

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
```



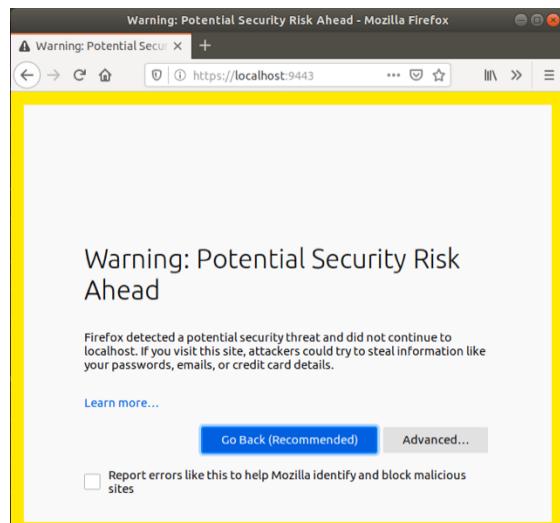
bin/wso2server.sh

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

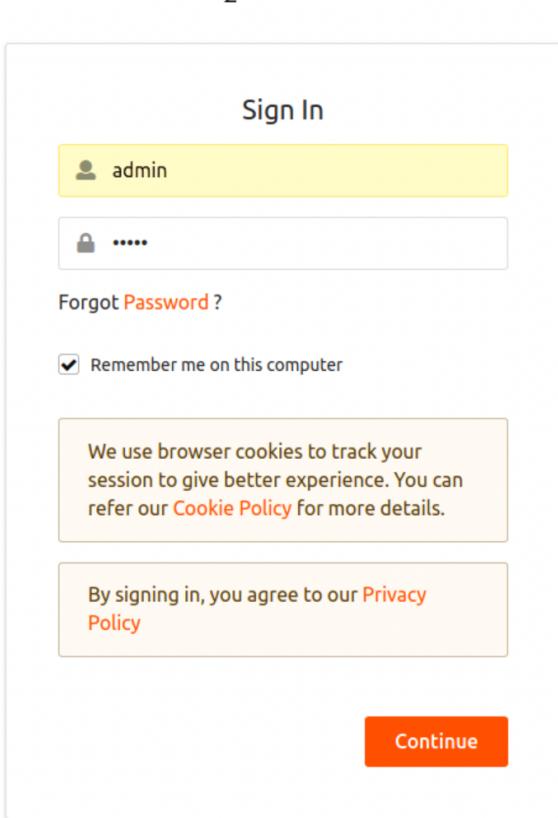
- 6) Once both servers are started, check that you can access the web interface of the API Manager

 - 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**.





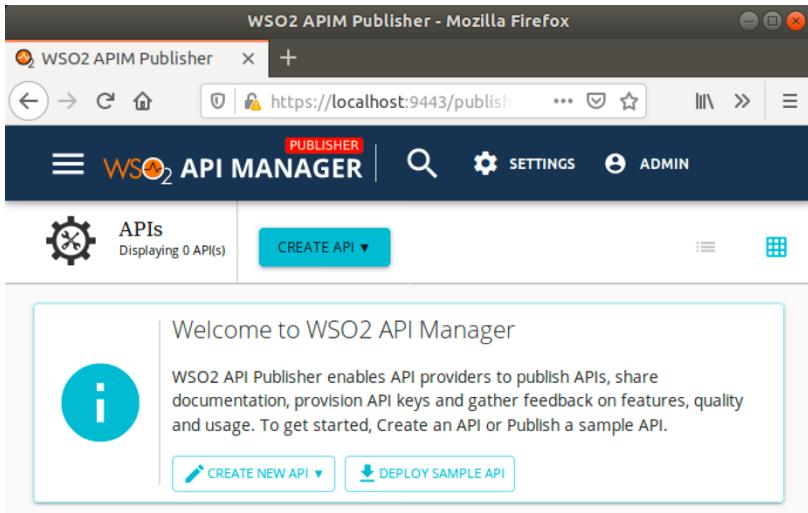
- 7) You should see something like this:



The screenshot shows the WSO2 API Manager Sign In page. At the top is the WSO2 logo and the text "API MANAGER". Below that is a "Sign In" heading. A yellow box contains the text "admin" next to a user icon. Below it is a password field with a lock icon and the placeholder ".....". To the right of the password field is a link "Forgot Password?". Below the password field is a checked checkbox labeled "Remember me on this computer". A callout box contains the text: "We use browser cookies to track your session to give better experience. You can refer our [Cookie Policy](#) for more details." Another callout box contains the text: "By signing in, you agree to our [Privacy Policy](#)". At the bottom is a large orange "Continue" button.

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

You should see:



The screenshot shows the WSO2 API Manager Publisher interface in Mozilla Firefox. The title bar says "WSO2 APIM Publisher - Mozilla Firefox". The main header has tabs for "PUBLISHER", "SETTINGS", and "ADMIN". On the left, there's a gear icon and the text "APIs Displaying 0 API(s)". A blue button says "CREATE API ▾". The main content area has a teal sidebar with an info icon and the text "Welcome to WSO2 API Manager". It says "WSO2 API Publisher enables API providers to publish APIs, share documentation, provision API keys and gather feedback on features, quality and usage. To get started, Create an API or Publish a sample API.". Below this are two buttons: "CREATE NEW API ▾" and "DEPLOY SAMPLE API".

- 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:

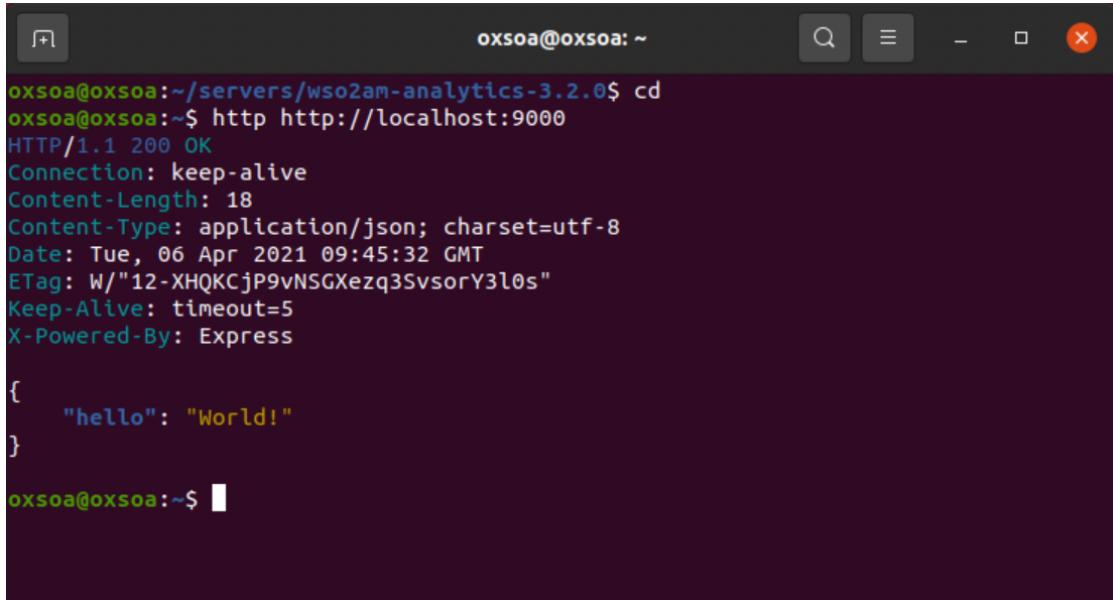
```
docker run -d -p 9000:8000 pzfre/o/hello-api:1.0.0
```

(Source code is here: <https://github.com/pzfre/o/hello-api.git>)

Notice we are exposing the port on 9000

11) Test it out to see how it works:

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



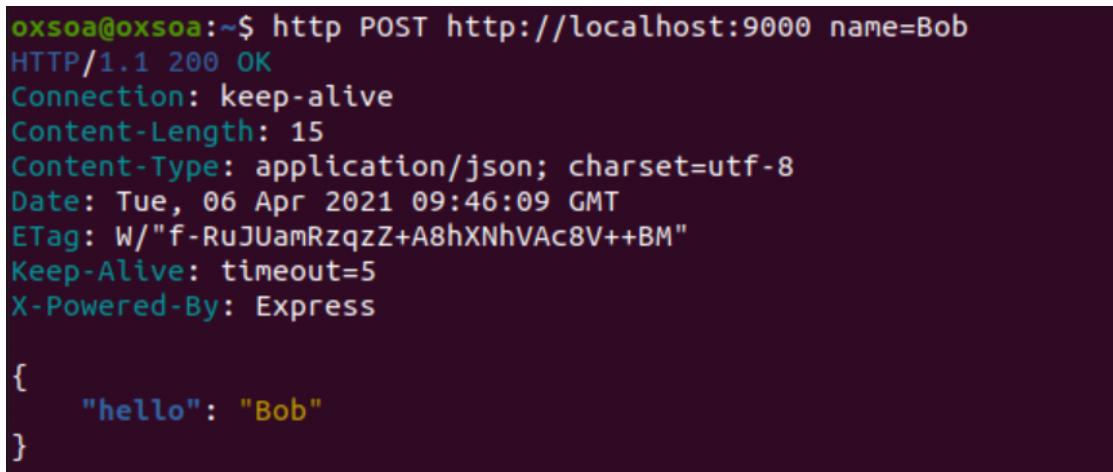
```
oxsoa@oxsoa:~/servers/wso2am-analytics-3.2.0$ cd
oxsoa@oxsoa:~$ http http://localhost:9000
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 18
Content-Type: application/json; charset=utf-8
Date: Tue, 06 Apr 2021 09:45:32 GMT
ETag: W/"12-XHQKCjP9vNSGXezq3SvsorY3l0s"
Keep-Alive: timeout=5
X-Powered-By: Express

