



Labkit
for
WSO2 API Manager 2.0.0
Developer Fundamentals

Email: training@wso2.com

Table of Contents

[Lab: Getting Started with WSO2 API Manager](#)
[Lab: Defining Users and Roles](#)
[Lab: Creating and Publishing the PizzaShack API](#)
[Lab: Working with Tenants](#)
[Lab: Subscribing to APIs](#)
[Lab: Invoking the API](#)
[Lab: Add Throttling Policy](#)
[Lab: Analyze Runtime Statistics](#)
[Lab: Managing Alerts with Real-Time Analytics](#)
[Lab: Analyzing Logs with the Log Analyzer](#)
[Lab: Using Published APIs](#)

Lab: Getting Started with WSO2 API Manager

Training Objective

Verify that the products required for running this tutorial are installed and configured. and deploy and test the data required to work with the sample.

Note: The participants are expected to be connected to the internet throughout in order to successfully complete the lab exercises.

Business Scenario

PizzaShack Limited wants to extend their website for placing and managing online orders as a part of their effort in becoming the #1 online pizza shop. They have also found it increasingly useful to build an application for smartphones. The application is a Web application allowing you to choose and buy a Pizza online. They have subcontracted the development of the smartphone application to FunkyApps LLC. John Doe, Chief Architect of FunkyApps had some interesting feedback for PizzaShack. He suggested that the company considers monitoring of consumer statistics and probably looking into complex event processing in the future. John, also suggested that they make use of an API Store backed by a modern API Gateway providing security features such as OAuth 2.0 access tokens.

In order to achieve this, PizzaShack will be implementing WSO2 API Manager and a number of other WSO2 products for monitoring statistics, single sign on and so on.

The application leverages an API with 3 resources, which are exposed via the API Manager. Corresponding services are hosted in the WSO2 Application Server. Data Analytics Server will be used for monitoring.

High Level Steps

- Install WSO2 API Manager
- Install WSO2 API Manager Analytics
- Install WSO2 Application Server
- Other installations
- Overview of the key directories in WSO2 API Manager
- Key configuration files
- Configure port offsets
- Deploy and test JAX-RS service

Detailed Instructions

Install WSO2 API Manager

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentation [1] for detailed instructions. If prerequisites are fulfilled, instructions on installing the product can be found for:

- Linux or OS X at [2]
- Windows [3]

[1] <https://docs.wso2.com/display/AM200/Installation+Prerequisites>

[2] <https://docs.wso2.com/display/AM200/Installing+on+Linux+or+OS+X>

[3] <https://docs.wso2.com/display/AM200/Installing+on+Windows>

Install WSO2 API Manager Analytics

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentation [1] for detailed instructions.

[1] <https://docs.wso2.com/display/AM200/Analytics>

Install WSO2 Application Server

Before installing the product, ensure that the installation prerequisites have been met. Refer to the documentation [1] for detailed instructions. If prerequisites are fulfilled, instructions on installing the product can be found for:

- Linux or OS X at [2]
- Windows [3]

[1] <https://docs.wso2.com/display/AS530/Installation+Prerequisites>

[2] <https://docs.wso2.com/display/AS530/Installing+on+Linux+or+OS+X>

[3] <https://docs.wso2.com/display/AS530/Installing+on+Windows>

Other Installations

In order to complete the use case described in this labkit the following products must be installed:

Apache Ant [1], Apache Maven [2], MYSQL[3], JDBC driver for MySQL[4], CUrl[6], Google Chrome [7], Advanced Rest Client [8],

[1] <http://ant.apache.org/manual/install.html>

[2] <https://maven.apache.org/install.html>

[4] <https://www.mysql.com/downloads/>

[5] <http://dev.mysql.com/downloads/connector/j/>

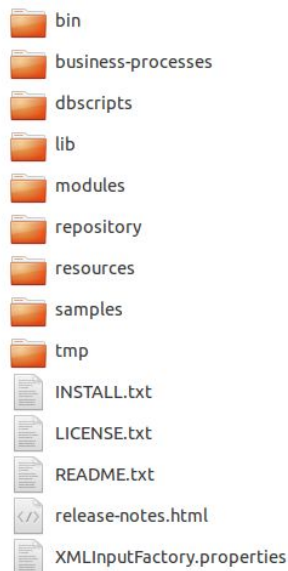
[6] <http://curl.haxx.se/download.html>

[7] <https://support.google.com/chrome/answer/95346?hl=en>

[8]<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddfdnphfgcellkdfbfbjeloo>

Overview of the Key Directories in WSO2 API Manager

The structure of the <APIM_HOME> folder is as follows.



bin - This folder contains all the executable files including those scripts that are used to start/stop the application on Linux and Windows environments e.g., wso2server.sh and wso2server.bat.

business-processes – This folder contains information related to business process execution for APIM Management related operations. With WSO2 API Manager we have added 4 workflow plug points for the below operations: user creation process, application creation process, application registration process, subscription process.

dbscripts – A collection of database scripts required to create the Carbon database and the API Management specific database on a variety of database management systems.

lib –The lib directory houses all the jar files that will be converted to OSGi bundles at startup and copied to the dropins directory.

modules – All the host objects belonging to the Jaggery module are declared within the modules folder in a file called module.xml.

repository – The main repository for all kind of deployments and configurations in Carbon. This includes all default services, created APIs, Carbon configurations etc.

resources – Contains additional resources that may be used by the APIM.

samples - Sample APIs that can be used to explore the WSO2 API Manager functionality.

tmp – Will contain temporary files that are created when a product is run. These files will be cleared from time to time based on housekeeping tasks.

Key Configuration Files

File	Description
<PRODUCT_HOME>/repository/conf/carbon.xml	The carbon server configuration file
<PRODUCT_HOME>/repository/conf/api-manager.xml	The main configuration file that governs the APIM specific functionality.
<PRODUCT_HOME>/repository/conf/datasources/master_datasources.xml	The main configuration file for carbon datasources. Registry and User Manager refer the datasource configuration defined in this file.
<PRODUCT_HOME>/repository/conf/identity/identity.xml	The configuration file that governs identity related functionality.
<PRODUCT_HOME>/repository/conf/log4j.properties	The log4j configuration file used by WSO2 Carbon.
<PRODUCT_HOME>/repository/conf/registry.xml	The carbon registry configuration file. This will be used when the WSO2 Embedded Registry is used.
<PRODUCT_HOME>/repository/conf/user-mgt.xml	The User Manager configuration file used for configuring user management details.
<PRODUCT_HOME>/repository/conf/security/secret-conf.properties	The secret manager configuration that is used by the secret vault component.

Configure Port Offset

When you run multiple WSO2 products, multiple instances of the same product, or multiple WSO2 product clusters on the same server or virtual machines (VMs), you must change their default ports with an offset value to avoid port conflicts. The default HTTP and HTTPS ports (without offset) of a WSO2 product are 9763 and 9443 respectively. Port offset defines

the number by which all ports defined in the runtime such as the HTTP/S ports will be changed. For example, if the default HTTP port is 9763 and the port offset is 1, the effective HTTP port will change to 9764. For each additional WSO2 product instance, you set the port offset to a unique value. The default port offset is 0.

There are two ways to set an offset to a port:

- Pass the port offset to the server during startup. The following command starts the server with the default port incremented by 3: `/wso2server.sh -DportOffset=3`
- Set the Ports section of `<PRODUCT_HOME>/repository/conf/carbon.xml` as follows: `<Offset>3</Offset>`

We will be using APIM, APIMAnalytics, BPS and AS for these exercises. Since all these servers need to run in the same machine for this demo, we must change the port offset in `home/repository/conf/carbon.xml` file. Enter the following port offsets for each product:

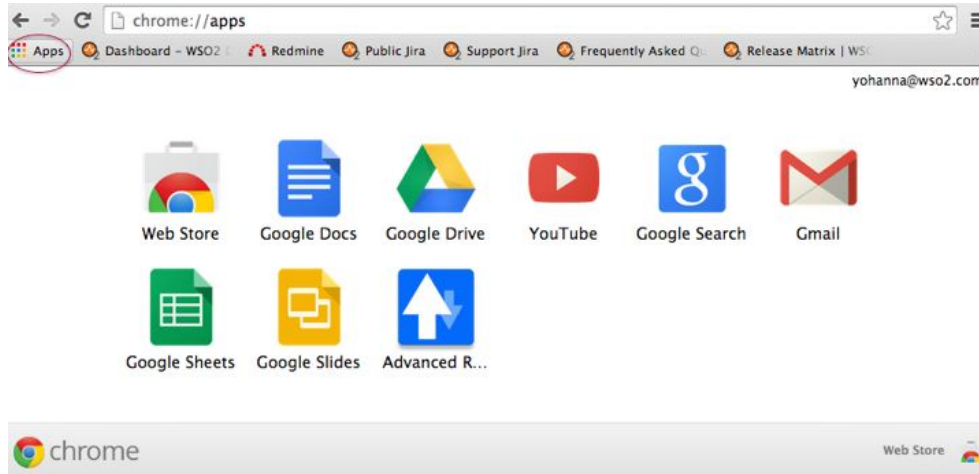
File	Port Offset
<code><APIM_HOME>/repository/conf/carbon.xml</code>	0
<code><Analytics_HOME>/repository/conf/carbon.xml</code>	1
<code><AS_HOME>/repository/conf/carbon.xml</code>	3

Deploy and Test JAX-RS Service

1. Open a Command Line Interface.
2. Start the WSO2 Application Server by navigating to the `<AS_HOME>/bin` directory and running `wso2server.bat` (on Windows) or `sh wso2server.sh` (on Linux)
3. Unzip the PizzaShack.zip file that you will find at <https://github.com/wso2/WSO2-Training/releases/download/APIM2.0.0DF/PizzaShack.zip> to a folder called SAMPLE.
4. Open a Command Line Interface.
5. Navigate to the `SAMPLE/PizzaShack/pizza-shack-api` folder and run `mvn clean install`. - this will build the JAX-RS service.
6. Copy the war file from the target directory to `<AS_HOME>/repository/deployment/server/webapps/` so that it is deployed.
7. The service will be deployed after a few seconds.

Once you have deployed the service, you can test it using the Chrome Advanced REST Client, which is a very useful extension to Chrome.

1. Start Chrome from the toolbar and open a new tab.
2. Click on **Apps** and then on **Advanced Rest Client**.



1. Enter <http://localhost:9766/pizzashack-api-1.0.0/api/menu> as the URL. (9766 is used because the default port is 9763 and a port offset of 3 was defined for the AS server)
2. Set the verb as **GET**.
3. Click **Send**.

