

# 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).**

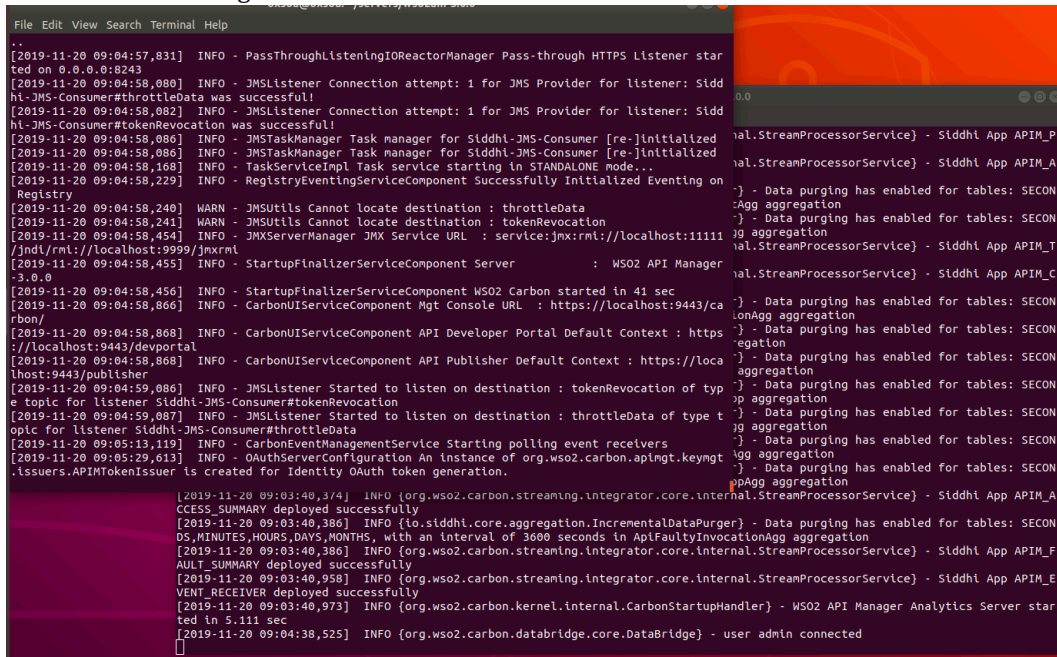
Change `enable` from **false** to **true**, then save.

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]]
```

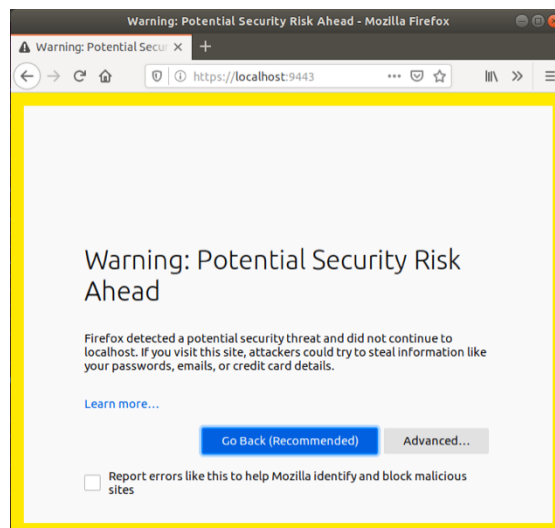
- 3) From a fresh terminal window or tab start the WSO2 API Manager Analytics:
  - a. `cd ~/servers/wso2-am-analytics-3.0.0`
  - b. `bin/worker.sh --run`
- 4) The API Manager uses its own internal AMQP server, also on port 5762, so first stop RabbitMQ:  
`sudo service rabbitmq-server stop`
- 5) Then 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:



```
[2019-11-20 09:04:57,831] INFO - PassThroughListenerIOReactorManager Pass-through HTTPS Listener started on 0.0.0.0:8243
[2019-11-20 09:04:58,000] INFO - JMSListener Connection attempt: 1 for JMS Provider for listener: Siddhi-JMS-Consumer#throttleData was successful!
[2019-11-20 09:04:58,082] INFO - JMSListener Connection attempt: 1 for JMS Provider for listener: Siddhi-JMS-Consumer#tokenRevocation was successful!
[2019-11-20 09:04:58,086] INFO - JMSTaskManager Task manager for Siddhi-JMS-Consumer [re-]initialized
[2019-11-20 09:04:58,086] INFO - JMSTaskManager Task manager for Siddhi-JMS-Consumer [re-]initialized
[2019-11-20 09:04:58,108] INFO - TaskServiceImpl Task service starting in STANDBY mode...
[2019-11-20 09:04:58,229] INFO - RegistryEventingServiceComponent Successfully Initialized Eventing on Registry
[2019-11-20 09:04:58,240] WARN - JMSUtils Cannot locate destination : throttleData
[2019-11-20 09:04:58,241] WARN - JMSUtils Cannot locate destination : tokenRevocation
[2019-11-20 09:04:58,454] INFO - JMSXServiceManager JMX Service URL : service:jmx:rmi://localhost:11111/jndi/rmi://localhost:9999/jmxrmi
[2019-11-20 09:04:58,455] INFO - StartupFinalizerServiceComponent Server : WS02 API Manager
[2019-11-20 09:04:58,456] INFO - StartupFinalizerServiceComponent WSO2 Carbon started in 41 sec
[2019-11-20 09:04:58,866] INFO - CarbonUIServiceComponent Mgt Console URL : https://localhost:9443/carbon/
[2019-11-20 09:04:58,868] INFO - CarbonUIServiceComponent API Developer Portal Default Context : https://localhost:9443/devportal
[2019-11-20 09:04:58,868] INFO - CarbonUIServiceComponent API Publisher Default Context : https://localhost:9443/publisher
[2019-11-20 09:04:59,086] INFO - JMSListener Started to listen on destination : tokenRevocation of type tokenRevocation for listener Siddhi-JMS-Consumer#tokenRevocation
[2019-11-20 09:04:59,087] INFO - JMSListener Started to listen on destination : throttleData of type throttleData for listener Siddhi-JMS-Consumer#throttleData
[2019-11-20 09:05:13,119] INFO - CarbonEventManagerService Starting polling event receivers
[2019-11-20 09:05:29,613] INFO - OAuthServerConfiguration An instance of org.wso2.carbon.apimgt.keymgmt.issuers.APIMTokenIssuer is created for Identity OAuth token generation.
[2019-11-20 09:03:40,374] INFO {org.wso2.carbon.streaming.integrator.core.internal.StreamProcessorService} - Siddhi App APIM_A SUCCESS_SUMMARY deployed successfully
[2019-11-20 09:03:40,386] INFO {io.siddhi.core.aggregation.IncrementalDataPurger} - Data purging has enabled for tables: SECONDS, MINUTES, HOURS, DAYS, MONTHS, with an interval of 3600 seconds in ApiFaultyInvocationAgg aggregation