{
    "hello": "World!"
}

oxsoa@oxsoa:~$
```

And

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



```
oxsoa@oxsoa:~$ http POST http://localhost:9000 name=Bob
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 15
Content-Type: application/json; charset=utf-8
Date: Tue, 06 Apr 2021 09:46:09 GMT
ETag: W/"f-RuJUamRzqzZ+A8hXNhVAc8V++BM"
Keep-Alive: timeout=5
X-Powered-By: Express

{
    "hello": "Bob"
}
```



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

The screenshot shows the WSO2 API Manager Publisher interface in Mozilla Firefox. The URL is https://localhost:9443/publisher/apis. The top navigation bar has tabs for 'PUBLISHER', 'APIs', 'API Products', and a search bar. Below the navigation is a sub-header with 'APIs' and 'DISPLAYING 0 API(s)' followed by a 'CREATE API' button. The main content area is titled 'Welcome to WSO2 API Manager' and contains a large 'i' icon. It lists several options for creating APIs: 'Design a New REST API' (selected), 'I Have an Existing REST API', 'I Have a SOAP Endpoint', 'I Have a GraphQL SDL schema', and 'Design New Websocket API'. At the bottom of the page, it says 'WSO2 API Manager v2.0.0 | © 2019 WSO2 Inc.' and the URL 'https://localhost:9443/publisher/apis/create/rest'.

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.



Create an API

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

Name*

Context* Version*

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

Endpoint

Business plan(s)

Select one or more throttling policies for the API

* Mandatory fields

Create **Create & Publish** **Cancel**

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

The screenshot shows the WSO2 API Manager interface with the following details:

- API Overview:** Hello :1.0.0 (PUBLISHED)
- Configuration:**
 - Transports: HTTP, HTTPS
 - API Security: OAuth2
 - Access Control: None
 - Workflow Status: -
 - Visibility on Developer Portal: Public
 - Business Plans: Bronze, Gold, Silver
 - Tags: -
- Endpoints:**
 - Production Endpoint: http://localhost:9000/
 - Sandbox Endpoint: http://localhost:9000/
 - Endpoint Security: -
- Resources:** GET, POST, PUT, DELETE, PATCH



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

The screenshot shows a dark-themed user interface for managing APIs. On the left is a vertical sidebar with the following items:

- Overview
- Design Configurations
- Runtime Configurations
- Resources
- Endpoints
- Subscriptions
- Lifecycle
- API Definition
- Environments
- Scopes
- Business Info
- Properties
- Documents
- Monetization

The main content area is titled "Overview" and contains a sub-section titled "Metadata". The "Metadata" section includes fields for Description, Provider, Context, Version, Type, Created, Last Updated, Business, and Technical.

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:

The screenshot shows the WSO2 Developer Portal interface. At the top, there's a blue header bar with the WSO2 logo, 'DEVELOPER PORTAL', and navigation links for 'APIs' and 'Applications'. A search bar says 'Search APIs' and a dropdown says 'All'. On the right, there's a user icon labeled 'ADMIN'. The main content area has a dark sidebar on the left with icons for 'Overview', 'Subscriptions', 'Try Out', 'Comments', 'Documentation', and 'SDKs'. The 'Overview' tab is selected. To its right, the 'Hello' API is displayed. The API details include:

- Version:** 1.0.0
- Context:** /hello/1.0.0
- Provider:** admin
- Technical Owner:** (empty)
- Key Managers:** All Applicable
- Rating:** ★★★★☆ (with a crossed-out star)
- Gateway Environments:** Production and Sandbox

Below this, there are two sections: 'Subscriptions' and 'Resources'. The 'Subscriptions' section says 'No application subscriptions.' and has a 'SUBSCRIBE' button. The 'Resources' section shows a list of endpoints with their HTTP methods: /* (GET, PUT, POST, DELETE, PATCH).

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**.

WSO₂ API MANAGER

Start Signing Up

Enter your username here

Username

If you do not specify a tenant domain, you will be registered under super tenant

[Cancel](#) [Proceed to Self Register](#)

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

WSO₂ API MANAGER

Create New Account

Fill in the form below to complete registration

First Name *

Last Name *

Password *

Confirm password *

Email *

Organization

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

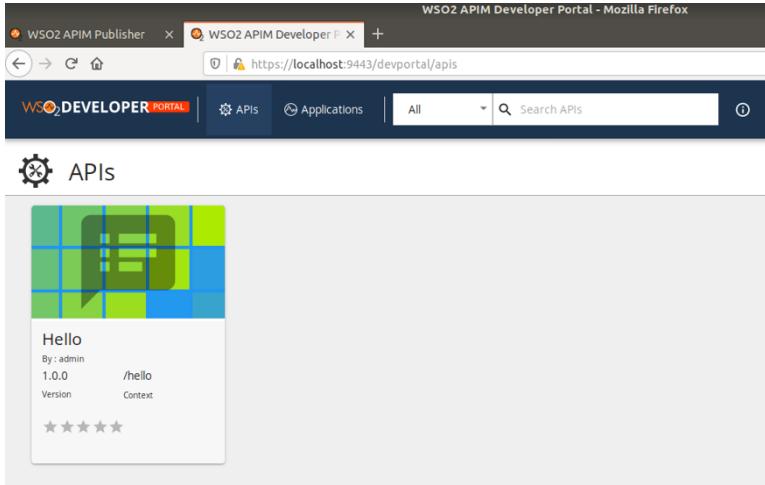


17) Now sign in using your new credentials.



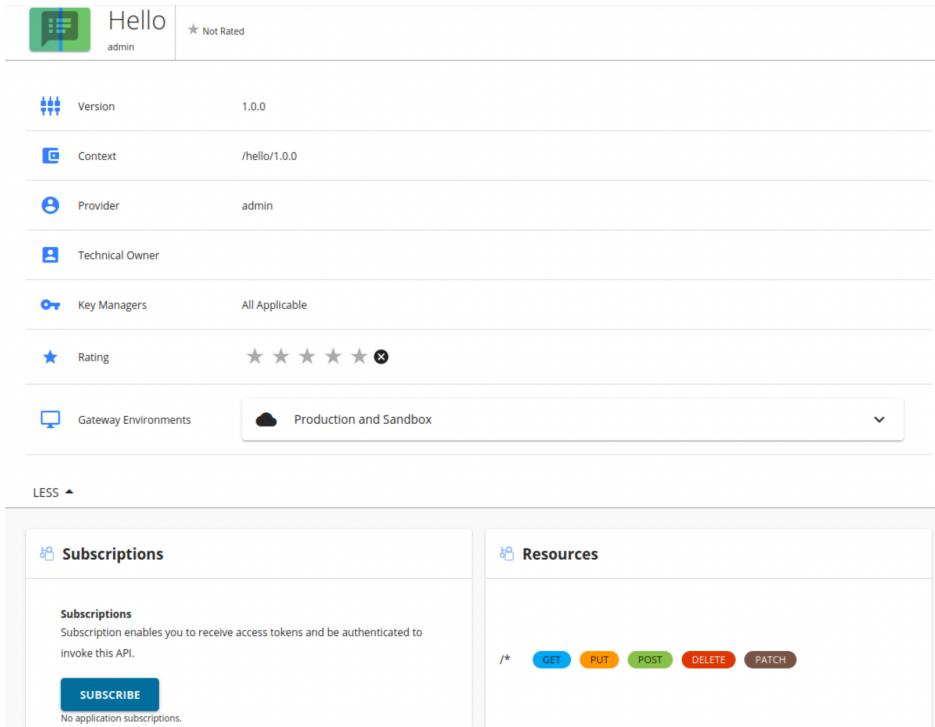
The screenshot shows the WSO2 API Manager Sign In page. It has a blue header bar with the text "SIGN IN". Below it is a form with two input fields: "Username" containing "pzfreo" and "Password" containing ".....". There is also a checkbox labeled "Remember me on this computer".

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



The screenshot shows the WSO2 Developer Portal. The browser title bar says "WSO2 APIM Publisher" and "WSO2 APIM Developer Portal - Mozilla Firefox". The URL is https://localhost:9443/devportal/apis. The main page displays a single API entry titled "Hello". The "Hello" API is version 1.0.0, provided by "admin", and its context is "/hello". It has a 5-star rating. The interface includes tabs for "APIs" and "Applications", a search bar, and a "All" dropdown.