You should see an answer similar to the one below:


```

-0: {
  name: "BBQ Chicken Bacon"
  icon: "/images/6.png"
  description: "Grilled white chicken, hickory-smoked bacon and fresh sliced onions in barbeque sauce"
  price: "22.99"
}
-1: {
  name: "Chicken Parmesan"
  icon: "/images/1.png"
  description: "Grilled chicken, fresh tomatoes, feta and mozzarella cheese"
  price: "19.99"
}
-2: {
  name: "Chilly Chicken Cordon Bleu"
  icon: "/images/10.png"
  description: "Spinash Alfredo sauce topped with grilled chicken, ham, onions and mozzarella"
  price: "19.99"
}
-3: {
  name: "Double Bacon 6Cheese"
  icon: "/images/9.png"
  description: "Hickory-smoked bacon, Julienne cut Canadian bacon, Parmesan, mozzarella, Romano, Asiago and and Fontina cheese"
  price: "24.99"
}
-4: {
  name: "Garden Fresh"
  icon: "/images/3.png"
  description: "Slices onions and green peppers, gourmet mushrooms, black olives and ripe Roma tomatoes"
  price: "11.99"
}
. . .

```

Expected Outcome

As APIM was not given an offset, it will run on the default port while the other products will run on the relevant according to the given port offset.

The JAX-RS service is deployed in the WSO2 Application Server.

[1] <https://docs.wso2.com/display/AM200/Running+the+Product>

Lab: Defining Users and Roles

Training Objective

In this section, you will learn how to set up custom roles and users. Roles contain permissions for users to manage the server. You can create different roles with various combinations of permissions and assign them to a user or a group of users. User roles can be reused throughout the system and prevent the overhead of granting multiple permissions to each and every user individually.

Business Scenario

PizzaShack has an employee who will be creating the menu, order and delivery APIs and another employee who will be publishing this to the website. API consumers can log in to the site and access these APIs. Separate user roles and users should be created for API creators and publishers.

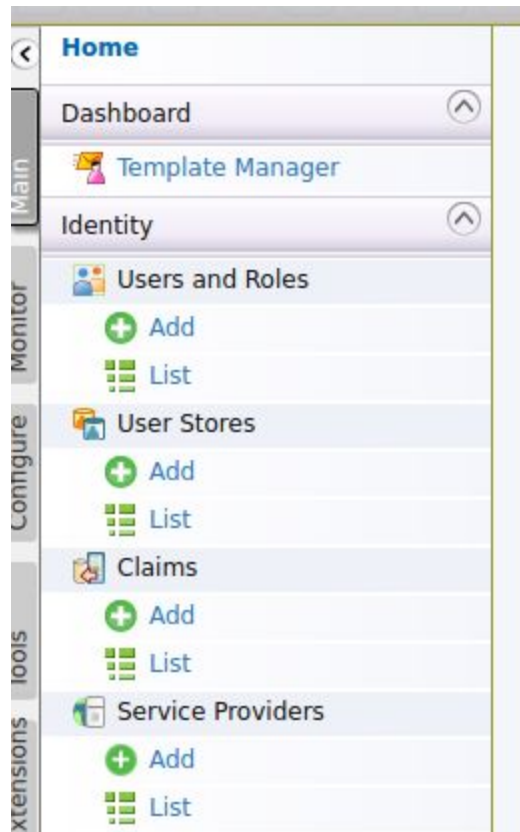
High Level Steps

- Define roles
- Define users via the admin console

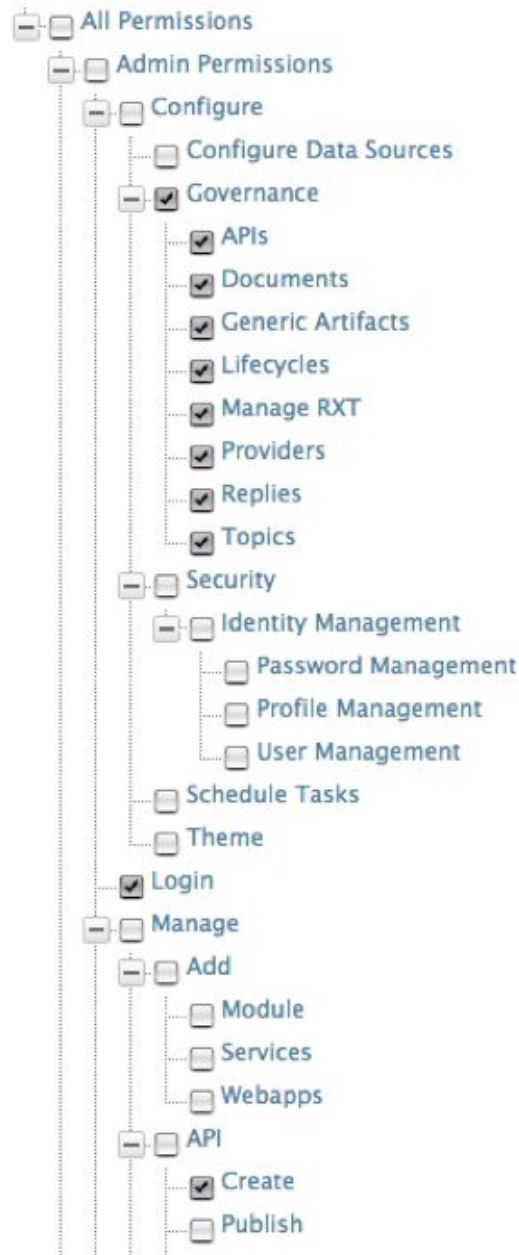
Detailed Instructions

Define Roles

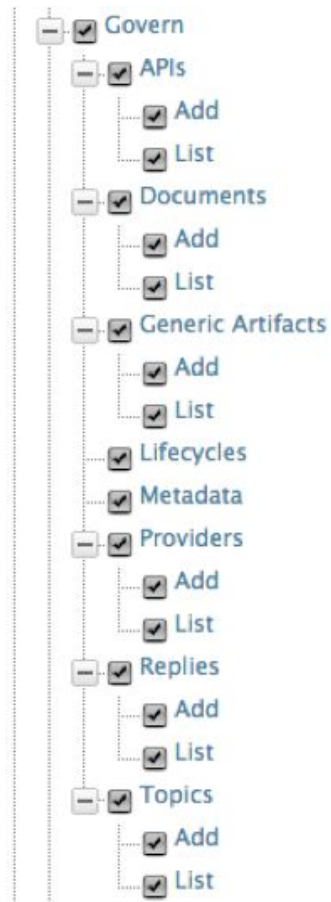
1. Open a Command Line Interface.
2. Start the WSO2 API Manager by navigating to the <APIM_HOME>/bin directory and running wso2server.bat (on Windows) or sh wso2server.sh (on Linux)
3. Log in to the APIM admin console, which is available by default at:
<https://hostname:9443/carbon>. You can log in to the console using the default admin/admin credentials.



4. Click **Main > Users and Roles > Add**.
5. Provide **creator** as the role name.
6. Click **Next** - You will be presented with a list of permissions. For the creator role, you need to select the following permissions:
 - **Configure > Governance** and all underlying permission
 - **Login**
 - **Manage > API > Create**



- **Manage > Resources > Govern** and all underlying permissions.



10. Click **Finish** (at the bottom of the page).

Repeat the steps to create the **publisher** role, with the following permissions:

- **Login**
- **Manage > API > Publish**

Define Users via the Admin Console

You can now create a user in each of those roles. To do so:

1. Click **Main > Users and Roles > Add**.
2. Click **Users**.
3. Click **Add New User**.
4. Provide user name (apicreator) and password.
5. Click **Next**.
6. Select the creator role.
7. Click **Finish**.

Repeat the steps to create a user (apipublisher) in the publisher role.

Expected Outcome

A creator role with permissions for creating APIs and a publisher role with permissions for publishing APIs have been created. Users named apicreator and apipublisher have been created and given the appropriate roles.

Lab: Creating and Publishing the PizzaShack API

Training Objective

Learn how to create an API, add documentation to it and publish it to the store using the Publisher.

Business Scenario

After setting up the APIM, the API is created and published through the API Publisher in order to make it subscribable from the store.

Business Scenario: PizzaShack Limited is providing a store from which consumers can subscribe to their API. This works as a secondary business function for PizzaShack and attracts many developers to the PizzaShack website. The API will be comprehensively documented for ease of use.

High Level Steps

- Add the PizzaShack API to the store
- Implement APIs
- Manage APIs
- Add documentation
- Publish the APIs

Detailed Instructions

Add the PizzaShack API to the Store

Now that we set up the APIM and added users, we are ready to publish the API the Pizza Shack application requires.

To add the API to the store, follow those steps:

1. Open the API Publisher web application from <https://localhost:9443/publisher>.
2. Log in using the user in creator role you defined previously (apicreator).
3. Click **Add New API**.
4. Select **Design New API**.
5. Click **Start Creating**.
6. Provide information on the API as per the table below

Field	Value	Description
Name	PizzaAPI	Name of API as you want it to appear in the API store
Context	pizzashack	URI context path that is used by API consumers (Application Developers)
Version	1.0.0	API version (in the form of version.major.minor)
Visibility	Public	Whether this API is visible to all or restricted to certain roles.
Thumbnail Image	Download a PizzaShack logo image and upload it	Icon to be displayed in API store (can be jpeg, tiff, png format) - Under Advanced Options .
Description	Pizza API: Allows to manage pizza orders (create, update, retrieve orders)	High level description of API functionality
Tags	pizza, order, pizza-menu	One of more tags. Tags are used to group/search for APIs
API Definition		<p>An API is made up of one or more resources. Each resource handles a particular type of requests. A resource is analogous to a method (function) in a larger API.</p> <p>API resources can accept following optional attributes:</p> <ul style="list-style-type: none"> • verbs: Specifies the HTTP verbs a particular resource would accept. Allowed values are GET, POST, PUT, DELETE. Multiple values can be specified. • uri-template: A URI template as defined in http://tools.ietf.org/html/rfc6570 (eg: /phoneverify/{phoneNumber}) • url-mapping: A URL mapping as defined as per the servlet specification (extension mappings, path mappings and exact mappings)

For the PizzaShack API, we will be defining 4 resources as defined below.

Resource URL	Methods
menu	GET
order	POST
order/{orderid}	GET
delivery	GET

- Enter the resource URL and click **Add** and repeat for each resource URL.



URL Pattern: /pizzashack/1.0.0 Uri Pattern E.g.: path/to/resource

☐ GET ☐ POST ☐ PUT ☐ DELETE ☐ PATCH ☐ HEAD [more](#)

[Add](#)

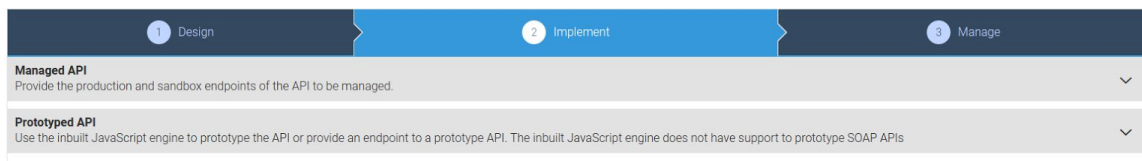
POST	/order	+ Summary	B
GET	/menu	+ Summary	B
PUT	/order/{orderid}	+ Summary	B
GET	/order/{orderid}	+ Summary	B

[Save](#) [Next: Implement >](#)

- Click **Implement**.

