

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.

```
<Analytics>  
    <!-- Enable Analytics for API Manager -->  
    <Enabled>true</Enabled>
```

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

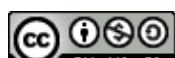
- a. cd ~/servers/wso2-am-analytics-2.6.0
- b. bin/worker.sh --run

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

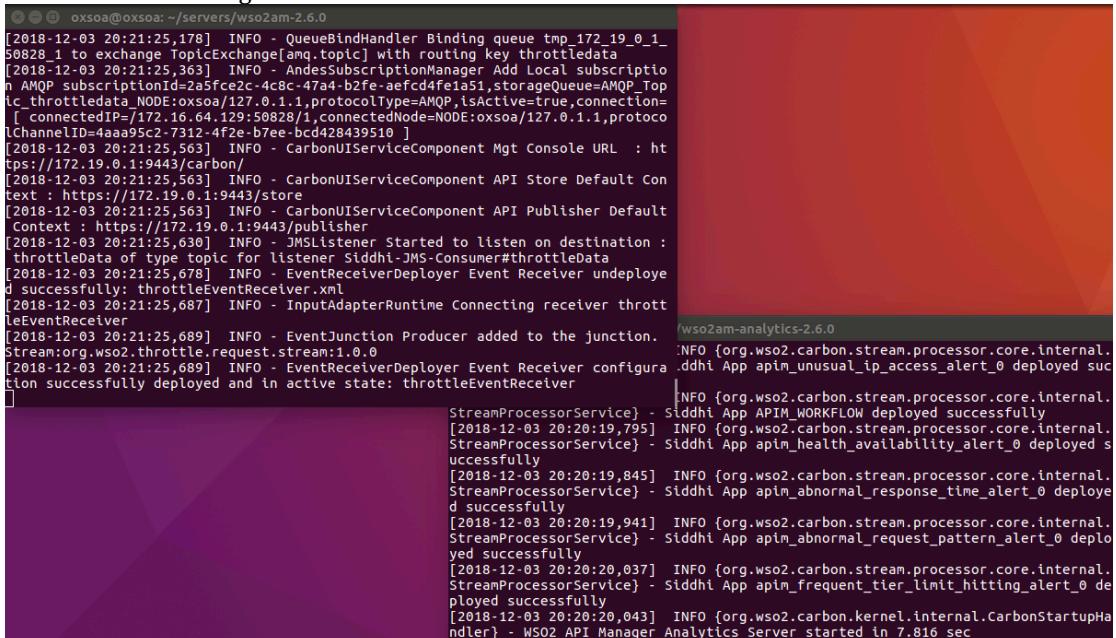
```
sudo service rabbitmq-server stop
```