[2019-11-20 09:03:40,386] INFO {org.wso2.carbon.streaming.integrator.core.internal.StreamProcessorService} - Siddhi App APIM_F SUCCESS_SUMMARY deployed successfully
[2019-11-20 09:03:40,958] INFO {org.wso2.carbon.streaming.integrator.core.internal.StreamProcessorService} - Siddhi App APIM_E EVENT_RECEIVER deployed successfully
[2019-11-20 09:03:40,973] INFO {org.wso2.carbon.kernel.internal.CarbonStartupHandler} - WS02 API Manager Analytics Server started in 5.111 sec
[2019-11-20 09:04:38,525] INFO {org.wso2.carbon.databridge.core.DataBridge} - user admin connected
```

- 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**.



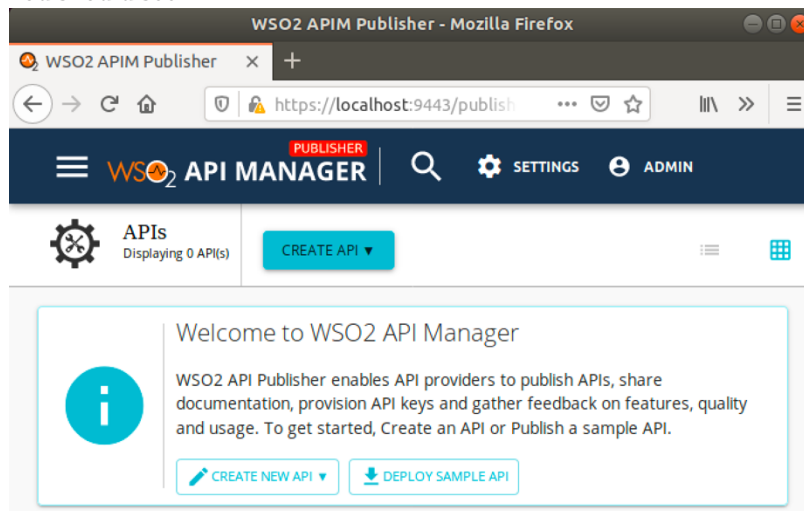
- 8) You should see something like this:



A sign-in form with a blue header containing the text "SIGN IN". Below the header, there are two input fields: "Username" and "Password". At the bottom of the form, there is a checkbox labeled "Remember me on this computer".

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

You should see:



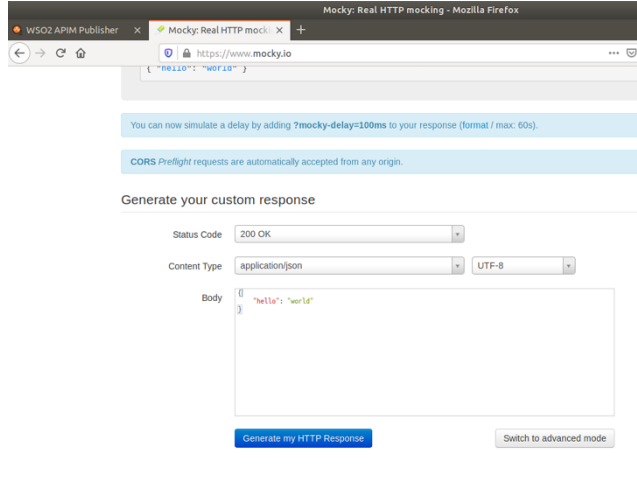
- 10) An API creator uses the **API Publisher** system to create and publish APIs into the **API Store (Developer Portal)**. Firstly we will follow the tutorial to create a managed API. Then we will try it out using the API Store.

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**

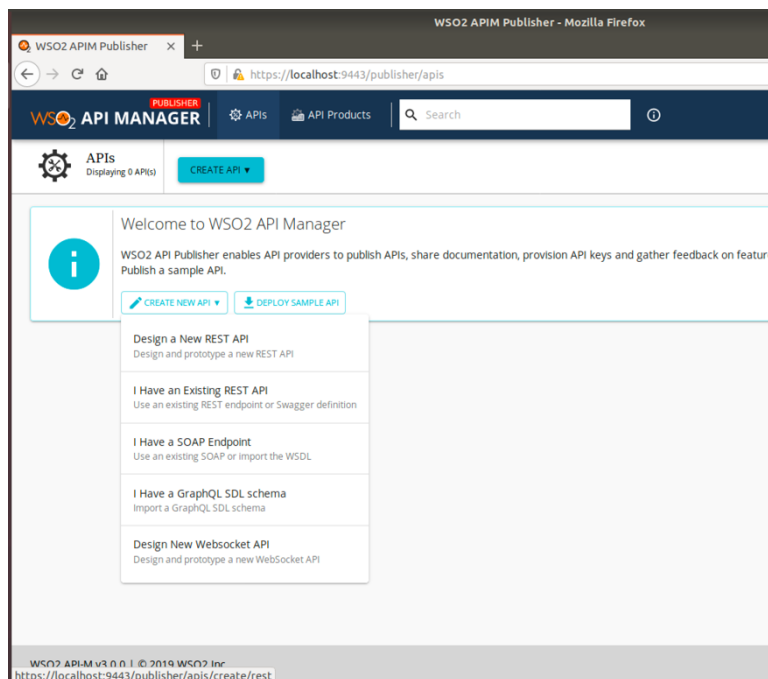
This creates a new URI that acts as your API.

You should see something like:

Your link is ready: <http://www.mocky.io/v2/5dd5678b3300008700f3813f>

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

**Create an API**  
Create an API by providing a Name, a Version, a Context, Backend Endpoint(s) (optional), and Business Plans (optional).

Name\*  
Hello

Context\*  
/hello

Version\*  
1.0.0

API will be exposed in /hello/1.0.0 context at the gateway

Endpoint  
http://www.mocky.io/v2/5dd5678b3300008700f3813f ✓

Business plan(s)  
Bronze, Gold, Silver ▼

Select one or more throttling policies for the API

\* Mandatory fields

CREATE CREATE & PUBLISH CANCEL

## 16) Now click **Create and Publish**

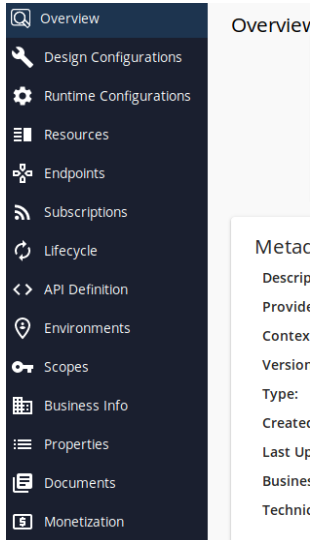
Your screen should look like:

The screenshot shows the WSO2 API Manager interface. The browser address bar displays the URL: <https://localhost:9443/publisher/apis/013c53a4-d33d-4e2b-acab-1a8f1274cd01/overview>. The page title is "WSO2 API Manager Publisher". The main content area shows the "Overview" of the API "Hello : 1.0.0", created by "admin". The API is in the "PUBLISHED" state. A progress bar indicates the lifecycle: "Created" (checked), "Endpoint" (checked), "Business plans" (checked), and "Published" (checked). The "Published" step includes a link to "View in Dev Portal". Below the progress bar, there are two tables: "Metadata" and "Configuration".