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



The screenshot shows the detailed view of the "Hello" API. At the top, there is a summary card with the API name "Hello", provider "admin", and rating "Not Rated". Below this, a table lists various metadata: Version (1.0.0), Context (/hello/1.0.0), Provider (admin), Technical Owner, Key Managers (All Applicable), and Rating (5 stars). A "Gateway Environments" section indicates "Production and Sandbox". At the bottom, there are two sections: "Subscriptions" (with a note about subscriptions and a "SUBSCRIBE" button) and "Resources" (with a list of HTTP methods: GET, PUT, POST, DELETE, PATCH).

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**

An application is primarily used to decouple the consumer from the APIs. It allows you to generate and use a single key for multiple APIs and subscribe multiple times to a single API with different SLA levels.

Subscribe

Subscribe to an application and generate credentials

Application: DefaultApplication Throttling Policy: Bronze

Select an Application to subscribe Available Policies - Bronze,Gold,Silver

SUBSCRIBE

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

23) You should see:

Subscription & Key Generation Wizard

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

Application Name * My Application

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.

Application Description

(512) characters remaining

CANCEL **NEXT**



24) Fill in the Form a bit like this:

Subscription & Key Generation Wizard

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

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

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

Application Description
(500) characters remaining

CANCEL **NEXT**

Click Next

25) Now you can choose a subscription level

Key Generation Wizard

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

Application Select an Application to subscribe

Throttling Policy Available Policies - Bronze,Gold,Silver

CANCEL **NEXT**

Click Next



26) Now you can choose to generate keys.

Subscription & Key Generation Wizard

The screenshot shows the 'Generate Keys' step of the wizard. It includes fields for Key Manager (Resident Key Manager), Environment (Sandbox), Token Endpoint (<https://localhost:9443/oauth2/token>), Revoke Endpoint (<https://localhost:9443/oauth2/revoke>), User Info Endpoint, and Grant Types (Refresh Token, SAML2, Password, Client Credentials, IWA-NTLM, Device Code, Code, JWT). Buttons at the bottom include 'CANCEL' and 'NEXT'.

Click Next

27) You should see

Subscription & Key Generation Wizard

The screenshot shows the 'Generate Access Token' step of the wizard. It includes fields for Scopes (empty) and a note about scopes. Buttons at the bottom include 'CANCEL' and 'NEXT'.

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.

Key Generation Wizard

The screenshot shows the 'Copy Access Token' step of the wizard. A message box says 'Please Copy the Access Token' with instructions to copy the generated token value. Below is the token value: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Ip... . Buttons at the bottom include 'TEST', 'RESET', and 'FINISH'.

Click Test



29) Paste the access token into the right place:

Security

The Resident Key Manager is selected for try out console.

Security Type

OAuth API Key Basic

Applications

HelloApp

Subscribed applications

Key Type

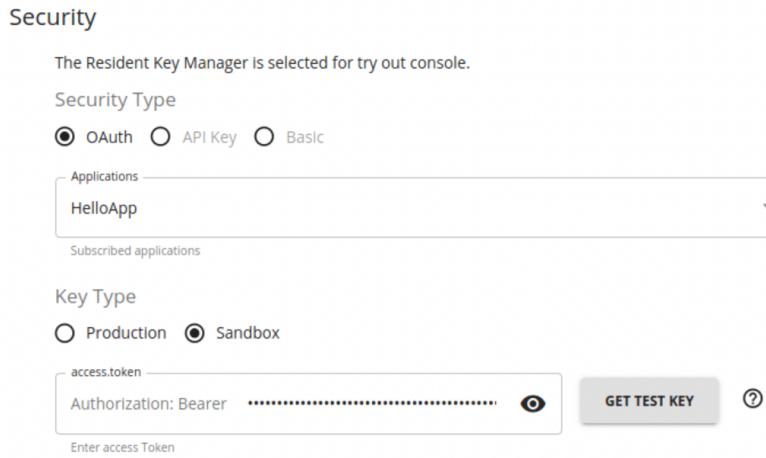
Production Sandbox

access.token

Authorization: Bearer

GET TEST KEY

Enter access Token



Gateway

Environment

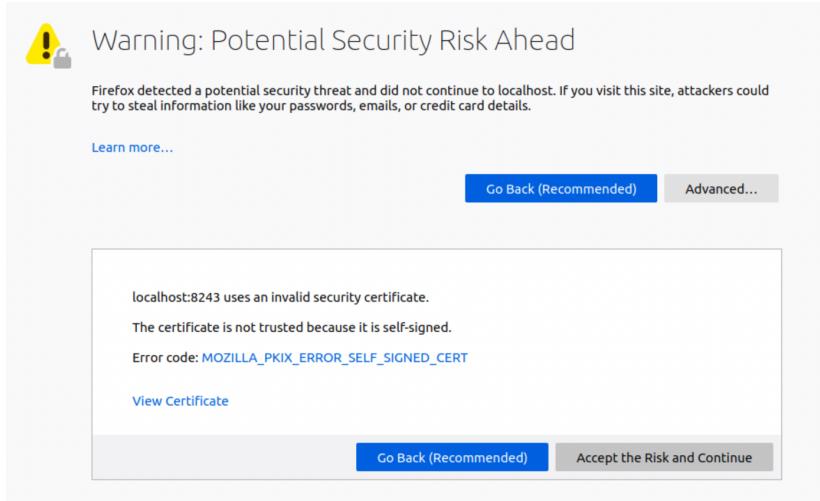
Production and Sandbox

Please select an environment



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.

default

The screenshot shows a REST API documentation interface. At the top, there is a blue button labeled "GET" and a path placeholder "/*". To the right of the path is a lock icon and a "Try it out" button. Below this, there are two sections: "Parameters" and "Responses". The "Parameters" section is titled "Parameters" and contains the message "No parameters". The "Responses" section is titled "Responses" and contains a table with three columns: "Code", "Description", and "Links". There is one row in the table for the code 200, which is described as "OK". The "Links" column for this row contains the message "No links".

Click **Try it Out**



33) Click **Execute**

The screenshot shows the API management interface with a successful response. The response body is a JSON object: { "Hello": "World!" }. The status code is 200 OK.

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

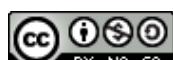
35) Keep clicking and eventually you should see the throttling happen:

The screenshot shows a 429 error response. The response body contains XML error details: <amt:fault xmlns:amt="http://wso2.org/apimanager/throttling"> <amt:code>900803</amt:code> <amt:message>Message throttled out</amt:message> <amt:description>You have exceeded your quota</amt:description> <amt:nextAccessTime>2019-Nov-29 17:42:00+0000 UTC</amt:nextAccessTime> </amt:fault>

- 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
```