- 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:



```
[2018-12-03 20:21:25,178] INFO - QueueBindHandler Binding queue tmp_172_19_0_1_50828_1 to exchange TopicExchange[amq.topic] with routing key throttledata
[2018-12-03 20:21:25,363] INFO - AddressSubscriptionManager Add Local subscriptionId=2a5fce2c-4c8c-47a4-b2fe-aefcd4fe1a51,storageQueue=AMQP_Topic,throttledData_NODE=oxtsoa/127.0.1.1,protocolType=AMQP,isActive=true,connection=[connectedIP=/172.16.129.50828/1,connectedPort=129.50828,connectedNode=NODE:oxtsoa/127.0.1.1,protocolID=4aaa95c2-7312-4f2e-b7ee-bcd428439510]
[2018-12-03 20:21:25,563] INFO - CarbonUIServiceComponent Mgt Console URL : https://172.19.0.1:9443/carbon/
[2018-12-03 20:21:25,563] INFO - CarbonUIServiceComponent API Store Default Context : https://172.19.0.1:9443/store
[2018-12-03 20:21:25,563] INFO - CarbonUIServiceComponent API Publisher Default Context : https://172.19.0.1:9443/publisher
[2018-12-03 20:21:25,630] INFO - JMSListener Started to listen on destination : throttleData of type topic for listener Siddhi-JMS-Consumer#throttledData
[2018-12-03 20:21:25,678] INFO - EventReceiverDeployer Event Receiver undeployed successfully: throttleEventReceiver.xml
[2018-12-03 20:21:25,687] INFO - InputAdapterRuntime Connecting receiver throttleEventReceiver
[2018-12-03 20:21:25,689] INFO - EventJunction Producer added to the junction. Stream:org.wso2.throttle.request.stream:1.0.0
[2018-12-03 20:21:25,689] INFO - EventReceiverDeployer Event Receiver configuration successfully deployed and in active state: throttleEventReceiver
]
[wso2am-analytics-2.6.0
INFO {org.wso2.carbon.stream.processor.core.internal.StreamProcessorService} - Siddhi App APIM_WORKFLOW deployed successfully
[2018-12-03 20:20:19,795] INFO {org.wso2.carbon.stream.processor.core.internal.StreamProcessorService} - Siddhi App apim_health_availability_alert_0 deployed successfully
[2018-12-03 20:20:19,845] INFO {org.wso2.carbon.stream.processor.core.internal.StreamProcessorService} - Siddhi App apim_abnormal_response_time_alert_0 deployed successfully
[2018-12-03 20:20:19,941] INFO {org.wso2.carbon.stream.processor.core.internal.StreamProcessorService} - Siddhi App apim_abnormal_request_pattern_alert_0 deployed successfully
[2018-12-03 20:20:20,037] INFO {org.wso2.carbon.stream.processor.core.internal.StreamProcessorService} - Siddhi App apim_frequent_tier_limit_hitting_alert_0 deployed successfully
[2018-12-03 20:20:20,043] INFO {org.wso2.carbon.kernel.internal.CarbonStartupHandler} - WSO2 API Manager Analytics Server started in 7.816 sec
```

- 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!



Your connection is not private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

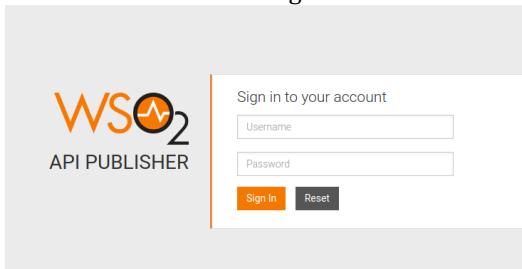
[HIDE ADVANCED](#)

[Back to safety](#)

This server could not prove that it is **localhost**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

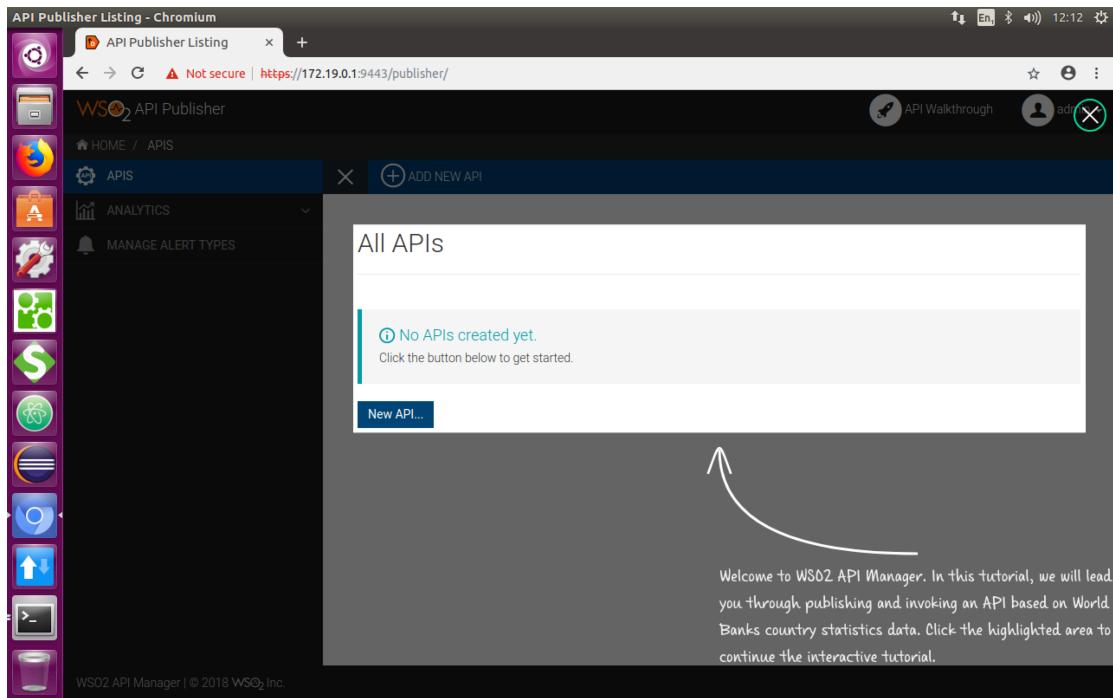
[Proceed to localhost \(unsafe\)](#)