Metadata	
Description	-
Provider	admin
Context	/hello
Version	1.0.0
Type	HTTP
Created Time	A few seconds ago
Last Updated Time	A few seconds ago
Business Owner	-
Technical Owner	-

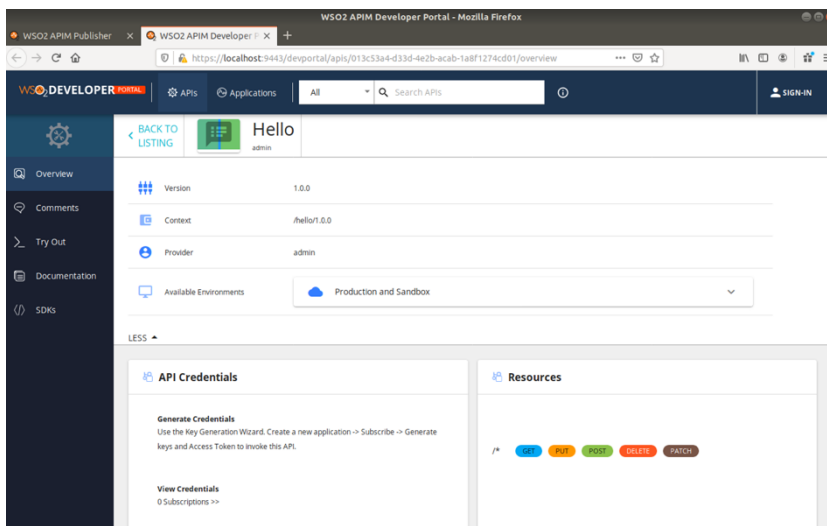
Configuration	
Transports	HTTP, HTTPS
API Security	OAuth2
Access Control	None
Workflow Status	-
Visibility on Developer Portal	Public
Business Plans	Bronze, Gold, Silver
Tags	-

- 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.

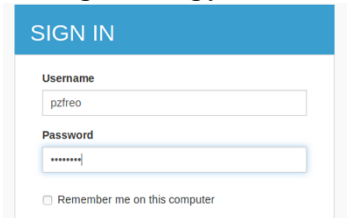
For example, these portals are based on this technology:

<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.

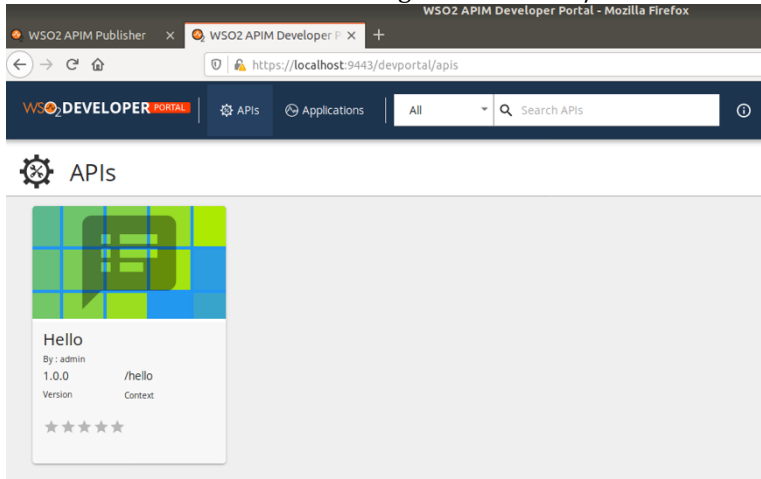
*PS the system can also be configured to use Github/Twitter/Google credentials using an OAuth2 flow.*

21) Now sign in using your new credentials.



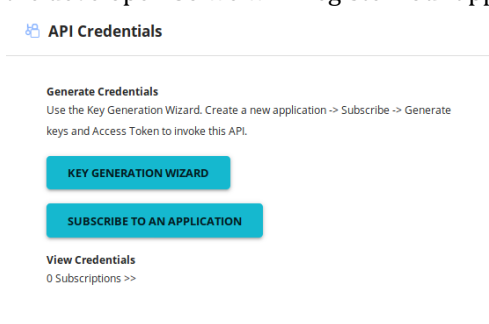
A screenshot of the 'SIGN IN' form in the WSO2 Developer Portal. It features a blue header with the text 'SIGN IN'. Below the header, there are two input fields: 'Username' with the value 'pzfreeo' and 'Password' with masked characters. A checkbox labeled 'Remember me on this computer' is located below the password field.

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:

### Key Generation Wizard

1 Create application — 2 Subscribe to new application — 3 Generate Keys — 4 Generate Access Token — 5 Copy Access Token

Application Name \*

HelloApp

Enter a name to identify the Application. You will be able to pick this application when subscribing to APIs

Per Token Quota \*

10PerMin

Assign API request quota per access token. Allocated quota will be shared among all the subscribed APIs of the application.

Token Type \*

JWT

Select token type

Application Description

My Hello App

Describe the application

CANCEL NEXT

Click Next

26) Now you can choose a subscription level

### Key Generation Wizard

1 Create application — 2 Subscribe to new application — 3 Generate Keys — 4 Gener

Application

HelloApp

Select an Application to subscribe

Throttling Policy

Bronze

Available Policies - Bronze,Gold,Silver

CANCEL NEXT

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**.

Key Generation Wizard

1 Create application — 2 Subscribe to new application — 3 Generate Keys — 4 Generate Access Token — 5 Copy Access Token

PRODUCTION SANDBOX

Grant Types

☒ Refresh Token ☒ SAML2 ☐ Implicit ☒ Password ☒ Client Credentials ☒ IWA-NTLM ☐ Code ☒ JWT

The application can use the following grant types to generate Access Tokens. Based on the application requirement you can enable or disable grant types for this application.