Implement APIs

- Select **Managed API**.



1 Design 2 **Implement** 3 Manage

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

Prototyped API
Use the inbuilt JavaScript engine to prototype the API or provide an endpoint to a prototype API. The inbuilt JavaScript engine does not have support to prototype SOAP APIs

- Specify the following.

Field	Value	Description
Endpoint Type	HTTP Endpoint	
Production Endpoint	http://localhost:9766/pizzashack-api-1.0.0/api/ (9766 is used because the default port is 9763 and a port offset of 3 was defined for the AS server)	Endpoint of the backend service URL

Sandbox URL	N/A	Endpoint of sandbox (testing) back end service. A sandbox URL is meant to be used for online testing of an API with easy access to an API key. We have no sandbox in this sample.
Endpoint Security Scheme:*	Non Secured	
Enable Message Mediation	Not Selected	Define your own message mediation policy for incoming and outgoing messages.
Enable API based CORS Configuration	Not Selected	Enable CORS for the API

Manage APIs

Managing an API involves specifying its management attributes such as throttling tiers, external sequences, and so on. Provide the following information on the **Manage** tab of the API.

Field	Value	Description
Make this default version	Not selected	The default version option allows you to mark one API, from a group of API versions, as the default one, so that it can

		be invoked without specifying the version number in the URL.
Transports	HTTP/HTTPS	APIs can be exposed in HTTP and/or HTTPS transport: The transport protocol on which the API is exposed. Both HTTP and HTTPS transports are selected by default. If you want to limit API availability to only one transport (e.g., HTTPS), un-check the other transport.
Response Caching	Disabled	Response caching is used to enable caching of response messages per API. Caching protects the backend systems from being exhausted due to serving the same response (for same request) multiple times. If you select the enable option, specify the cache timeout value (in seconds) within which the system tries to retrieve responses from the cache without going to the backend.
Maximum Backend Throughput	Unlimited	Limits the total number of calls the API Manager is allowed to make to the backend. While the other throttling levels define the quota the API invoker gets, they do not ensure that the backend is protected from overuse. Hard throttling limits the quota the backend can handle.
Subscription Tiers	Unlimited	The API can be available at different level of service; you can select multiple entries from the list. At subscription time, the consumer chooses which tier they are interested in.
Advanced Throttling Policies	Apply per Resource	Throttling policies can be applied per resource (different policies for each resource) or one policy for all resources

For this use case, we will also be making use of OAuth2.0 scopes. Therefore we will be creating a scope named `order_pizza` and allowing that scope to only users with `webuser` role.

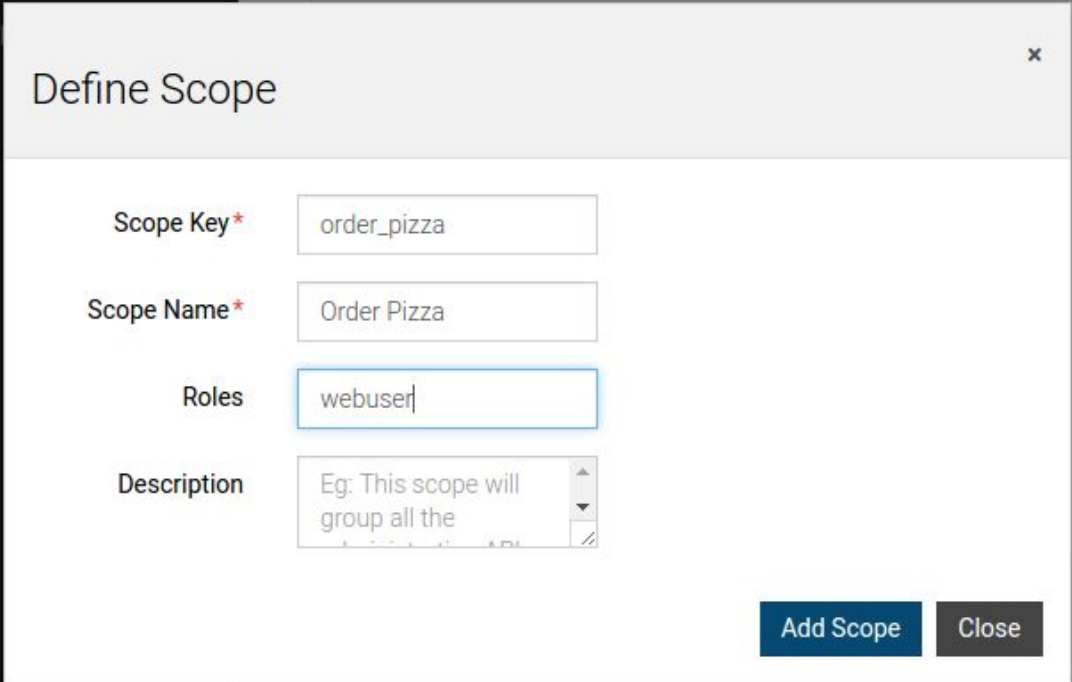
Scopes enable fine-grained access control to API resources based on user roles. You define scopes to an API's resources. When a user invokes the API, his/her OAuth 2 bearer token cannot grant access to any API resource beyond its associated scopes.

Field	Description
Scope Key	A unique key for identifying the scope. Typically, it is prefixed by part of the API's name for uniqueness, but is not necessarily reader-friendly.
Scope Name	A human-readable name for the scope. It typically says what the scope does.
Roles	The user role(s) that are allowed to obtain a token against this scope. E.g., manager, employee.

To invoke an API protected by scopes, you need to get an access token via the Token API. Tokens generated from the **APPLICATIONS** page in the API Store will not work.

Prerequisite - The `webuser` role needs to be created first. See **Test the PizzaShack Application** section to learn how to create this role.

1. Click **Add Scopes**.
2. Enter the following information and click **Add Scope**.



Define Scope

Scope Key*: order_pizza

Scope Name*: Order Pizza

Roles: webuser

Description: Eg. This scope will group all the

Add Scope **Close**

Business Information

- Once the scope is defined, we need to assign that scope to the appropriate resources. Assign it to the /order and /order/{orderid} resources.

Scopes: **order_pizza** : Order Pizza

Roles : webuser

+ Add Scopes

POST	/order	+ Summary	Application & Application User	Unlimited	Order Pizza
GET	/menu	+ Summary	Application & Application User	Unlimited	+ Scope
PUT	/order/{orderid}	+ Summary	Application & Application User	Unlimited	+ Scope
GET	/order/{orderid}	+ Summary	Application & Application User	Unlimited	Order Pizza

Save **Save & Publish** **Cancel**

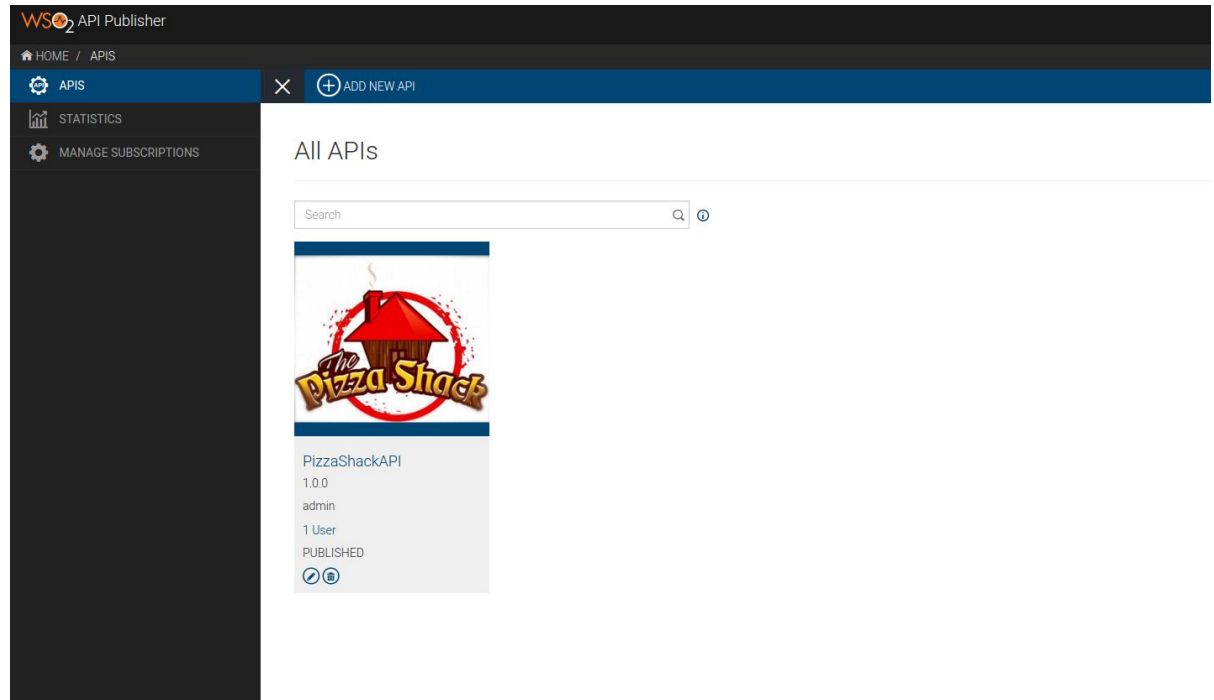
Note: The order in which the scopes are shown in the image above can differ from the order on screen. Make sure you add the scopes on the POST/order and GET/order{orderid}.

Once a request has been accepted by a resource, it will be mediated through an in-sequence. Any response from the back-end is handled through the out-sequence. A fault sequence is used to mediate any unhandled errors that might occur in either the in or out sequence. Default in-sequence, out-sequence and fault sequence are generated when the API is published.

4. Click **Save**.

Add Documentation


1. Once the API has been created, click **Browse** and then click the PizzaShack icon and open its details.



You see something similar to the image below:

PizzaShackAPI - 1.0.0

Overview
Lifecycle
Versions
Docs
Users



0 Users

PUBLISHED

Docs

[View in Store](#)

Description

This document describe a RESTful API for Pizza Shack online pizza delivery store.

Visibility	Public
Context	/pizzashack/1.0.0
Production URL	https://localhost:9443/am/sample/pizzashack/v1/api/
Sandbox URL	https://localhost:9443/am/sample/pizzashack/v1/api/
Date Last Updated	8/4/2016, 1:51:09 PM
Tier Availability	Unlimited
Default API Version	None
Tags	order,pizza-menu,pizza
Business Owner	Jane Roe [marketing@pizzashack.com]
Technical Owner	John Doe [architecture@pizzashack.com]
Published Environments	Production and Sandbox

- Click the **Docs** tab and add documentation to the API. Documentation can be provided inline or via a URL or file. For inline documentation, you can edit the contents directly from the API publisher interface.