- 8) You should see something like this:



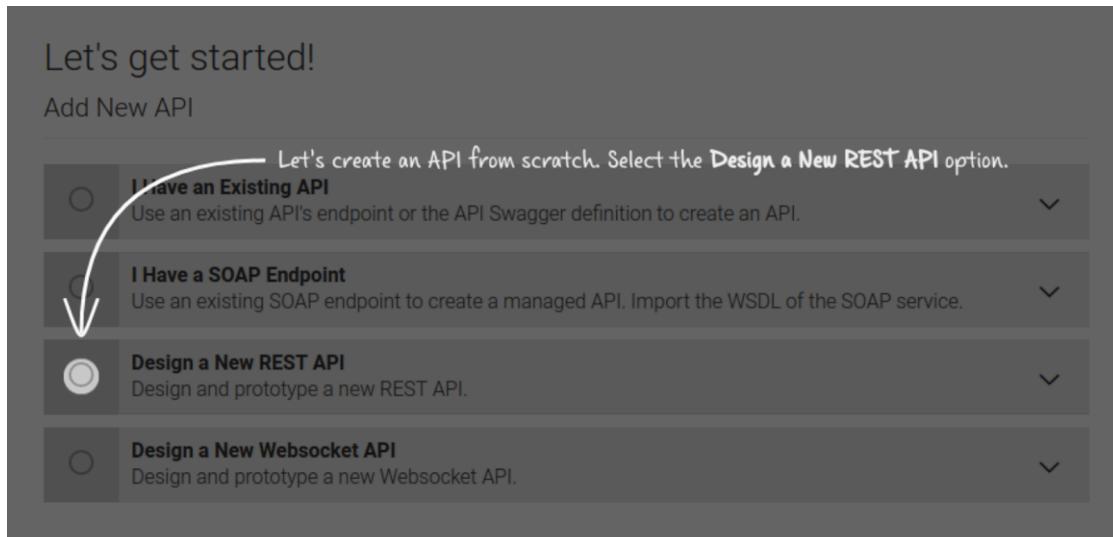
- 9) Log in as **admin/admin**

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:



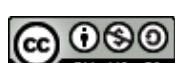
- 11) Follow the instructions on screen, through a number of steps.
- 12) You should get to the point where you have created an API, and it is published. The screen should look like:

The screenshot shows the WSO2 API Publisher interface. On the left, there's a sidebar with 'APIS' selected. The main area displays the 'WorldBank - 1.0.0' API. The 'Overview' tab is active. Below the title, there's a purple square icon with a white 'W'. To its right, it says '0 Users' and 'PUBLISHED'. Below that is a 'Docs' section with a 'View in Store' button. To the right of the button is a tooltip: 'Click the Go to API Store link to open the API in your default API Store.' Further down, there's a table with API details like Access Control (All), Visibility on Store (Public), and Production URL (<http://api.worldbank.org>). The 'Lifecycle' tab is also visible at the top.

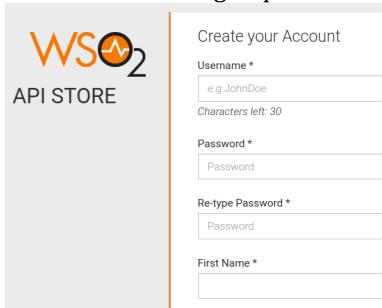
- 13) When you "View in Store", you will see:

The screenshot shows the WSO2 API Store interface. On the left, there's a sidebar with 'APPS' selected. The main area displays the 'WorldBank - 1.0.0' API. The 'API Console' tab is active. It shows basic information: Version 1.0.0, By admin, Updated 03/Dec/2018 20:28:37 PM GMT, and Status PUBLISHED. Below this, it lists 'Production and Sandbox Endpoints' and 'Production and Sandbox URLs' with links to <http://172.19.0.1:8280/wb/1.0.0> and <https://172.19.0.1:8243/wb/1.0.0>. The 'Overview' tab is also visible at the top.