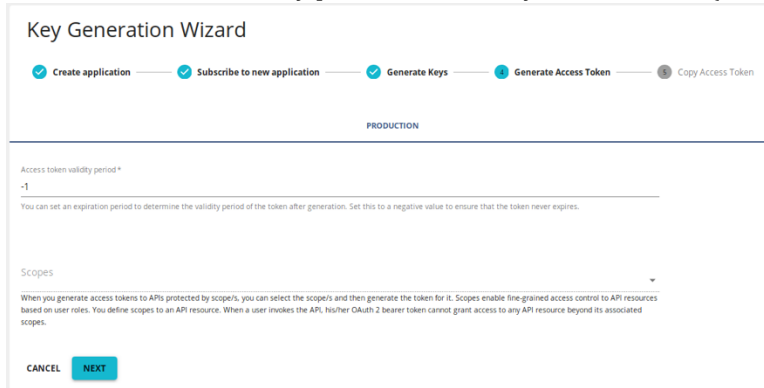
Callback URL

Callback URL is a redirection URI in the client application which is used by the authorization server to send the client's user-agent (usually web browser) back after granting access.

CANCEL NEXT



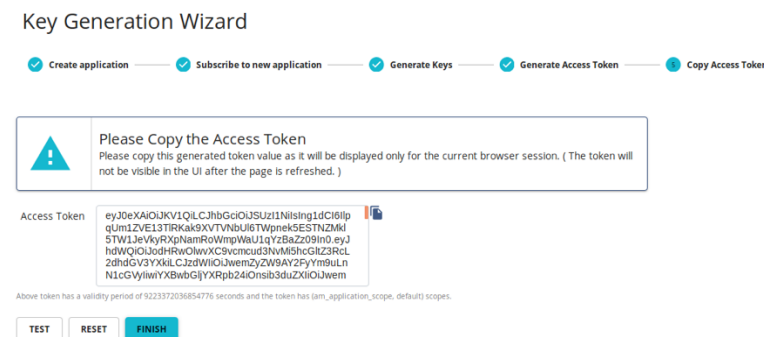
28) Set the access time validity period to -1 so they don't time out (since this is just for testing).



The screenshot shows the 'Key Generation Wizard' interface. At the top, a progress bar indicates the steps: 'Create application' (checked), 'Subscribe to new application' (checked), 'Generate Keys' (checked), 'Generate Access Token' (current step, highlighted in blue), and 'Copy Access Token' (disabled). Below the progress bar, the environment is set to 'PRODUCTION'. The main form area has a label 'Access token validity period \*' with a value of '-1'. A note below states: 'You can set an expiration period to determine the validity period of the token after generation. Set this to a negative value to ensure that the token never expires.' There is a 'Scopes' section with a dropdown menu. At the bottom, there are 'CANCEL' and 'NEXT' buttons.

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.



The screenshot shows the 'Key Generation Wizard' interface at the final step. The progress bar now shows 'Copy Access Token' as the active step. A message box says: 'Please Copy the Access Token. Please copy this generated token value as it will be displayed only for the current browser session. ( The token will not be visible in the UI after the page is refreshed. )'. Below this, the 'Access Token' is displayed as a long alphanumeric string. A small copy icon is visible next to the token. A note below the token states: 'Above token has a validity period of 9223372036854776 seconds and the token has (am\_application\_scope, default) scopes.' At the bottom, there are 'TEST', 'RESET', and 'FINISH' buttons.

Click **Test**

30) Paste the access token into the right place:

Applications

HelloApp

Please select an applications

Key Type

PRODUCTION

Please select an key type

Environment

Production and Sandbox

Please select an environment

Access Token

Authorization: Bearer OJJSVUh4Z1I1Wnl3U041c0Z2dU5yc3BOSVI1aWthlwIZ

Enter access Token

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

**default**

GET /\*

Try it out

Parameters

No parameters

Responses

Code	Description	Links
200	OK	No links

Click **Try it Out**

32) Click **Execute**

**Server response**

Code	Details
Undocumented	TypeError: NetworkError when attempting to fetch resource.

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!