Several documents types are available:

- How To
- Samples and SDK
- Public Forum
- Support Forum
- Other

To create a How-To document:

- Select the **How To** type.
- Provide a name for the document, such as “How to use this API”.
- Provide a short description of the document (this will appear in the API store).
- Choose the **Inline** option under **Source**.
- Click **Add Document**.

Once the document has been added, you can edit the contents by clicking on the **Edit Content** link. An embedded editor allows you to edit the document contents.

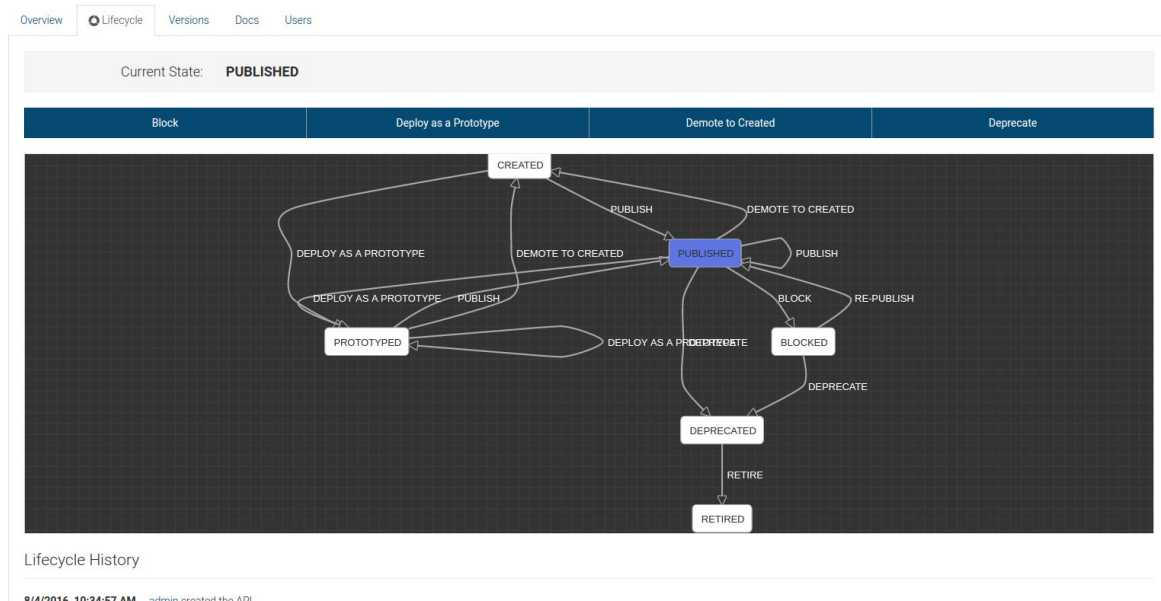
Publish the API

The API is now ready to be published. This has to be done by a user with the publisher role.

To publish the API:

1. Log out as apicreator and log in as apipublisher.
2. Click on the API - You can see that an additional tab is now available, allowing you to manage the API lifecycle.
3. To publish the API, select **PUBLISH**.

The API is now published and visible to consumers in the API store. The API life cycle history is visible at the bottom of the page.



Expected Outcome

The PizzaShack API which manages pizza orders has been created and published and can be accessed through the Store.

Lab: Working with Tenants

Training Objective

Learn how to create tenants and use tenants to share APIs.

Business Scenario

PizzaShack Limited would like to create a separate tenant for employees in order to share APIs only with them. The first API that will be shared will be used for capturing statistics on customers. This API will be shared by the PizzaShack head office and shared among the branches.

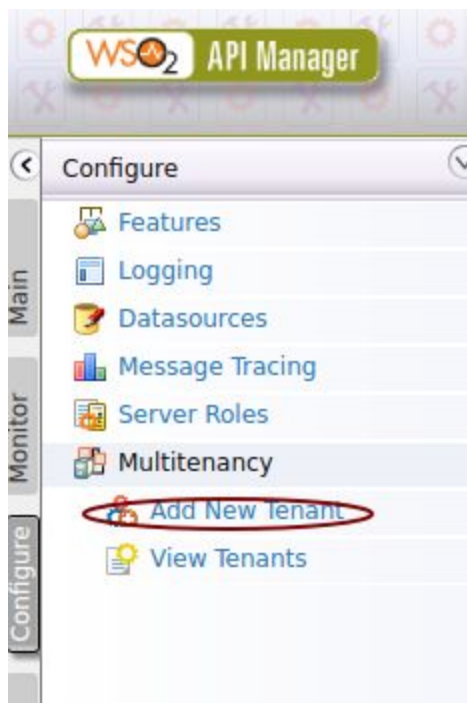
High Level Steps

- Create tenants
- Share API within tenant

Detailed Instructions

Create Tenants

1. Log in to the Management Console as an admin user.
2. Click **Configure** > **Multitenancy** > **Add New Tenant**.



3. Register a new domain named pizzashack.com as follows:

Home > Configure > Multitenancy > Add New Tenant

Help

Register A New Organization

Domain Information

Domain * pizzashack.com

Use a domain for your organization, in the format "example.com". This domain should be unique.

Usage Plan Information

Select Usage Plan For Tenant* Demo

According to the selected plan, resources will be allocated to you. You can update or downgrade your plan later according to your requirements.

Tenant Admin

First Name* James

Last Name* Brown

Admin Username * admin @pizzashack.com

Admin Password *

Admin Password (Repeat) *

Contact Details

Email* james@pizzashack.com

Save

Enter the Tenant Domain: Find

Help


Tenants List

Domain	Email	Created Date	Active	Edit
pizzashack.com	WSO2 Carbon	5 16:13:13	<input checked="" type="checkbox"/>	Edit

You have registered the Organization Successfully

OK


- Go to the URL for the store <https://localhost:9443/store/>. The super tenant and the newly created tenant will be displayed.



API STORES
AVAILABLE ON THIS
SERVER

carbon.super	Visit Store
pizzashack.com	Visit Store

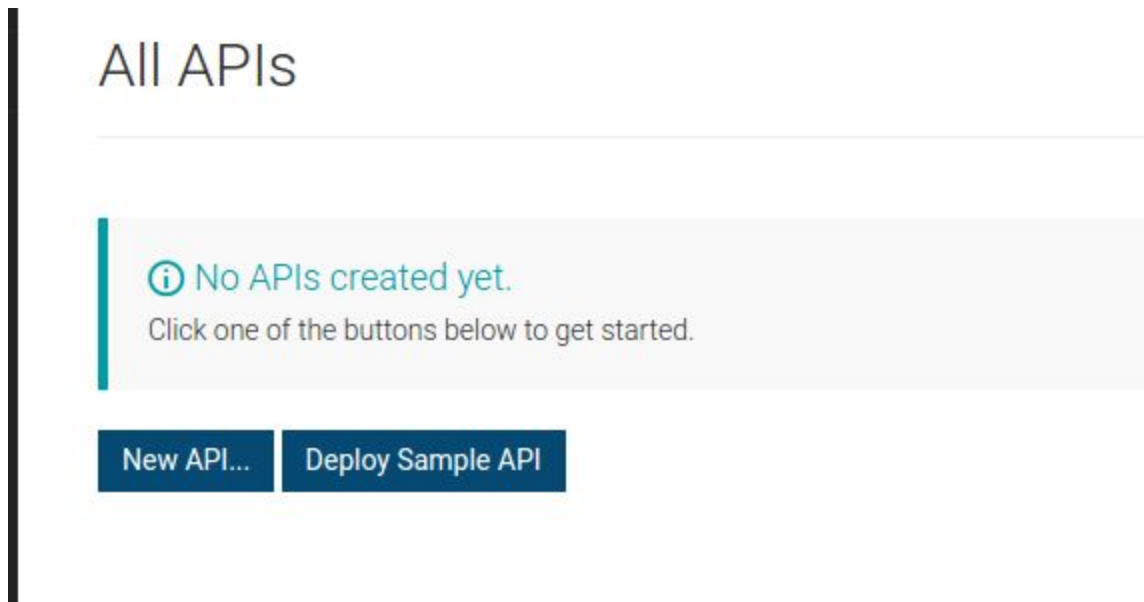
- Click on pizzashack.com and note that no APIs have been published yet.
- Log in as admin@pizzashack.com. There are no APIs displayed even after logging in.



Sign in to your account

Share API within Tenant

1. Log in to the Publisher as admin@pizzashack.com.
2. Create a new API.



All APIs

No APIs created yet.

Click one of the buttons below to get started.

3. Click **Design new API** and then click **Start Creating**.

Let's get started!

Add New API

<input type="radio"/>	I have an Existing API Use an existing API's endpoint or the API definition to create an API.	▼
<input type="radio"/>	I have a SOAP Endpoint Use an existing SOAP endpoint to create a managed API. Import WSDL of the SOAP service.	▼
<input checked="" type="radio"/>	Design New API Design and prototype a new API.	^

4. Enter the following details.

1 Design 2 Implement

General Details

Name: * ? CustomerInfo

Context: * ? customer

Version: * 1.0.0

Visibility: ? Visible to my domain

Description:
Maximum 20000 characters.