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**

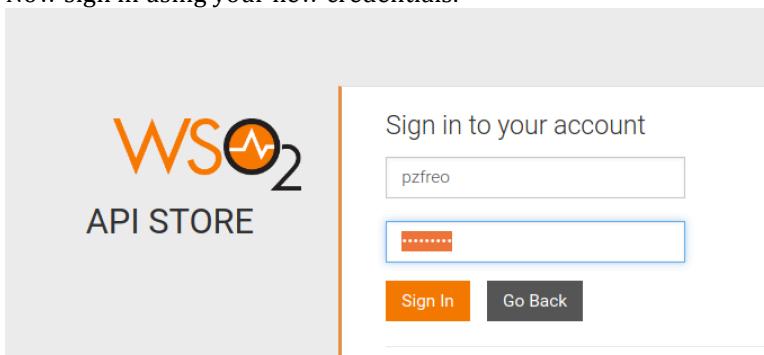


The screenshot shows the 'Create your Account' form on the WSO2 API Store. It includes fields for Username, Password, Re-type Password, and First Name, each with validation messages below them.

- 15) Do the usual sort of thing.

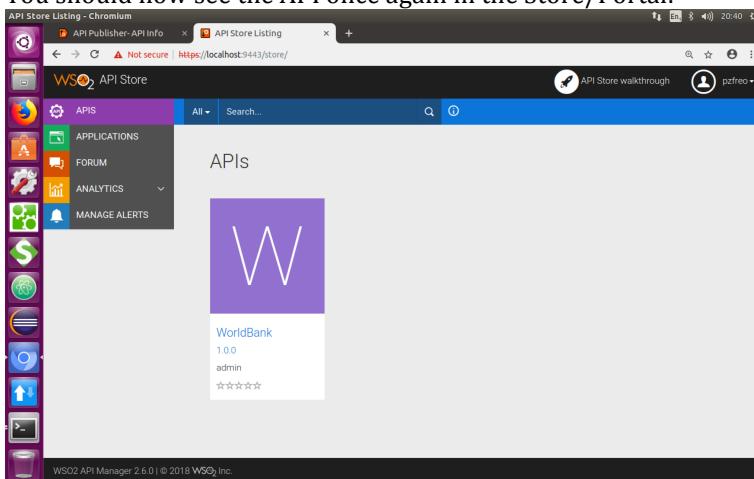
(PS the system can also be configured to use Github/Twitter/Google credentials using an OAuth2 flow).

- 16) Now sign in using your new credentials.



The screenshot shows the 'Sign in to your account' form on the WSO2 API Store. It has fields for Username and Password, and buttons for 'Sign In' and 'Go Back'.

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



The screenshot shows the WSO2 API Manager portal. The left sidebar has icons for APIs, Applications, Forum, Analytics, and Manage Alerts. The main area displays a card for the 'WorldBank' API, version 1.0.0, created by 'admin'. The card shows a rating of 5 stars.

- 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:

Curl

```
curl -k -X GET "https://172.19.0.1:8243/wb/1.0.0/countries/US" -H "accept: application/json" -H "Authorization: Bearer e85dc444-1451-36ea-9ab7-ec217a6375f5"
```

Request URL

```
https://172.19.0.1:8243/wb/1.0.0/countries/US
```

Server response

Code	Details
200	Response body

```
<?xml version="1.0" encoding="utf-8"?>
<wb:countries page="1" pages="1" per_page="50" total="1"
xmlns:wb="http://www.worldbank.org">
<wb:country id="USA">
<wb:iso2Code>US</wb:iso2Code>
<wb:name>United States</wb:name>
<wb:region id="NAC">North America</wb:region>
<wb:adminregion id="" />
<wb:incomeLevel id="HIC">High income</wb:incomeLevel>
<wb:lendingType id="LNX">Not classified</wb:lendingType>
<wb:capitalCity>Washington D.C.</wb:capitalCity>
<wb:longitude>-77.032</wb:longitude>
<wb:latitude>38.8895</wb:latitude>
</wb:country>
</wb:countries>
```