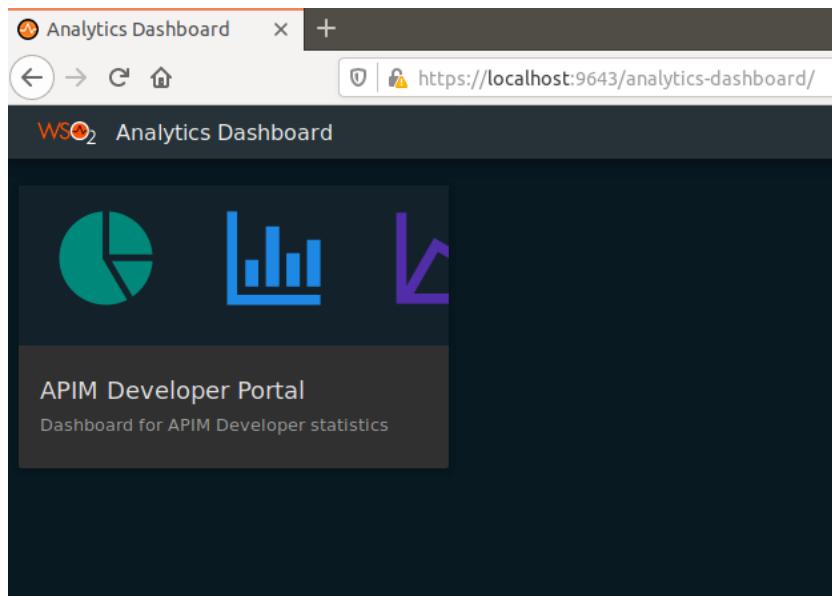
39) Let's look at the dashboard page: <https://localhost:9643/analytics-dashboard>

Go through the certificate approval again.