Tags: ? Add tags
Type a Tag and Enter

5. Add the following resources.

API Definition

URL Pattern: customer/1.0.0 Url Pattern E.g.: path/to/resource Import Edit Source

☐ GET ☐ POST ☐ PUT ☐ DELETE ☐ PATCH ☐ HEAD more

+ Add

POST	/CreateCustomer	+ Summary	ⓘ
GET	/QueryCustomerInfo	+ Summary	ⓘ
PUT	/UpdateCustomerInfo	+ Summary	ⓘ
DELETE	/DeleteOrderInfo	+ Summary	ⓘ

Save Next: Implement >

6. For the CreateCustomer resource, change **Required** to **True** for the **Payload** parameter.

POST /CreateCustomer [+ Summary](#) ⓘ

Description: [+ Add Implementation Notes](#)

Produces: [Empty](#) Consumes: [Empty](#)

Parameters:

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
Payload	Request Body	body	+ Empty	True	ⓘ

6. Add the following parameter for the QueryCustomerInfo resource:

GET /QueryCustomerInfo [+ Summary](#)

Description : [+ Add Implementation Notes](#)

Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
MobileNumber	The caller's phone number	query	string	True	Delete

7. For the **UpdateCustomerInfo** resource, change **Required** to **True** for the **Payload** parameter.

PUT /UpdateCustomerInfo [+ Summary](#)

Description : [+ Add Implementation Notes](#)

Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
Payload	Request Body	body	+ Empty	True	Delete

8. Add the following parameter to the DeleteOrderInfo resource.

DELETE /DeleteOrderInfo [+ Summary](#)

Description : [+ Add Implementation Notes](#)

Produces : [Empty](#) Consumes : [Empty](#)

Parameters :

Parameter Name	Description	Parameter Type	Data Type	Required	Delete
orderid	The order ID that should be deleted	query	string	True	Delete

9. Click **Implement**.
10. Click **Prototyped API**.

CustomerInfo: /t/pizzashack.com/customer/1.0.0

1 Design 2 **Implement** 3 Manage

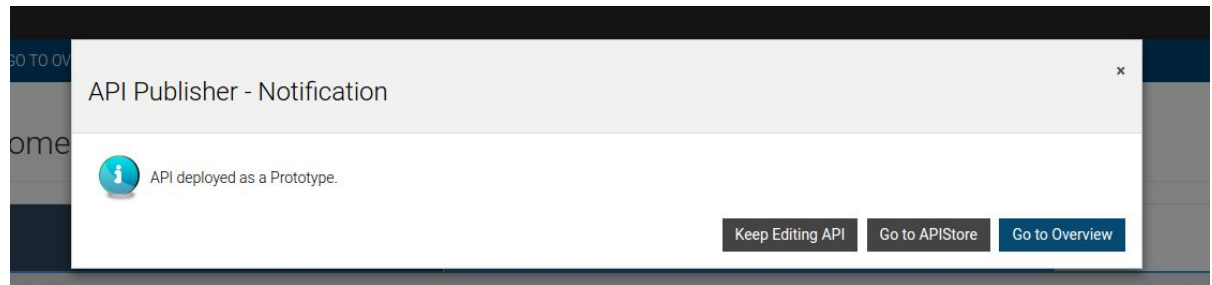
Managed API
Provide the production and sandbox endpoints of the API to be managed.

Prototyped API
Use the inbuilt JavaScript engine to prototype the API or provide an endpoint to a prototype API. The inbuilt JavaScript engine does not have support to prototype SOAP APIs

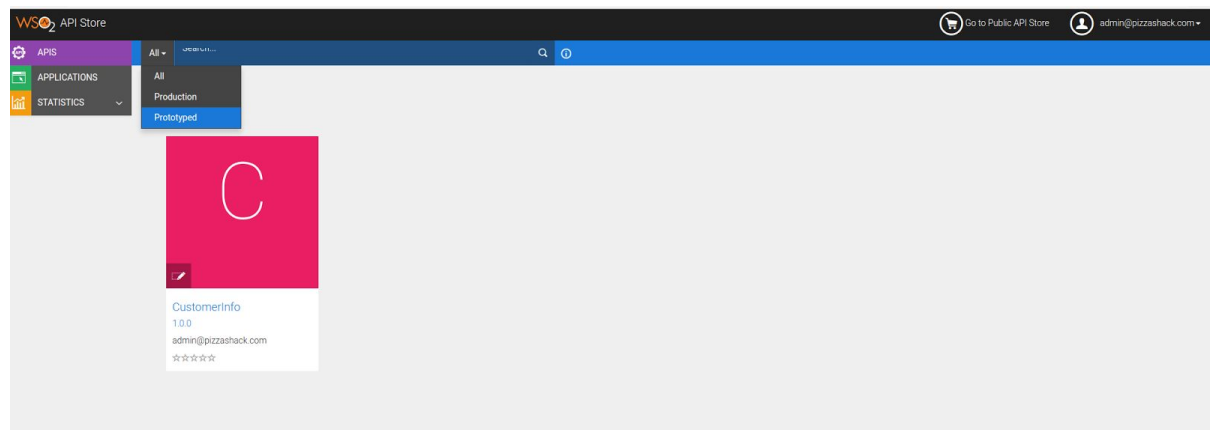
11. Select **Inline** as the implementation method.



12. Click **Deploy as Prototype**.



13. Click **Go to Overview** and to see the overview of the newly created API.
14. Go to the API Store <https://localhost:9443/store/>.
15. Go to the **carbon.super** store and note that the newly created API does not appear under **APIs** or **Prototyped APIs**.
16. Try logging in as admin and note that the API is still not visible.
17. Log in to the **pizzashack.com** store as `admin@pizzashack.com` and view API under Prototyped APIs.



Expected Outcome

A tenant is created to contain the employees of PizzaShack. An API is created to capture statistical data of customers and this is shared with only this tenant.

Lab: Subscribing to APIs

Training Objective

Learn how to subscribe to the APIs using the store.

Business Scenario

After PizzaShack successfully publishes the APIs other partners who would like to use the PizzaShack APIs as a base can open the API store and check its contents and subscribe to the API if interested.

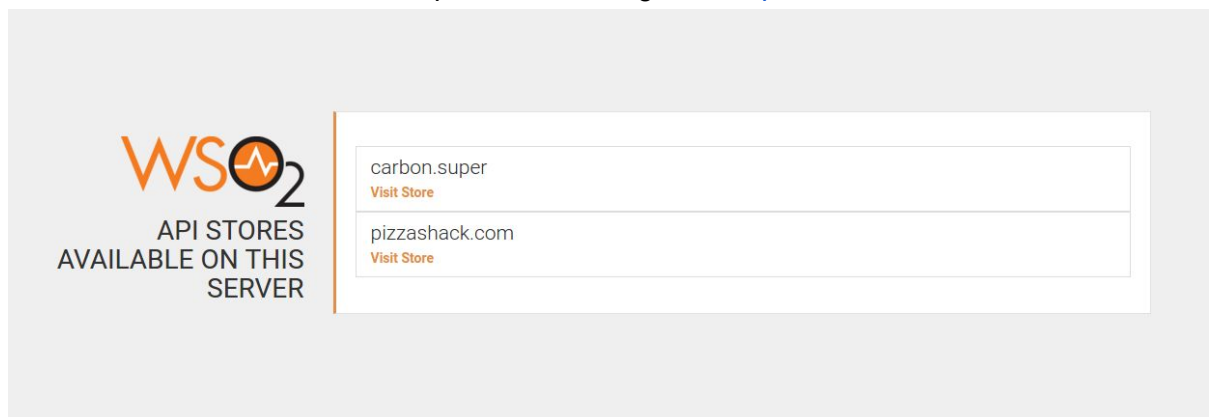
High Level Steps

- Browse the store
- Define users via self-registration
- Subscribe to APIs

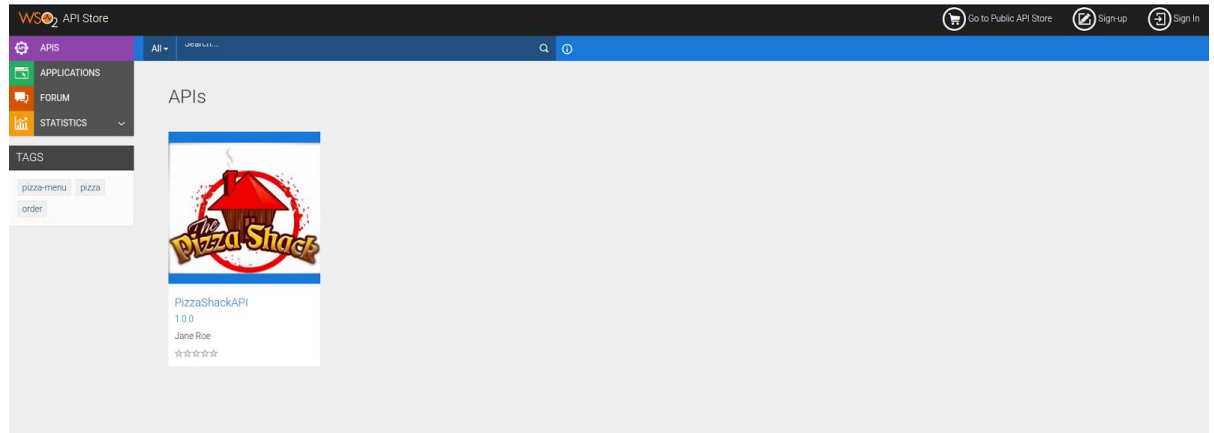
Detailed Instructions

Browse the Store

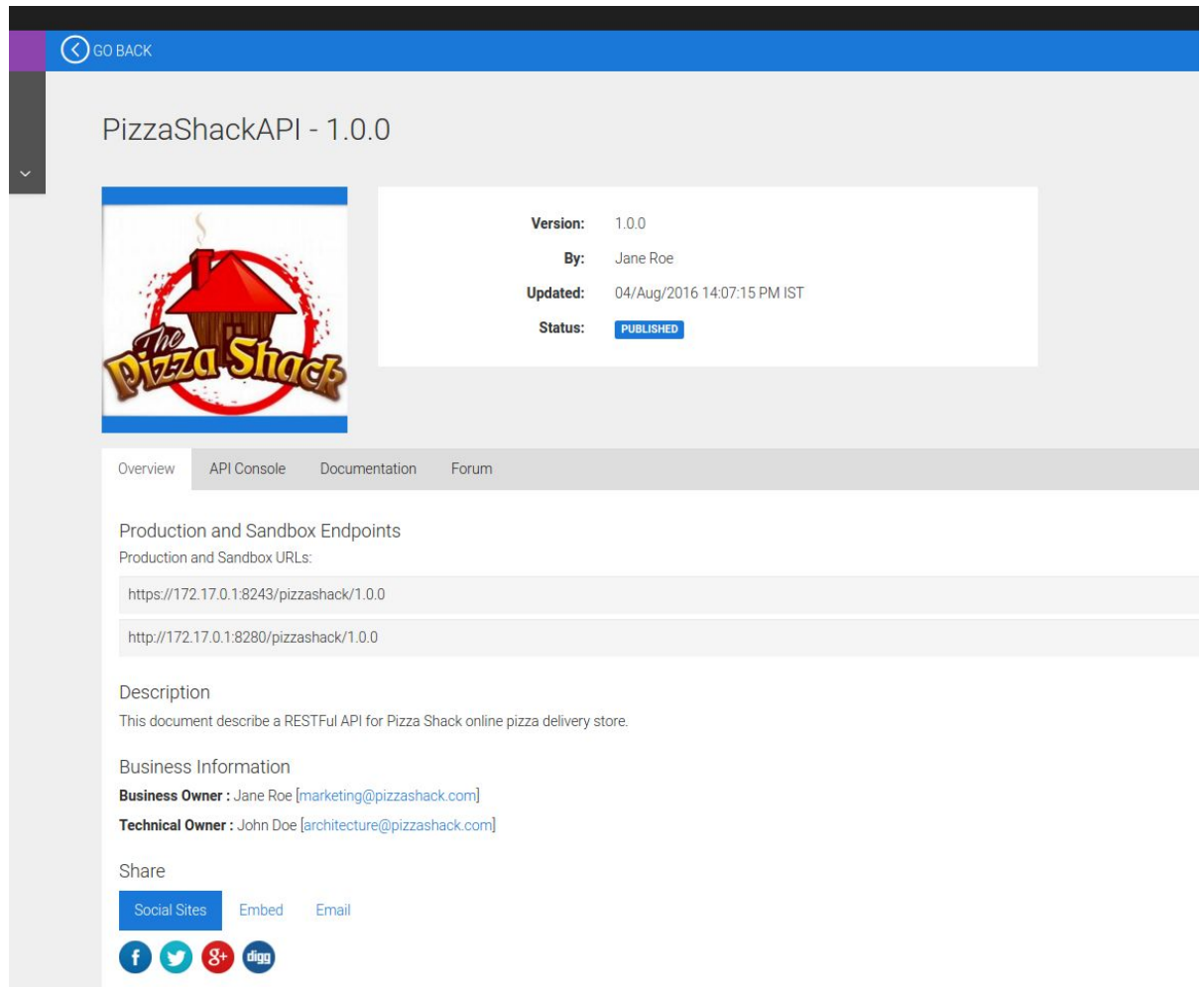
1. To view the API store contents, open the following URL: <https://localhost:9443/store>.