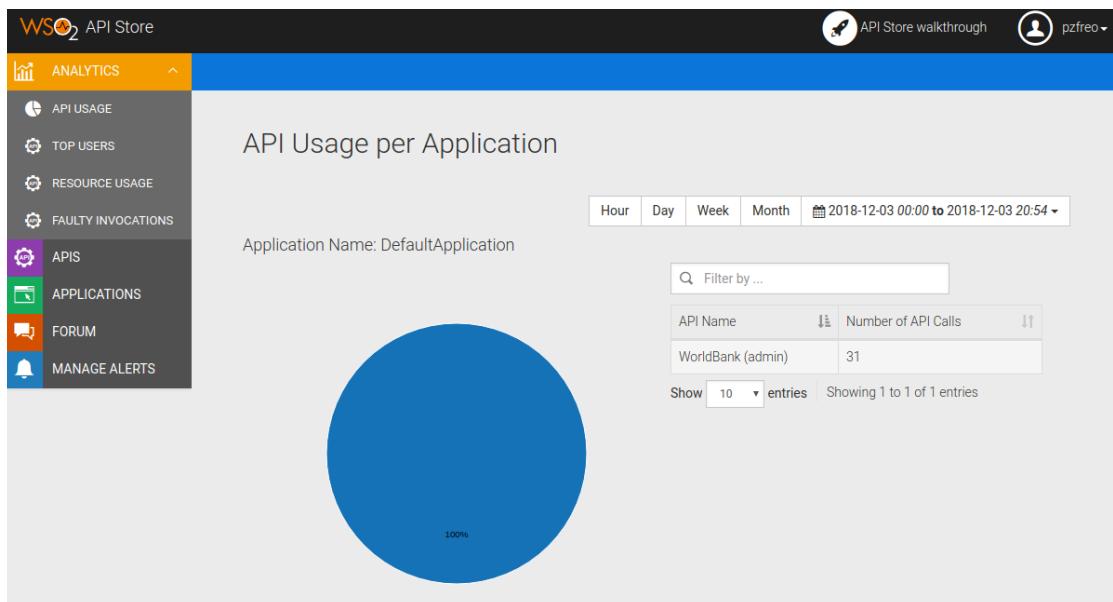
[Download](#)

Response headers

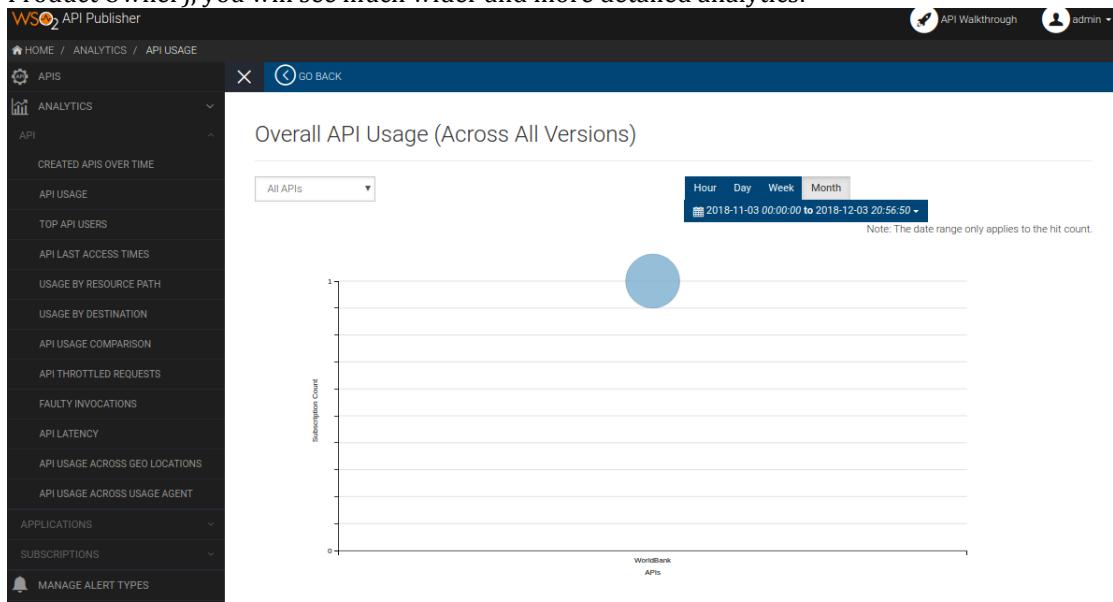
```
content-type: text/xml; charset=UTF-8
```