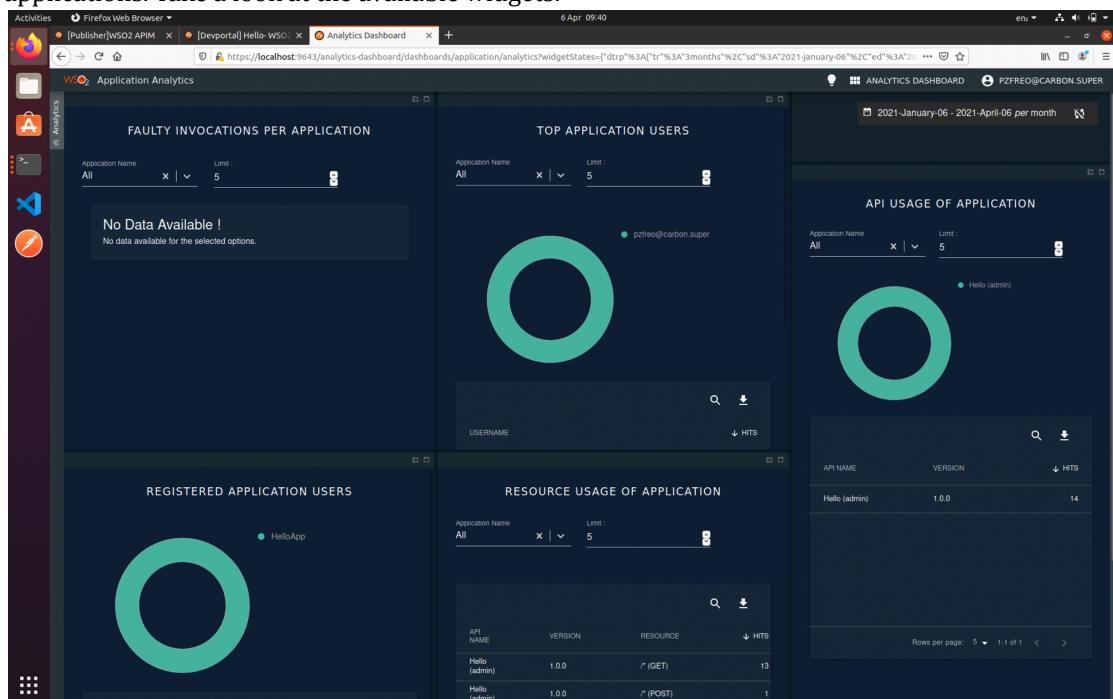
You will be prompted to login. You can login as either your subscriber or your publisher (admin).

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:

The screenshot shows the WSO2 Analytics Dashboard interface. At the top, there are four main sections: 'API Analytics' (Dashboard for APIs related analytics), 'Application Analytics' (Dashboard for application related analytics), 'Business Analytics' (Dashboard for business related analytics), and 'Monitoring' (Dashboard related to monitoring analytics). Below these, there is a section titled 'Reports' (Dashboard for reports generation). The interface has a dark theme with blue and purple icons for each section.

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

The screenshot shows the 'API Usage' section of the Analytics Dashboard. On the left, there is a sidebar with options like 'Usage Summary', 'API Usage', 'API Performance', and 'API Faults'. The main area displays 'OVERALL API USAGE' with a chart showing 'SUB_COUNT' (Y-axis, limit 5) and 'API_NAME : Hello (1.0.0)' (X-axis, count 28, created by admin). Below this is a table of API details:

API NAME	VERSION	APPLICATION	HITS
Hello (admin)	1.0.0	HelloApp (pcheo)	28

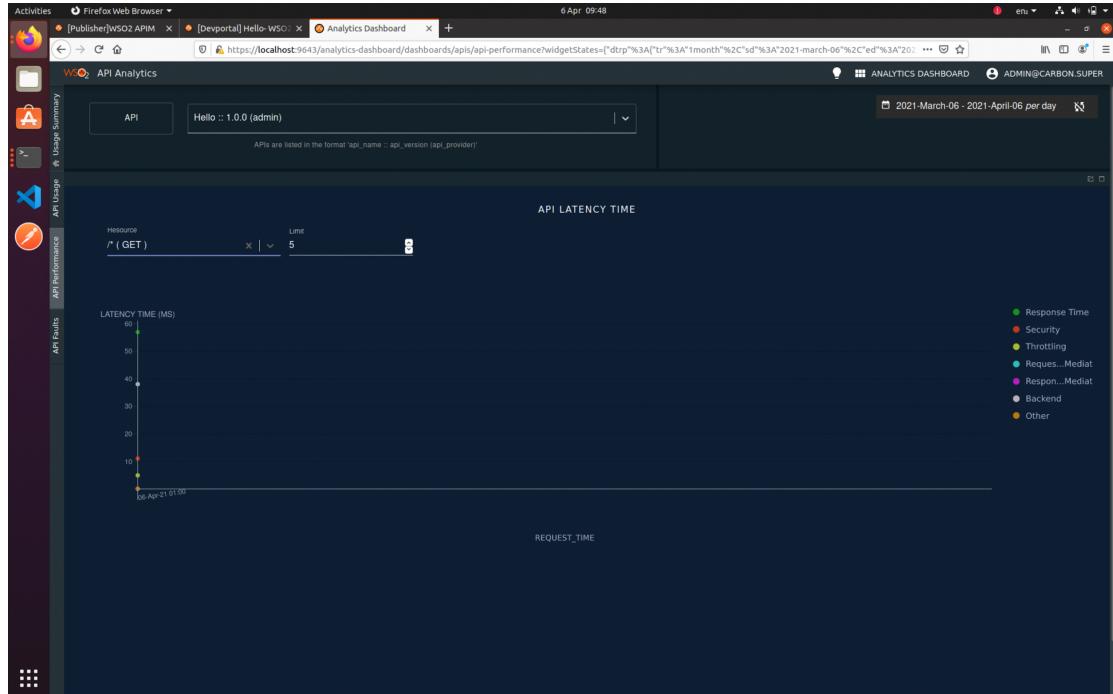
On the right, there is a section titled 'API USAGE BY APPLICATION' with a table:

API NAME	VERSION	APPLICATION	USAGE
Hello (admin)	1.0.0	HelloApp (pcheo)	28

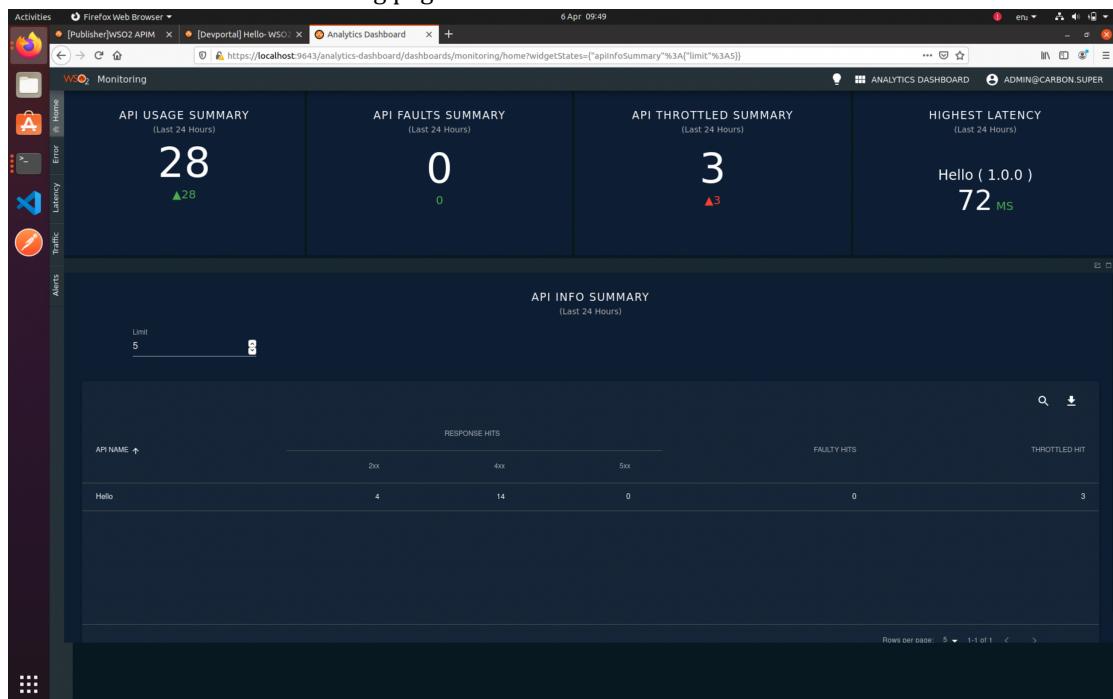
Both sections have a 'Limit' dropdown set to 5.



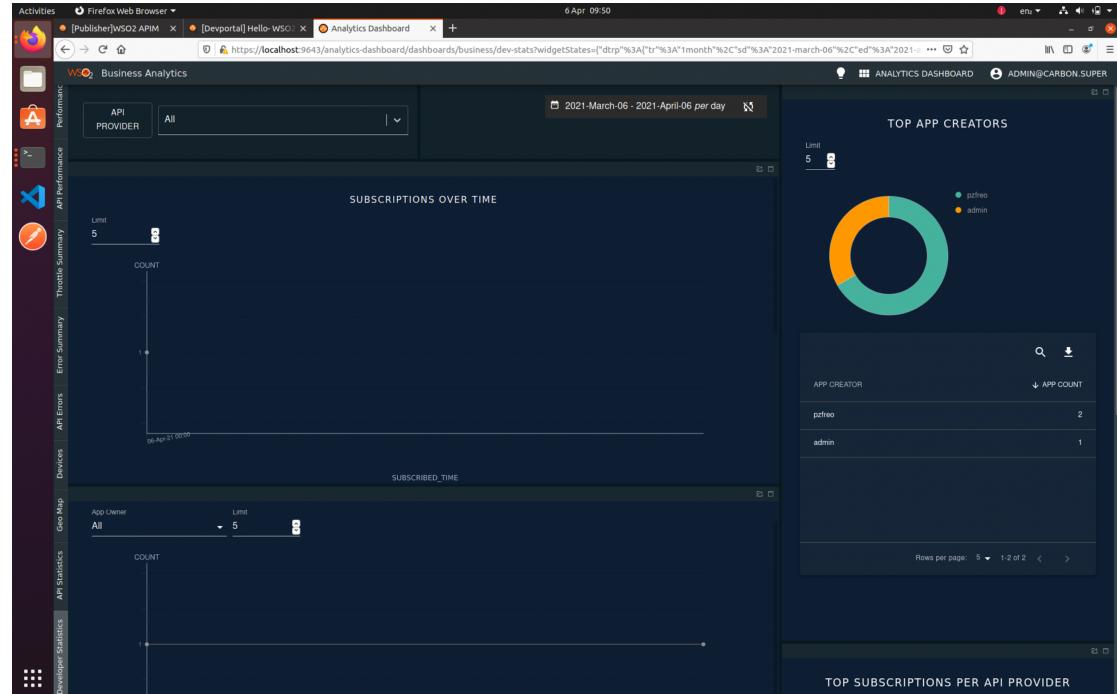
43) For example, you can view the latency of different aspects of the API:



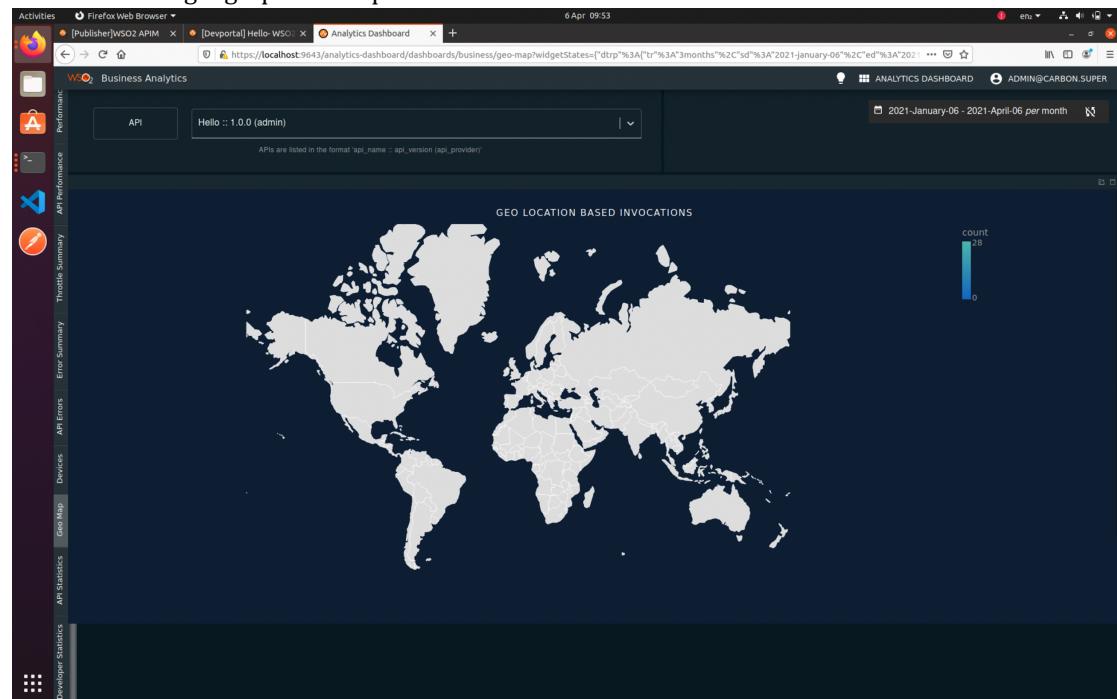
Also take a look at the monitoring page:



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**”

Create an API using an OpenAPI definition.

Create an API using an existing OpenAPI definition file or URL.

Provide OpenAPI Create API

* Input Type
 OpenAPI URL
 OpenAPI File/Archive

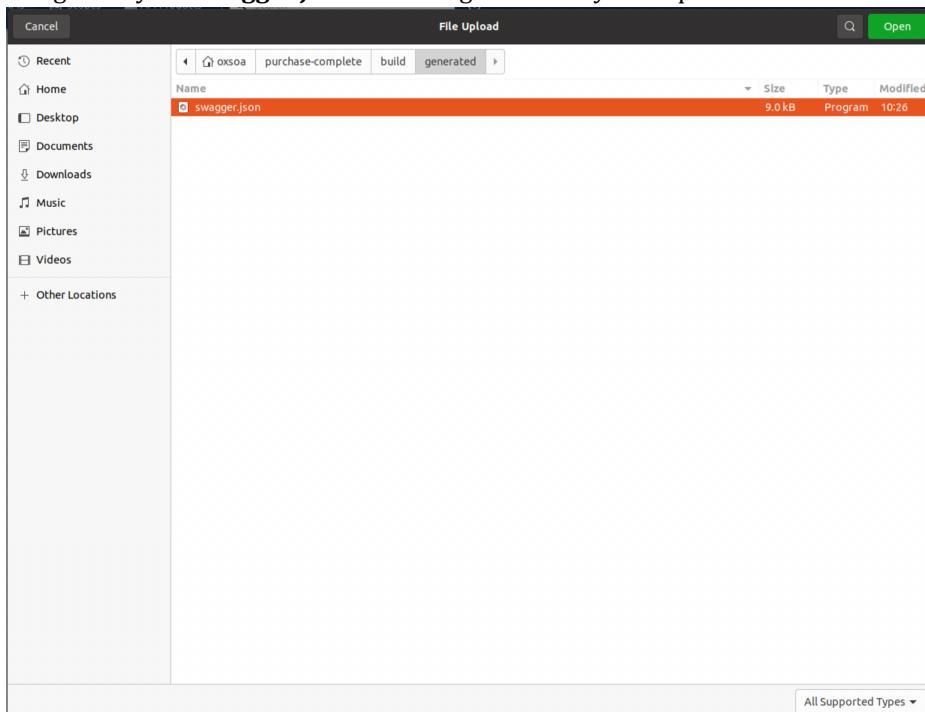
Drag & Drop Open API File/Archive here
or
Browse files
Browse File to Upload

Cancel **Next**



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.

Create an API using an OpenAPI definition.

Create an API using an existing OpenAPI definition file or URL.