2. Click carbon.super.
3. Log off if you have already logged in.



3. Click the icon to see the details entered by the API creator:



You can browse the API store and check the documentation without the necessity to provide credentials.

You can search API by their name, context, version or by clicking on the tags to the **left**.

You can also test the API from the API Console, but prior to that, you need to subscribe to the APIs to obtain a security token.

Define Users via Self-Registration

When a user connects to the API store for the first time, they can self-register.

1. While within the carbon.super tenant, click **Sign-Up** at the top right of the window.
2. Fill in the fields as required and click **Submit**.

The **subscriber** role is already defined out of the box, as it is used in the self-registration process.

Subscribe to an API

As a consumer, you can subscribe to an API by following those steps:

1. Log in to the store using the user created in the above exercise and access the **carbon.super tenant**. You can now see additional information for the API, and set ratings and provide comments.
2. Go to **APPLICATIONS** and create a PizzaShack application. Select **Unlimited** in the **Per Token Quota** field.

APPLICATION LIST

Add Application

An application is a logical collection of APIs. Applications allow you to use a single access token to invoke a collection of APIs and to subscribe to one API by default.


Name*
 Characters left: 60

Per Token Quota Allows unlimited requests
 This feature allows you to assign an API request quota per access token. Allocated quota will be shared among all the subscribed APIs of the application.

Description

3. Select the Pizza API, subscribe to the API using the PizzaShack Application - Select the **Unlimited** tiers level (we need to do several calls in a limited time from the Pizza web application).

PizzaShackAPI - 1.0.0



Version: 1.0.0
 By: Jane Roe
 Updated: 04/Aug/2016 14:07:15 PM IST
 Status: **PUBLISHED**
 Rating: ☆☆☆☆

Applications

Tiers

Overview API Console Documentation Forum

Production and Sandbox Endpoints
 Production and Sandbox URLs:

<https://172.17.0.1:8243/pizzashack/1.0.0>
<http://172.17.0.1:8280/pizzashack/1.0.0>

4. Click **Subscribe**.
5. Switch to the **APPLICATIONS** page. Select PizzaShack Application from the list. You can manage API keys from there
6. Click **Generate Keys**. (Entering a validity time of -1 and regenerating the keys would mean that the validity time of the user access token will be unlimited).

☒ SAML2
 ☒ IWA:NTLM
 ☐ Implicit
 ☒ Refresh Token

☒ Client Credential
 ☐ Code
 ☒ Password

Callback URL

Update

Generating Access Tokens

The following cURL command shows how to generate an access token using the password grant type.

```
curl -k -d "grant_type=password&username=Username&password=Password" \
-H "Authorization: Basic V3FUT0ZNNzkwbTFLR3NSdWdrUDVSeXR4ZmtvYTpyX0tPVEJFVmYwajhEQ0LEN3M00VJjdGZnTmdh" \
https://172.17.0.1:8243/token
```

In a similar manner, you can generate an access token using the client credential grant type with the following cURL command.

```
curl -k -d "grant_type=client_credentials" \
-H "Authorization: Basic V3FUT0ZNNzkwbTFLR3NSdWdrUDVSeXR4ZmtvYTpyX0tPVEJFVmYwajhEQ0LEN3M00VJjdGZnTmdh" \
https://172.17.0.1:8243/token
```

Generate a Test Access Token

Access Token

Above token has a validity period of -1 seconds. And the token has (am_application_scope default) scopes.

Scopes

Validity period

Re-generate

Note: User access tokens have a fixed expiration time, which is set to 60 minutes by default. Before deploying the APIM the default expiration time can be extended by editing the `<AccessTokenDefaultValidityPeriod>` tag in `<PRODUCT_HOME>/repository/conf/identity/identity.xml`.

You have now successfully subscribed to the API and can start using it.

Expected Outcome

As a result of this exercise, a user and application have been created for subscription, the API has been subscribed to, and access tokens have been generated.

Lab: Invoking the API

Training Objective

Learn how to test the API using cURL, build and deploy the web application and test the application.

Business Scenario

After subscribing to the API the partners can access the API through the web application which leverages the WSO2 API Manager token API to generate OAuth2 access tokens on demand.

High Level Steps

- Test the API
- Build and Deploy the PizzaShack Application
- Test the PizzaShack Application

Detailed Instructions

Test the API

To test the API through the API creator, we need to pass the right API key. The API Key must be passed inside an Authorization HTTP Header:

e.g.,

```
Authorization: Bearer vMxNW6ILwNrWvnKJyewejSlHZFka
```

Using cURL, this is very simple - Let's exercise the Menu API.

1. Open a new Command Line Interface.
2. Type

```
curl -v http://localhost:8280/pizzashack/1.0.0/menu
```

You will see the following message:

```
* About to connect() to localhost port 8280 (#0)
* Trying 127.0.0.1... connected
.....
< HTTP/1.1 401 Unauthorized
< WWW-Authenticate: OAuth2 realm="WSO2 API Manager"
....
<ams:fault xmlns:ams="http://wso2.org/apimanager/security">
<ams:code>900902</ams:code>
<ams:message>Missing Credentials</ams:message>
<ams:description>
```

```

    Required OAuth credentials not provided
</ams:description>
</ams:fault>

```

3. Now copy the **Access Token** in the **-APPLICATIONS** page and add it as the **Authorization Bearer**.

```
curl -H "Authorization: Bearer XXXXXXXX" -v http://localhost:8280/pizzashack/1.0.0/menu
```

where XXXXXXXX is the access token generated through the application. You should get a response similar to the one below:

```

* Adding handle: conn: 0x7fac09803a00
* Adding handle: send: 0
* Adding handle: recv: 0
* Curl_addHandleToPipeline: length: 1
* - Conn 0 (0x7fac09803a00) send_pipe: 1, recv_pipe: 0
* About to connect() to localhost port 8280 (#0)
*   Trying ::1...
* Connected to localhost (::1) port 8280 (#0)
> GET /pizzashack/1.0.0/menu HTTP/1.1
> User-Agent: curl/7.30.0
> Host: localhost:8280
> Accept: */*
> Authorization: Bearer WnH0XfYEa0mInyMbnwhM0X24rKoa
>
< HTTP/1.1 200 OK
< Access-Control-Allow-Headers:
authorization,Access-Control-Allow-Origin,Content-Type
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
< Content-Type: application/json
< Date: Mon, 22 Dec 2014 05:41:09 GMT
* Server WSO2-PassThrough-HTTP is not blacklisted
< Server: WSO2-PassThrough-HTTP
< Transfer-Encoding: chunked
<
* Connection #0 to host localhost left intact
[{"name":"BBQ Chicken Bacon","description":"Grilled white chicken,
hickory-smoked bacon and fresh sliced onions in barbeque
sauce","icon":"/images/6.png","price":"14.99"}, {"name":"Chicken
Parmesan","description":"Grilled chicken, fresh tomatoes, feta and
mozzarella
cheese","icon":"/images/1.png","price":"13.99"}, {"name":"Chilly
Chicken Cordon Bleu","description":"Spinash Alfredo sauce topped with
grilled chicken, ham, onions and
mozzarella","icon":"/images/10.png","price":"21.99"}, {"name":"Double

```

```
Bacon 6Cheese","description":"Hickory-smoked bacon, Julienne cut
Canadian bacon, Parmesan, mozzarella, Romano, Asiago and and Fontina
cheese","icon":"/images/9.png","price":"24.99"}, {"name":"Garden
Fresh","description":"Slices onions and green peppers, gourmet
mushrooms, black olives and ripe Roma
tomatoes","icon":"/images/3.png","price":"12.99"}, {"name":"Grilled
Chicken Club","description":"Grilled white chicken, hickory-smoked
bacon and fresh sliced onions topped with Roma
tomatoes","icon":"/images/8.png","price":"12.99"}, {"name":"Hawaiian
BBQ Chicken","description":"Grilled white chicken, hickory-smoked
bacon, barbeque sauce topped with sweet
pine-apple","icon":"/images/7.png","price":"19.99"}, {"name":"Spicy
Italian","description":"Pepperoni and a double portion of spicy
Italian
sausage","icon":"/images/2.png","price":"27.99"}, {"name":"Spinach
Alfredo","description":"Rich and creamy blend of spinach and garlic
Parmesan with Alfredo
sauce","icon":"/images/5.png","price":"17.99"}, {"name":"Tuscan Six
Cheese","description":"Six cheese blend of mozzarella, Parmesan,
Romano, Asiago and Fontina","icon":"/images/4.png","price":"12.99"}]
```

Build and Deploy the PizzaShack Application

This token API needs the Consumer Key and Consumer Secret which were generated previously. You need to provide this information inside the web.xml configuration of the web application.

To edit the file and build the app.

1. Open SAMPLE/PizzaShack/pizza-shack-web/src/main/webapp/WEB-INF/web.xml.
2. Update the consumer key and client secret with the values obtained when generating the access token and save the file.

☒ SAML2
 ☒ OAuth 1.0a
 ☐ Implicit
 ☒ Refresh Token

☒ Client Credential
 ☐ Code
 ☒ Password

Callback URL

Update

Generating Access Tokens

The following cURL command shows how to generate an access token using the password grant type.

```
curl -k -d "grant_type=password&username=Username&password=Password" \
-H "Authorization: Basic V3FUT0ZNMzkwbTFLR3NSdWdrUDVSeXR4ZmtvYTYX0tPVEJFVWYwajhEQ0lEN3M0OVJjdGZnTmdh" \
https://172.17.0.1:8243/token
```

In a similar manner, you can generate an access token using the client credential grant type with the following cURL command.

```
curl -k -d "grant_type=client_credentials" \
-H "Authorization: Basic V3FUT0ZNMzkwbTFLR3NSdWdrUDVSeXR4ZmtvYTYX0tPVEJFVWYwajhEQ0lEN3M0OVJjdGZnTmdh" \
https://172.17.0.1:8243/token
```

Generate a Test Access Token

Access Token

Above token has a validity period of -1 seconds. And the token has (am_application_scope default) scopes.