- 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:



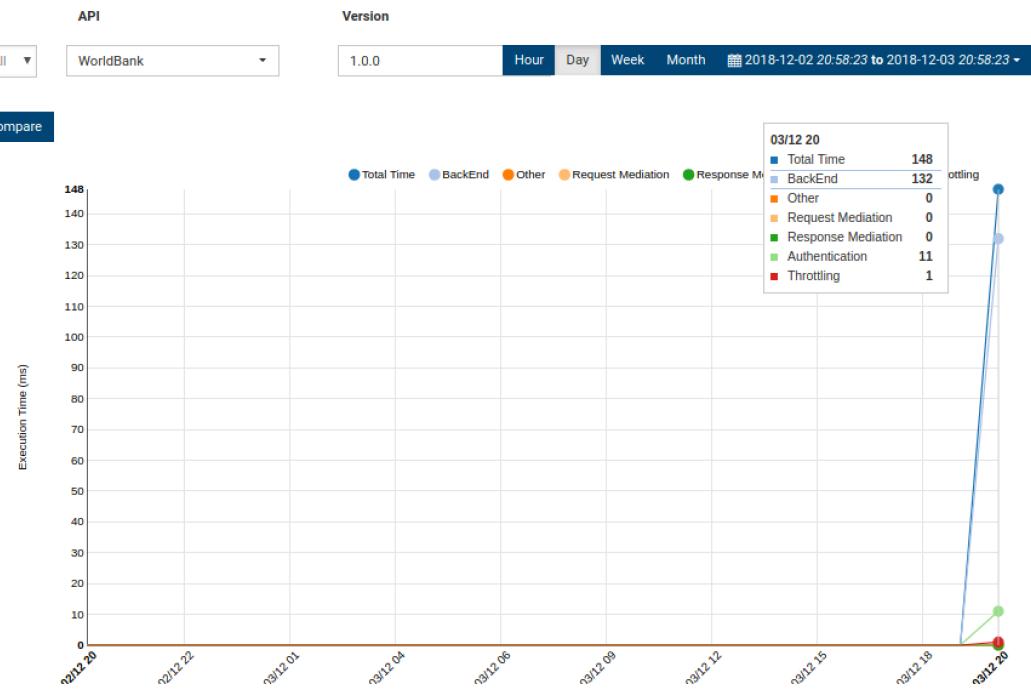


- 22) Go back to the **API Publisher** (its probably still in the previous tab).
- 23) Close down the guide (the little green and white cross in the top right corner).
- 24) Now look at **Analytics** in this system. As the publisher of APIs (sometimes known as a Product Owner), you will see much wider and more detailed analytics:



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

API Latency BreakDown



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).

Subscriber	Application	Approved
pzfro	DefaultApplication	Yes

Show 10 entries | Showing 1 to 1 of 1 entries

Usage by Current Subscribers (v-1.0.0)

Subscriber	Number of API calls
pzfro@carbon.super	31

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**”

GO BACK

Let's get started!

Add New API

I Have an Existing API
Use an existing API's endpoint or the API Swagger definition to create an API.

Swagger File Swagger URL

Start Creating

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.