```
Responses

Curl
curl -X GET "https://localhost:8243/hello/1.0.0" -H "accept: */*" -H "Authorization: Bearer eyJ0eXAiOiJKV1QiCjhbGciOiJSUzI1NiIsIngidCI6IgpUm1ZVE13TlRkak9XVTVNbUl6TWpnek5ESTNZMk15TW1JeVkyRmNamRowmpWaU1qYzBaZz09In0.eyJhdWQiOiJodHRwO1wvXc9vcmcuZ3NmV5hc6ltZ3RlZ2hdGV3YXkiCjZwdi0iJwemZyZW9hZDYyFyYm9uLnN1cGVyIiwiaXBwbgJlYXRpb24iOmsib3duZC1iOiJwemZyZW81LCJ0YWVyaWYjoMTBQZCJm4I1CjUwYWI1Ij0iS0VsbnB9bG9CChAilCjPzCIGMywidXVpZC16bnVsbH0sInNjb3B1IjoiYW1YFyXBwbgJlYXRpb25fc2NvcGUGzGVmYXVzdC1sImIzcyI6Imh0dH8zO1wvXc9sb2Nhbgchvc3Q6OT08M1wvbnB2F1dGgyXC90b2Z1biIsInRpZCJlbnZ1bjp7Ikjyb25hZ25lIeyJ2dG9wT25RdW90YVJlYmNoIj0p0cnVLLCjZcglrZUFycmVzdExpbnw10IjowLjCjZcglrZUFycmVzdFVuaXQ10m51bGx9f5wia2V5dHlwZSI6I1B5TORV01RjT04lCjZwdjY3Zj0YmVkb0VBJ3cvI6Bw3sic3Vic2l0aW4uZ3RlRmFudERvbnRvbnB1Ij0iImNmcmVb51zdXBlciIsIm5hbWU10iJ2NxsbnB1ImVbnRlEH0i01jclZhbGxvXc9XClA
```

```
Request URL
https://localhost:8243/hello/1.0.0

Server response

Code Details
200 Response body
{
  "hello": "world"
}
```

Download

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:

Code	Details
429 <i>Undocumented</i>	<p>Error:</p> <p><b>Response body</b></p> <pre>&lt;amt:fault xmlns:amt="http://ws2.org/apimanager/throttling"&gt;   &lt;amt:code&gt;900803&lt;/amt:code&gt;   &lt;amt:message&gt;Message throttled out&lt;/amt:message&gt;   &lt;amt:description&gt;You have exceeded your quota&lt;/amt:description&gt;   &lt;amt:nextAccessTime&gt;2019-Nov-20 17:42:00+0000 UTC&lt;/amt:nextAccessTime&gt; &lt;/amt:fault&gt;</pre> <p><b>Response headers</b></p> <pre>content-type: application/xml; charset=UTF-8</pre>

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
```

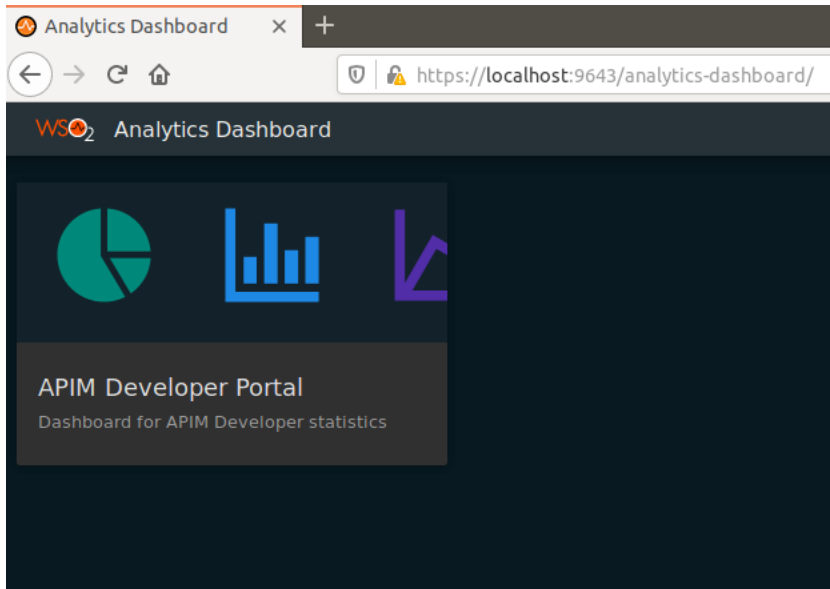
41) Let's look at the dashboard page: <https://localhost:9643/analytics-dashboard>

Go through the certificate approval again.

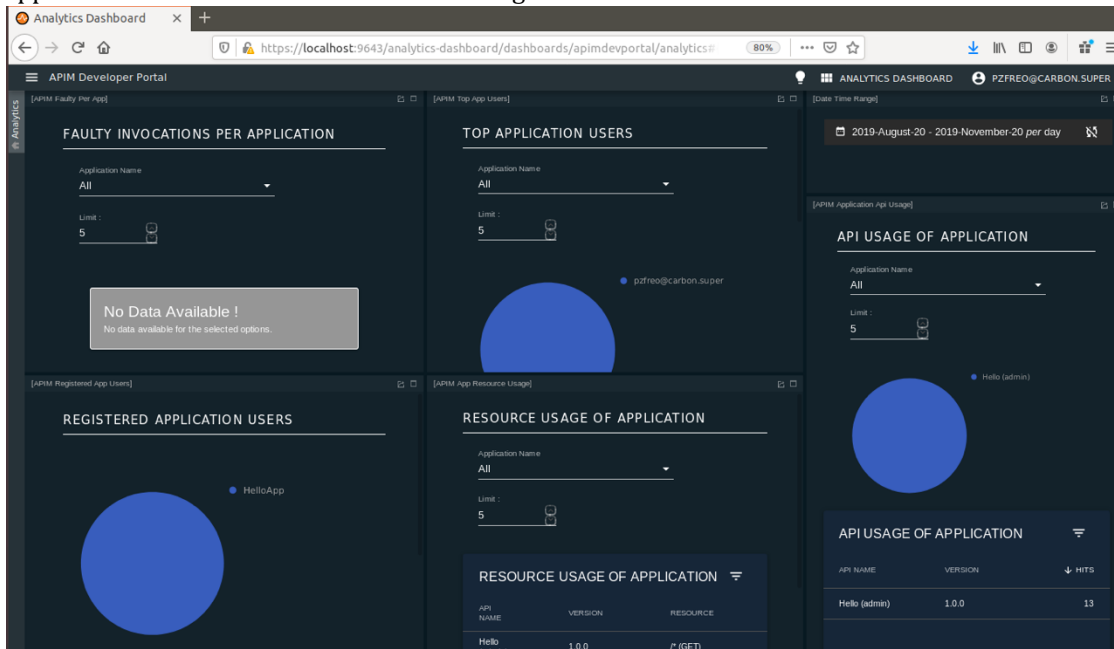
You will be prompted to login. You can login as either your subscriber or your publisher (admin).