1 Provide OpenAPI 2 Create API

Name*
purchase-api

Context*
/purchase

Version*
1.0.0

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

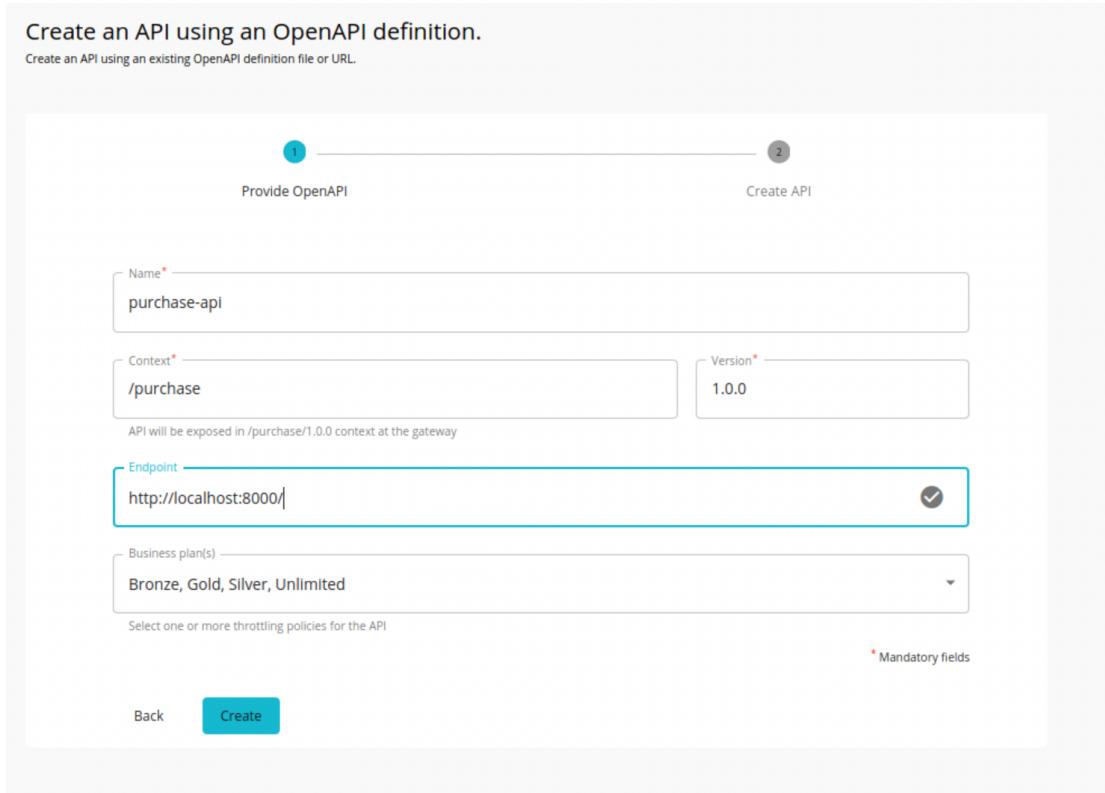
Endpoint
http://localhost:8000/

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

Select one or more throttling policies for the API

* Mandatory fields

Back Create



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

The screenshot shows the Swagger UI interface for a REST API. At the top, there is a header with fields for 'Access Token' and 'Authorization: Bearer' (with a placeholder 'Enter access Token'). To the right, it says 'SWAGGER (/SWAGGER.JSON)' with a logo. Below this, the 'Servers' dropdown is set to 'https://localhost:8243/purchase/0.0.2' and there is a green 'Authorize' button. The main area is titled 'default' and lists several operations for the '/purchase' endpoint:

- GET /purchase/{id} (blue background)
- PUT /purchase/{id} (orange background)
- DELETE /purchase/{id} (red background)
- GET /purchase (blue background)
- POST /purchase (green background)

Each operation has a small lock icon to its right.

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.

This screenshot shows a detailed view of a Swagger API response for a POST request to '/purchase'. The response includes:

- Curl:** A command-line example of the POST request.
- Request URL:** The URL for the request: <https://localhost:8243/purchase/1.0.0/purchase>.
- Server response:**
 - Code:** 201
 - Details:** Response body (JSON object), Response headers (Content-Type: application/json; charset=utf-8).
- Responses:**
 - Code:** 201
 - Description:** Created
 - Links:** No links
- Example Value:** Schema (application/json).



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

The screenshot shows the WSO2 API Manager interface with the 'purchase-api :1.0.0' API selected. The left sidebar shows various management options like Overview, Resources, Endpoints, Subscriptions, Lifecycle, API Definition, Environments, Local Scopes, Business Info, Properties, Documents, Test Console, and Monetization. The main content area displays the API Definition in YAML format. The YAML code defines an openapi 3.1.0 specification with a title 'purchase-tsoa', contact information for 'Paul Fremantle', and Apache 2.0 license. It includes a 'purchase/{uid}' endpoint with a 'get' operation, parameters for 'uid' (path, required, style: simple, explode: false, type: string), and responses for '200' (OK, application/json schema). It also includes a 'put' operation for 'UpdatePO' with parameters for 'uid' (path, optional: true).

```

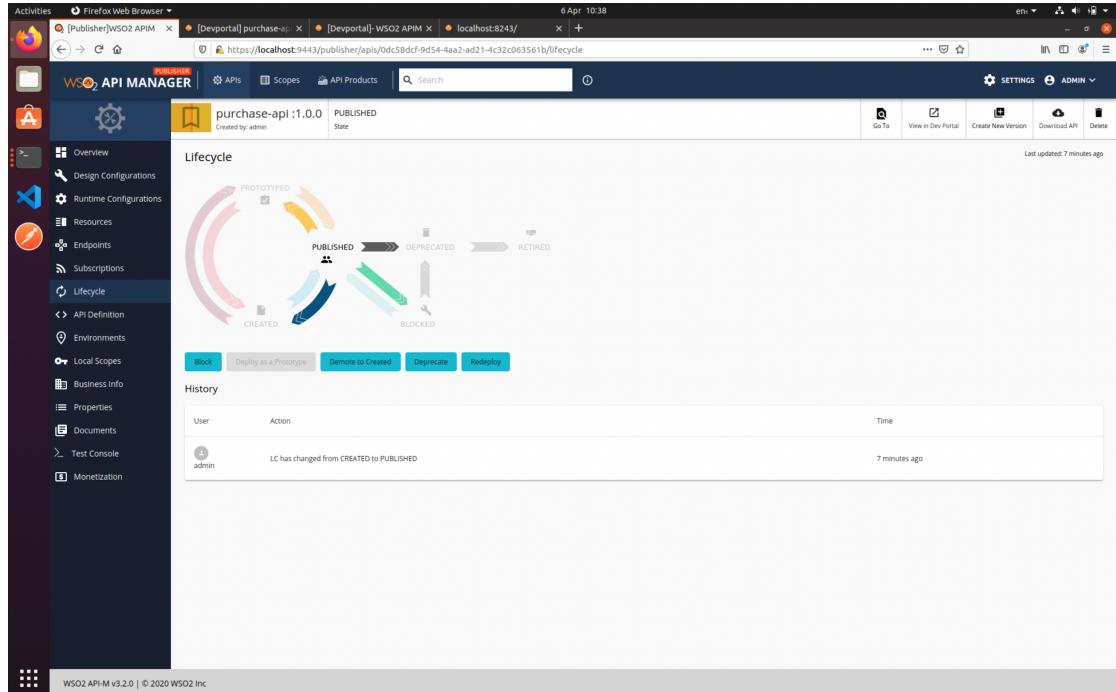
openapi: 3.1.0
info:
  title: purchase-tsoa
  contact:
    name: 'Paul Fremantle'
    email: pfreed@github.com
  license:
    name: Apache-2.0
  version: 1.0.0
servers:
  - url: /
  - url: /purchase/{uid}
  security:
    - default: []
paths:
  /purchase/{uid}:
    get:
      operationId: GetPurchase
      parameters:
        - name: uid
          in: path
          required: true
          style: simple
          explode: false
          type: string
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                anyOf:
                  - $ref: '#/components/schemas/PurchaseOrder'
                  - $ref: '#/components/schemas/ErrorReport'
      security:
        - default: []
        x-auth-type: null
        x-throttling-tier: null
        x-wso2-application-security:
          security-types:
            - auth2
            - optional: false
      put:
        operationId: UpdatePO
        parameters:
          - name: uid
            in: path
            optional: true
  
```

59) You can also add additional documentation:

The screenshot shows the 'Documents > Add New Document' dialog. The left sidebar is identical to the previous screenshot. The main dialog has fields for 'Name' (with validation 'Document name cannot be empty'), 'Summary' (with placeholder 'Provide a brief description for the document'), and 'Type' (radio buttons for How To, Sample & SDK, Public Forum, Support Forum, or Other). Below these is a 'Source' section with radio buttons for Inline, Markdown, URL, or File. A large text area contains the instruction 'Please save the document. The content can be edited in the next step.' At the bottom are 'Add Document' and 'Cancel' buttons.



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.