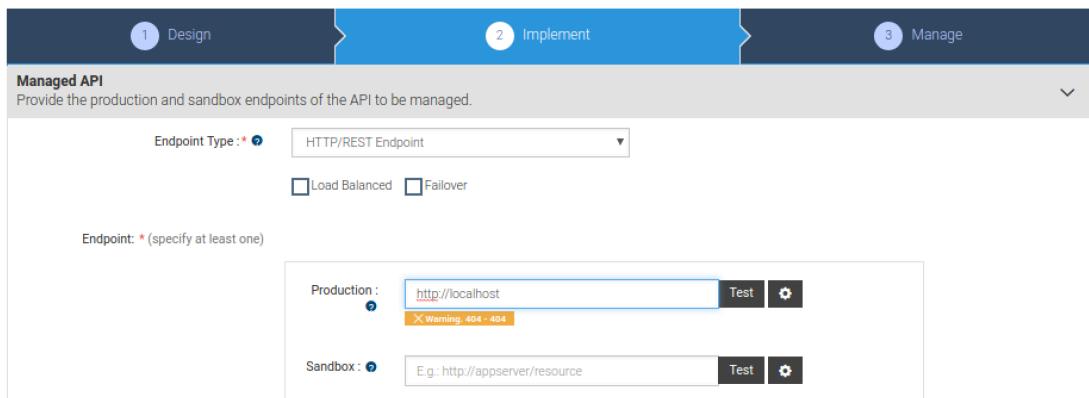
34) Set the context to be **purchase**

35) Click **Next - Implement**

36) Click **Managed API**



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



Managed API
Provide the production and sandbox endpoints of the API to be managed.

Endpoint Type: * **HTTP/REST Endpoint**

Load Balanced Failover

Endpoint: * (specify at least one)

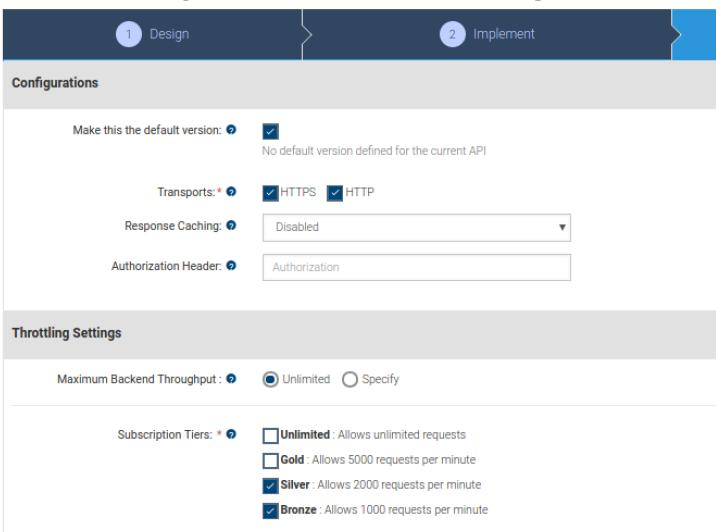
Production: **http://localhost** **Test** **Settings**
Warning: 404 - 404.

Sandbox: **E.g.: http://appserver/resource** **Test** **Settings**

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**



Configurations

Make this the default version: No default version defined for the current API

Transports: * HTTPS HTTP

Response Caching: Disabled

Authorization Header: Authorization

Throttling Settings

Maximum Backend Throughput: Unlimited Specify

Subscription Tiers: * **Unlimited**: Allows unlimited requests
 Gold: Allows 5000 requests per minute
 Silver: Allows 2000 requests per minute
 Bronze: Allows 1000 requests per minute

40) Click Save

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



42) Have a look at the API

The screenshot shows the API Publisher interface with the 'PurchaseAPI - 0.0.2' page. At the top, there are navigation links: 'GO BACK', 'EDIT API', 'CREATE NEW VERSION', and 'DOWNLOAD SOURCE'. Below this is a header with tabs: 'Overview' (selected), 'Lifecycle', 'Versions', 'Docs', and 'Subscriptions'. The main content area has a large red 'P' icon. To its left, there are sections for 'Users' (0), 'CREATED', 'Docs', and a 'View in Store' link. On the right, there is a table with the following data:

Access Control	All
Visibility on Store	Public
Context	/purchase/0.0.2
Production URL	http://localhost/purchase
Date Last Updated	06/12/2018, 17:56:17
Tier Availability	Bronze,Silver
Default API Version	0.0.2
Published Environments	Production and Sandbox

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.