Login as the subscriber (not admin) first.

You should see something like:

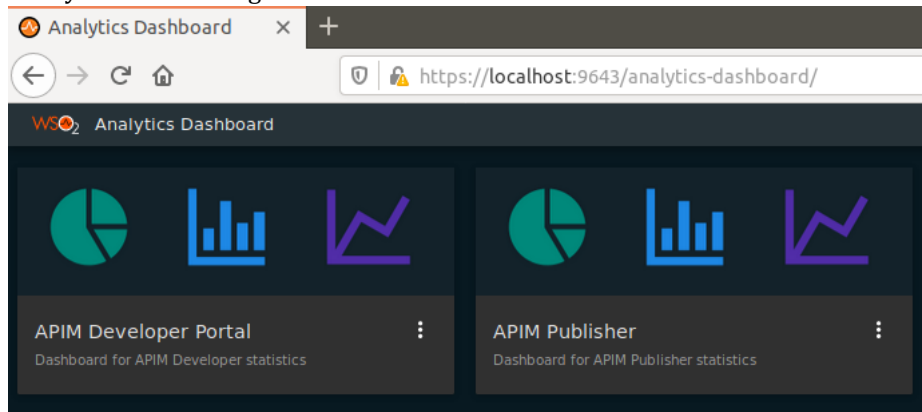


These are dashboards aimed at developers – allowing them to see the usage of APIs by their applications. Take a look at the available widgets.



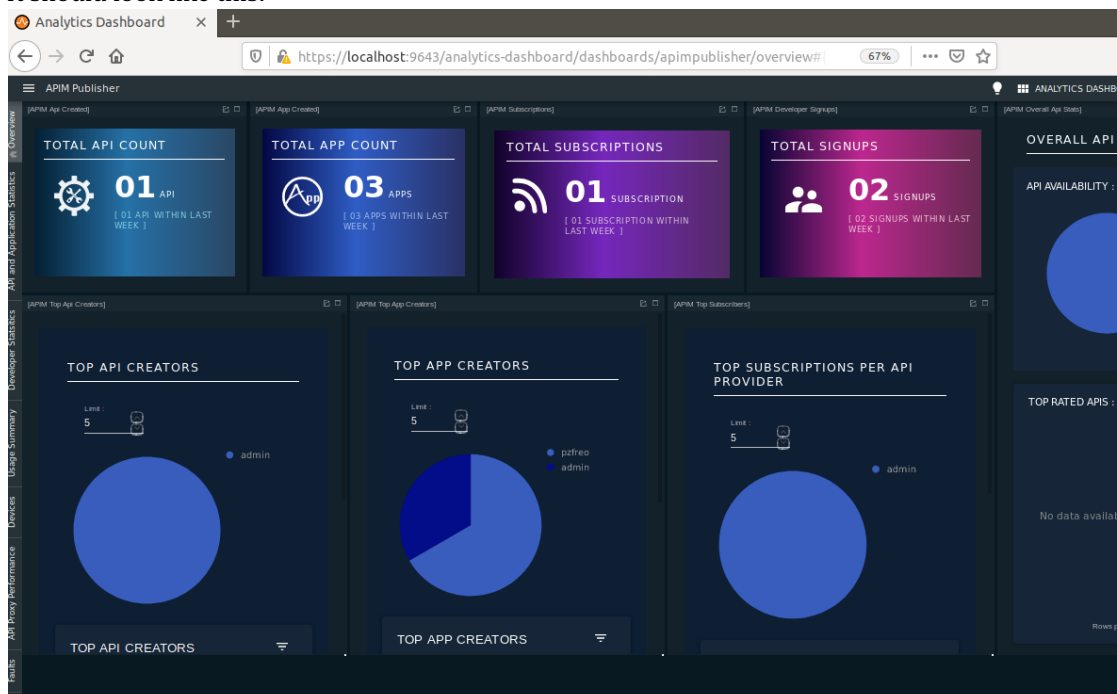
42) Now logout and log back in as **admin**.

