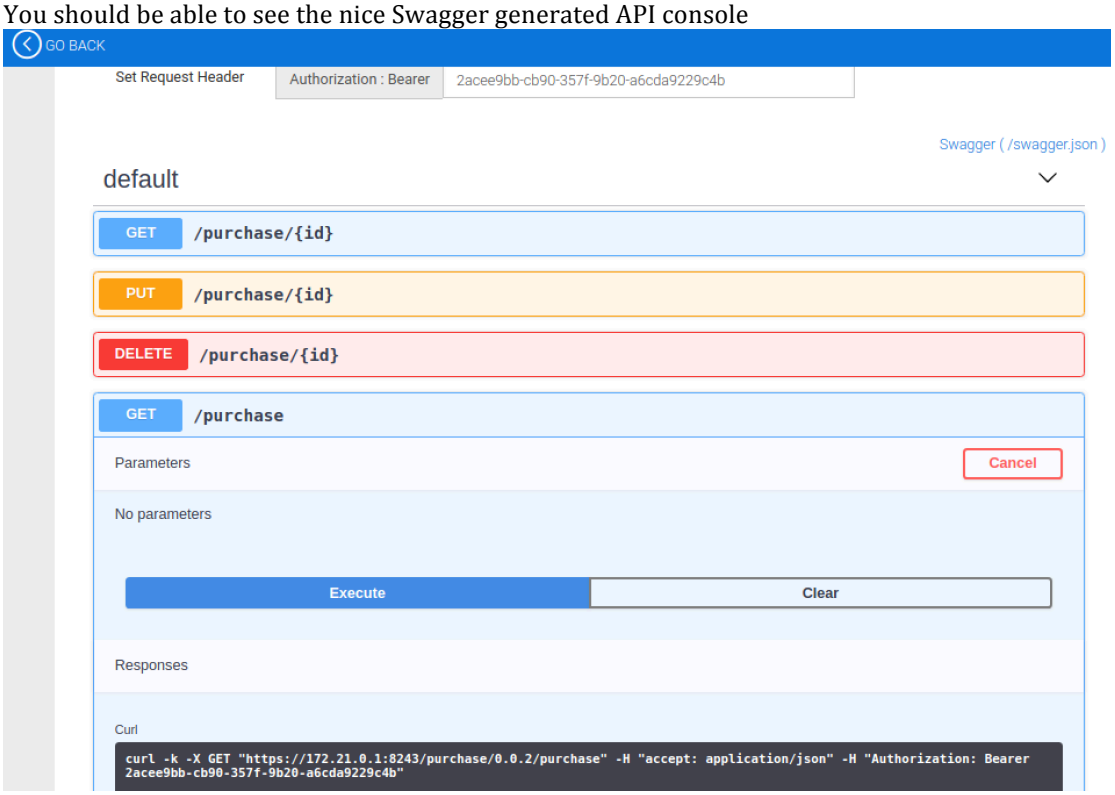
Notice that this page also captures the audit log:



Subscription

49) Once again subscribe to this API as before:
    a. Go back to the main tab for the API
    b. Click "**View in Store**"
    c. You may need to sign in again
    d. Subscribe this API as before.

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



51) 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.