The screenshot shows the 'Add New Document' dialog for the 'PurchaseAPI - 0.0.2' API. The dialog has fields for 'Name*' (containing 'HowTo'), 'Summary*' (containing 'Use the swagger to understand this API'), 'Type' (radio button selected for 'How To'), and 'Source' (radio button selected for 'Inline'). At the bottom are 'Add Document' and 'Cancel' buttons, and a note: '(i) No documentation associated with the API'.

46) Click Add Document.

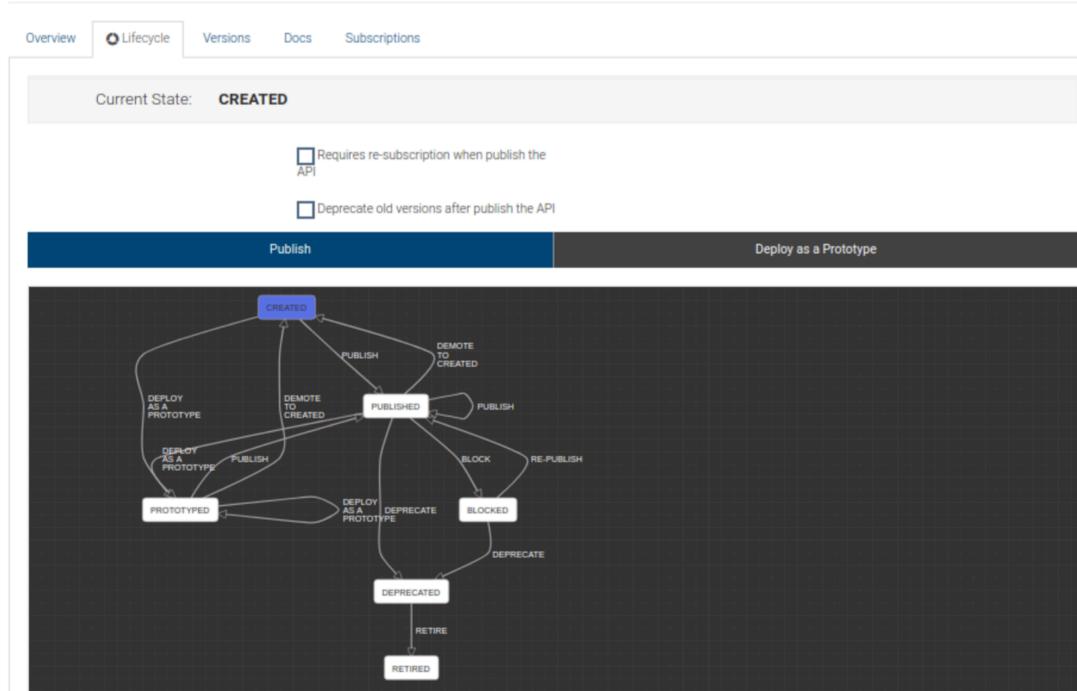
47) There is also an inline rich text editor you can use instead.



Publishing!

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

PurchaseAPI - 0.0.2



Notice that this page also captures the audit log:

[Lifecycle History](#)

06/12/2018, 17:49:05 admin created the API.
06/12/2018, 18:07:37 admin changed the API status from 'CREATED' to 'PUBLISHED'.

Subscription

49) Once again subscribe to this API as before:

- Go back to the main tab for the API
- Click “**View in Store**”
- You may need to sign in again
- Subscribe this API as before.

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

The screenshot shows the Swagger UI interface for a REST API. At the top, there are buttons for 'Set Request Header' and 'Authorization : Bearer' with the value '2acee9bb-cb90-357f-9b20-a6cda9229c4b'. To the right is a link to 'Swagger (/swagger.json)'. Below this, the 'default' API group is selected. The '/purchase' endpoint is shown with three methods: GET, PUT, and DELETE. The GET method is highlighted in blue. The PUT and DELETE methods are shown in orange and red respectively. Below the methods, the '/purchase' endpoint is expanded, showing its details. It has a 'GET /purchase' method with parameters and a 'Cancel' button. The 'Parameters' section indicates 'No parameters'. Below this is a 'Responses' section and a 'Curl' section containing the command: `curl -k -X GET "https://172.21.0.1:8243/purchase/0.0.2/purchase" -H "accept: application/json" -H "Authorization: Bearer 2acee9bb-cb90-357f-9b20-a6cda9229c4b"`.

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.