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

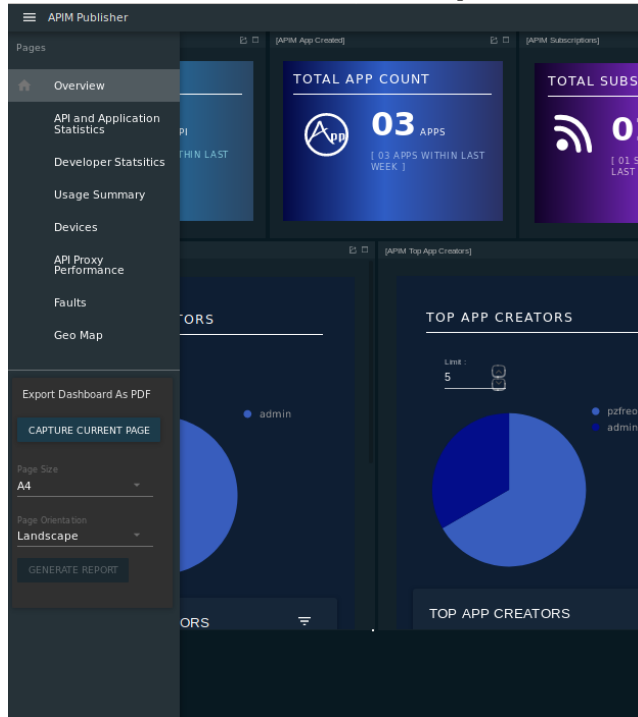


44) Click on APIM Publisher

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"**

Create an API using an OpenAPI definition.

Create an API using an existing OpenAPI definition (swagger) file or URL.

1 Provide OpenAPI 2 Create API

\* Input Type

☒ OpenAPI URL

☐ OpenAPI File

OpenAPI URL

Enter OpenAPI URL

Click away to validate the URL

CANCEL NEXT

51) Use **OpenAPI URL**

<http://localhost/openapi.yaml>

then tab away from the box to validate the URL

You should see a nice tick:

OpenAPI URL

http://localhost/openapi.yaml

Click away to validate the URL

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.

1 Provide OpenAPI 2 Create API

Name\*  
PurchaseAPI

Context\*  
/purchase

Version\*  
0.0.2

API will be exposed in /purchase/0.0.2 context at the gateway

Endpoint  
http://localhost ✓

Business plan(s)  
Bronze, Gold, Silver, Unlimited ▼

Select one or more throttling policies for the API

\* Mandatory fields

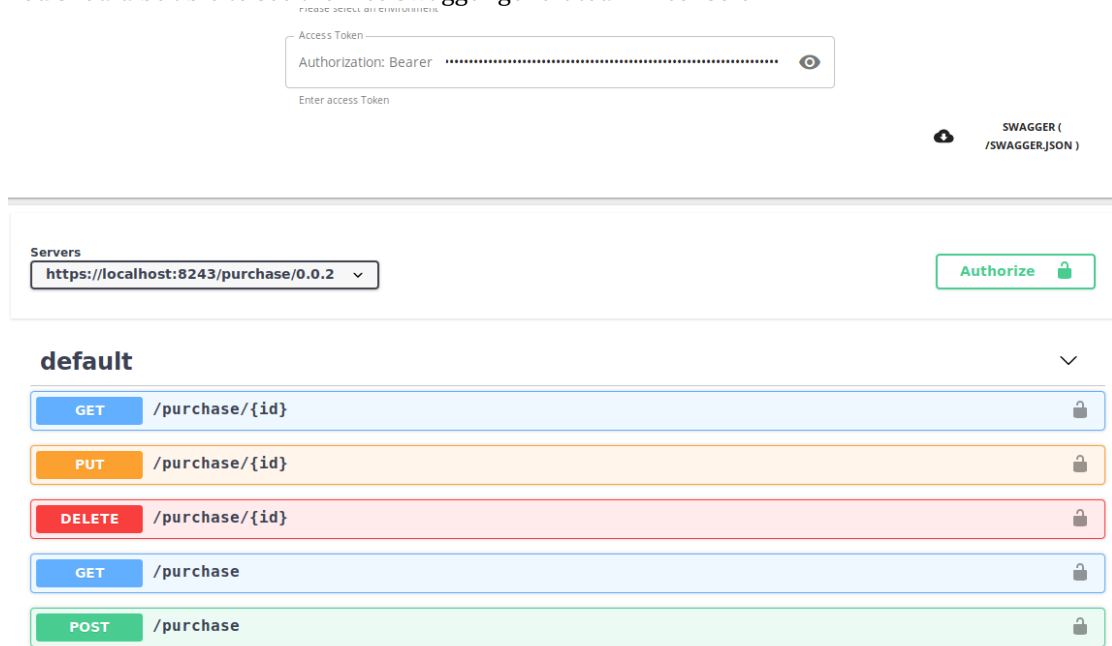
BACK CREATE

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.