Scopes

Validity period

Re-generate

```

</context-param>
<context-param>
    <param-name>consumerKey</param-name>
    <param-value>XXXXXXXXXXXXXXXXXXXXXXXXXXXX</param-value>
</context-param>
<context-param>
    <param-name>consumerSecret</param-name>
    <param-value>YYYYYYYYYYYYYYYYYYYYYYYYYYYY</param-value>
</context-param>

```

- Open a Command Line Interface.
- Go to the SAMPLE/PizzaShack/pizza-shack-web folder and run `mvn clean install`. - this will build the application.
- Copy the `pizzashack.war` file from the target folder to `<AS_HOME>/repository/deployment/server/webapps/` so that it is deployed.

Test the PizzaShack Application

To test the application, log in to the APIM Management Console and do the following.

Create a role named webuser and grant it **Login** permission:

1. Log in to the APIM admin console, which is available by default at: <https://localhost:9443/carbon>. You can log in to the console using the default admin/admin credentials.
2. Click **Main > Users and Roles > Add**.
3. Click **Add New Role**.
4. Provide **webuser** as the role name.
5. Click **Next** - You will be presented with a list of permissions. Select the **Login** permission.
6. Click **Finish**.

Create a user named john and assign him the webuser role:

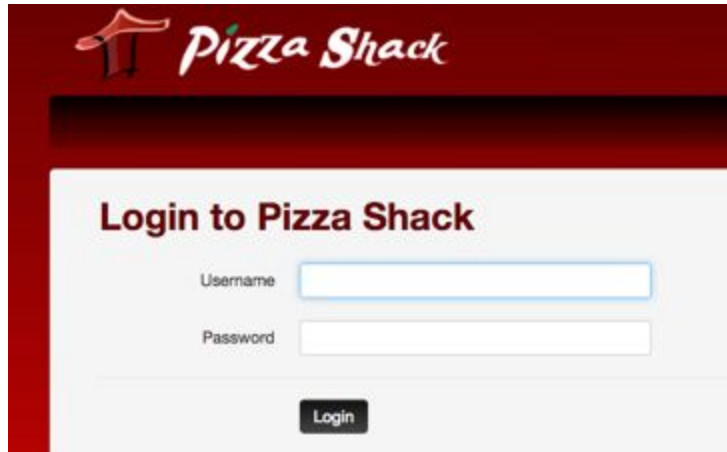
1. Click **Main > Users and Roles > Add**.
2. Click **Add New User**.
3. Provide user name (john) and password.
4. Click **Next**.
5. Select the webuser role.
6. Click **Finish**.

Create a user named mike. Do not assign him any roles:

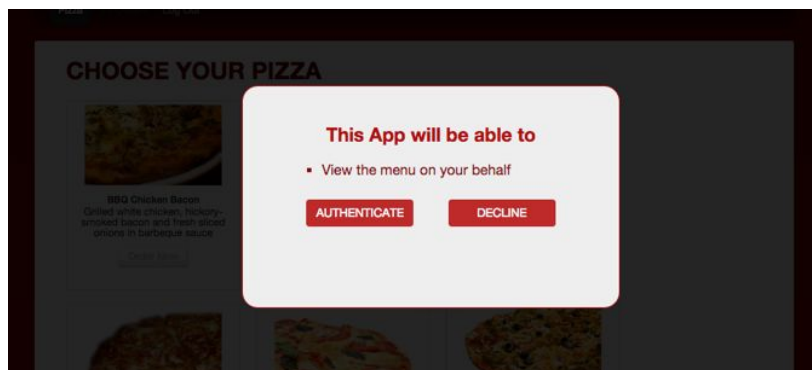
1. Click **Main > Users and Roles > Add**.
2. Click **Add New User**.
3. Provide user name (mike) and password.
4. Click **Finish**.

To access the application:

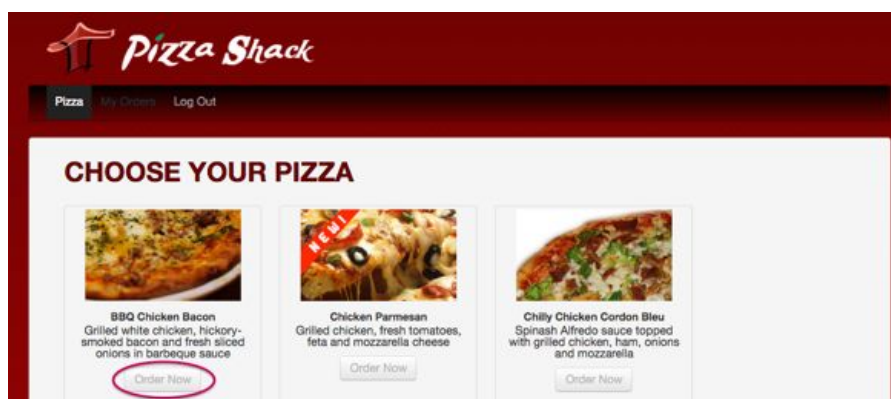
1. Go to: <http://localhost:9766/pizzashack/login.jsp> and log in as mike. (9766 is used because the default port is 9763 and a port offset of 3 was defined for the AS server).



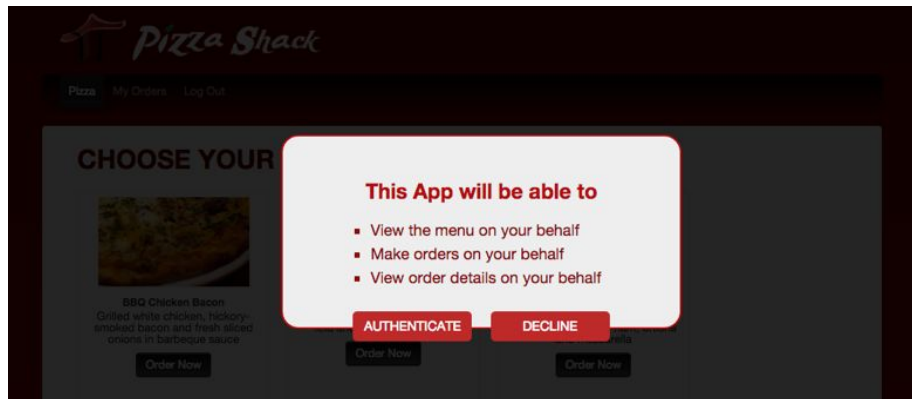
When mike logs in, he will not be able to get a token having the **order_pizza** scope since he doesn't have the **webuser** role. As a result, you will see the screen below.



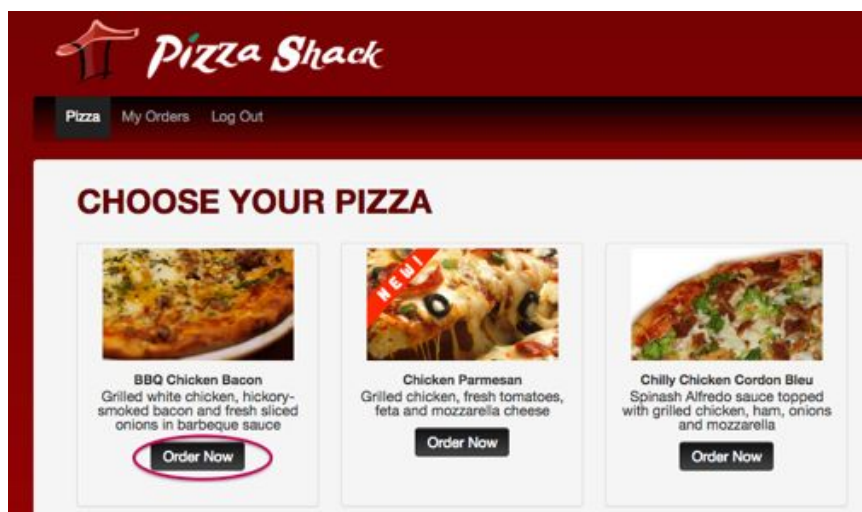
Note that the **Order Now** button is disabled.



2. Log in as john. Since user john has the **webuser** role, he is capable of getting an access token which has the **order_pizza** scope and can invoke the /order resource of the PizzaAPI. When you log in as john, you will see the screen below.



Note that the **Order Now** button is enabled.



Expected Outcome

In this exercise, the API was tested using cURL and the PizzaShack web application was built and deployed in WSO2 Application Server. The web application was tested using 2 users with different levels of permission.

Lab: Add Throttling Policy

Training Objective

Add new throttling tiers and define extra properties to throttling tiers using the Management Console. Throttling allows you to limit the number of hits to an API during a given period of time.

Business Scenario

PizzaShack's popularity is overwhelming and the amount of requests is increasing and they have decided to allow up to 100 requests per minute.

High Level Steps

- Add throttling policy

Detailed Instructions

Add Throttling Policy

1. Log in to the API Manager Admin Portal (<https://localhost:9443/admin/>) (admin/admin) and click **THROTTLE POLICIES**.
2. Select **SUBSCRIPTION TIERS** and **ADD TIER** at the top.

WSO₂ Admin Portal

THROTTLE POLICIES / SUBSCRIPTION TIERS

ADD TIER

Subscription Tier List

Filter by ...

Name	Quota Policy	Quota	Unit Time	Rate Limit	Time Unit	Edit	Delete
Bronze	requestCount	1000	1 min	NA	NA	Edit	Delete
Gold	requestCount	5000	1 min	NA	NA	Edit	Delete
Silver	requestCount	2000	1 min	NA	NA	Edit	Delete
Unauthenticated	requestCount	500	1 min	NA	NA	Edit	Delete

Show 10 entries Showing 1 to 4 of 4 entries

3. Enter the following information

× ← GO BACK

Add Subscription Tier

General Details

Name *

Platinum

Description

Quota Limits

☒ Request Count ☐ Request Bandwidth

Request Count *

25

Unit Time *

1

Minute(s)

Burst Control (Rate Limiting)

Request Count

5

Request/s

Policy Flags

Stop On Quota Reach

☒

Billing Plan

Free

Custom Attributes

+ Add Custom Attribute

Permissions

Roles *

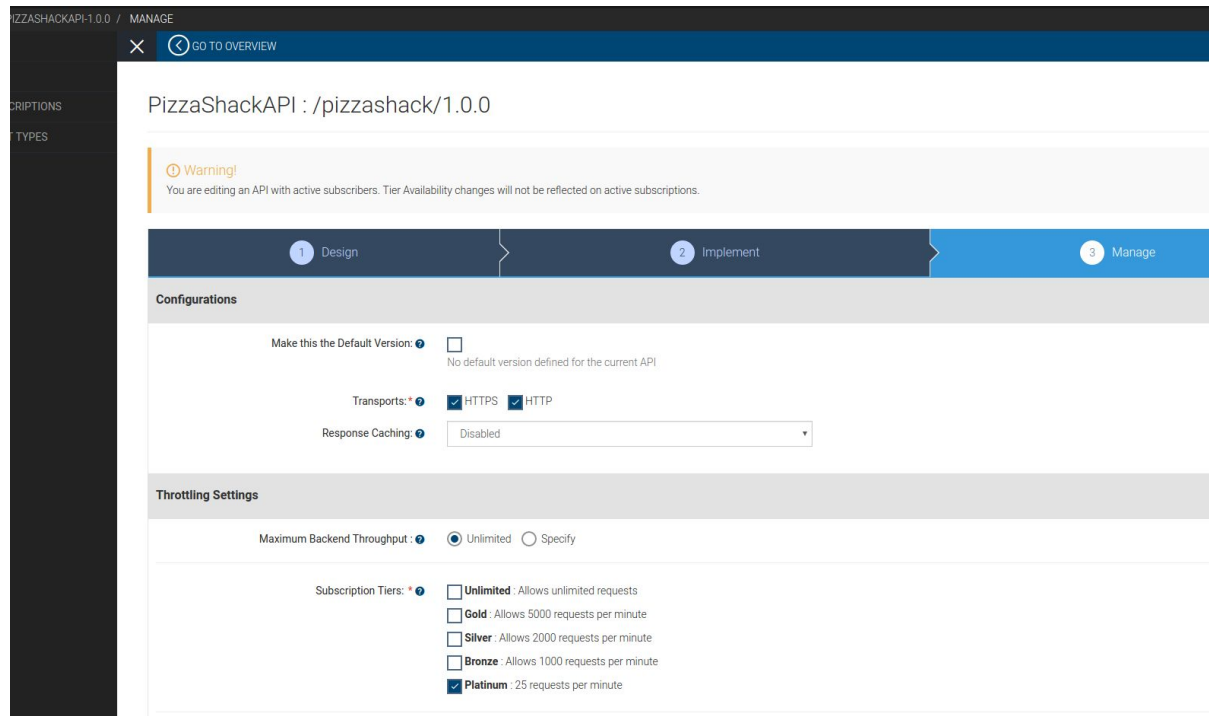
Internal/everyone

This tier is Allowed for above roles.

☒ Allow ☐ Deny

Save Cancel

In the API Publisher, edit the PizzaShack API and note that Platinum can now be selected under **Subscription Tiers** in the **THROTTLING SETTINGS** section.



Expected Outcome

Your new throttling policy (Platinum) is now successfully saved as an execution plan used by WSO2 API Manager. You can view this new throttle tier available for selection when creating a new API through the API Publisher or when editing an existing API.

Lab: Analyze Runtime Statistics

Training Objective

Set up WSO2 API Manager Analytics server to collect and analyze runtime statistics from the API Manager.

Business Scenario

PizzaShack Limited needs to monitor the use of their online portal and want to generate statistics about how many times consumers access the API.

High Level Steps

- Configure WSO2 API Manager
- View published statistics

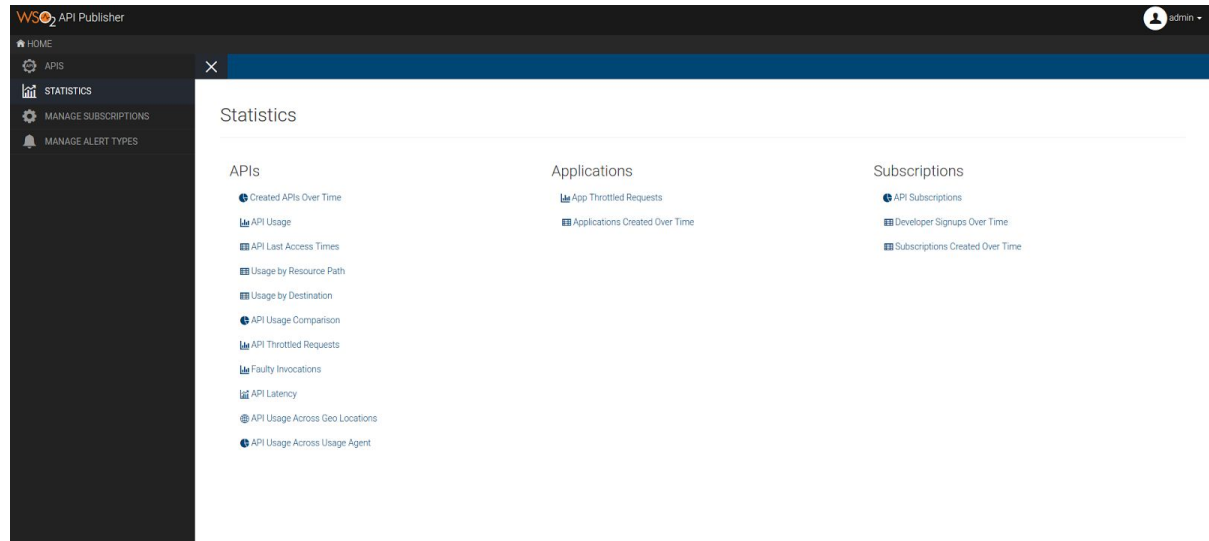
Detailed Instructions

Configure WSO2 API Manager

1. Open the <APIM_HOME>/repository/conf/api-manager.xml file and set the <Enabled> element in the <Analytics> section to true.
API Manager Analytics comes with a default port offset of 1. It points to an H2 RDBMS database which is used by the API Manager.
2. To run the setup, extract API Manager Analytics 2.0.0 to the same directory as the . Then first start the analytics server and then start the API Manager server

View Published Statistics

1. Invoke the API.
2. Log in to the API Publisher.
3. Click on **STATISTICS** to view the statistics.



Expected Outcome

As a result of this exercise, events are generated based on the API invocations and stored in the RDBMS tables shared with the API Manager and API Manage Analytics server.

Lab: Managing Alerts with Real-Time Analytics

Training Objective

Generate alerts for a scenario when the tier limit is hit frequently.

Business Scenario

PizzaShack Limited needs to generate alerts if users exceed their tier limits frequently.

High Level Steps

- Generate and view alerts
- Configure alert generation related parameters

Generate Alerts

Note: API Manager and API Manager Analytics must be configured for analytics. This has been covered in the previous exercise.

1. Create a subscriber level throttling tier with a small number of requests per minute. E.g., 2 requests per minute.
2. Log in to the API Publisher.
3. Click on the PizzaShack API and click **EDIT API**.
4. Click **Manage**.
5. Apply the new throttling tier and click **Save and Publish**.
6. Log in to API Store and select the PizzaShack API.
7. Subscribe to an application using the new tier.
8. Invoke the API rapidly till it throttles out. After 2 requests, you should get a throttled out message. Keep on doing request (more than 10) to generate an alert (by default an alert is generated when there are 10 alerts more than the limit).
9. Log in to Admin portal (<https://localhost:9443/admin/>) and select the alert icon on the top right corner and you will see a generated alert.

WSO2 Admin Portal

Alerts History

Alert Type: All Alerts

Filter by ...

Alert Timestamp	Type	Message	Severity
Wed Aug 03 2016 10:20:53 GMT+0530 (IST)	Tier Crossing	The Application DefaultApplication owned by admin@carbon.super frequently goes beyond the allocated quota when accessing the API admin-PizzaShackAPIv1.0.0.	info

Show 10 entries

Showing 1 to 1 of 1 entries

Configure Alert Generation Related Parameters

1. Log in to WSO2 API Manager Analytics carbon console (<https://localhost:9444/carbon>).
2. Select **Dashboard > Template Manager**.
3. In the **Template Manager**, select **APIMAnalytics**. This will open a configuration page for all the alert types.

The screenshot shows the 'Template Manager' interface for 'APIMAnalytics'. It features a 'Deployed Scenarios' table with columns for Scenario Name, Description, Scenario Type, and Actions. A 'Create New Scenario' button is visible above the table.

Scenario Name	Description	Scenario Type	Actions
APIRequestPatternChangeAnalysisMetric	Detection of request pattern changes	RequestPatternChangeDetection	Delete Edit
AbnormalRequestCountDetection	Detects abnormal request count	AbnormalRequestCountDetection	Delete Edit
AbnormalResponseAndBackendTimeDetection	Detects abnormal backend time and response time	AbnormalResponseAndBackendTimeDetection	Delete Edit
AbnormalTierAvailabilityAlert	Abnormal tier usage alerts	AbnormalTierUsageAlert	Delete Edit
ConfigureAccessToken	Configure access token to analyze data	ConfigureAccessToken	Delete Edit
FrequentTierLimitHitting	Frequent hitting of tier limits	FrequentTierLimitHitting	Delete Edit
HealthAvailabilityPerMin	Monitoring API health	HealthAvailabilityPerMinAlert	Delete Edit
MarkovStateClassifier	Classifier configurations for Request Patterns	MarkovStateClassifier	Delete Edit
DemocratDevAPI	Democrat dev API scenario	DemocratDevAPI	Delete Edit

4. To edit parameters related to the frequent tier limit hitting alert click **Edit** in the **FrequentTierLimitHitting** section

The screenshot shows the 'Edit Scenario' page for 'FrequentTierLimitHitting'. It includes fields for Scenario Type, Scenario Name, and Description. Below these are 'PARAMETER CONFIGURATIONS' for Time Interval, Alert Suppression Period in Minutes, No Of Tier Crossings, and Severity Level. An 'Update Scenario' button is at the bottom right.

Scenario Type: **FrequentTierLimitHitting**
 Detects frequent hitting of the tier limit

Scenario Name: **FrequentTierLimitHitting**

Description: **Frequent hitting of tier limits**

PARAMETER CONFIGURATIONS

Time Interval: **1 day**
 Time period (day/min/sec) of which the request count would be taken.

Alert Suppression Period in Minutes: **10**
 Time period in minutes to wait before resending the same alert

No Of Tier Crossings: **10**
 Max Number of tier crossings for the given time window

Severity Level: **3**
 Severity level of the alert(1:severe,2:moderate,3:mild)

Update Scenario

Expected Outcome

As a result of this exercise, Throttled out events are generated based on the API invocations and once the pre-defined tier crossing limit is exceeded, an alert is generated.

Lab: Analyzing Logs with the Log Analyzer

Training Objective

Analyze logs using the Log Analyzer.

Business Scenario

The Admin user wants to analyze log files to check any errors that happened while running the system.

High Level Steps

- Configure WSO2 API Manager and WSO2 API Manager Analytic.
- Analyze logs using log analyzer in Admin portal.

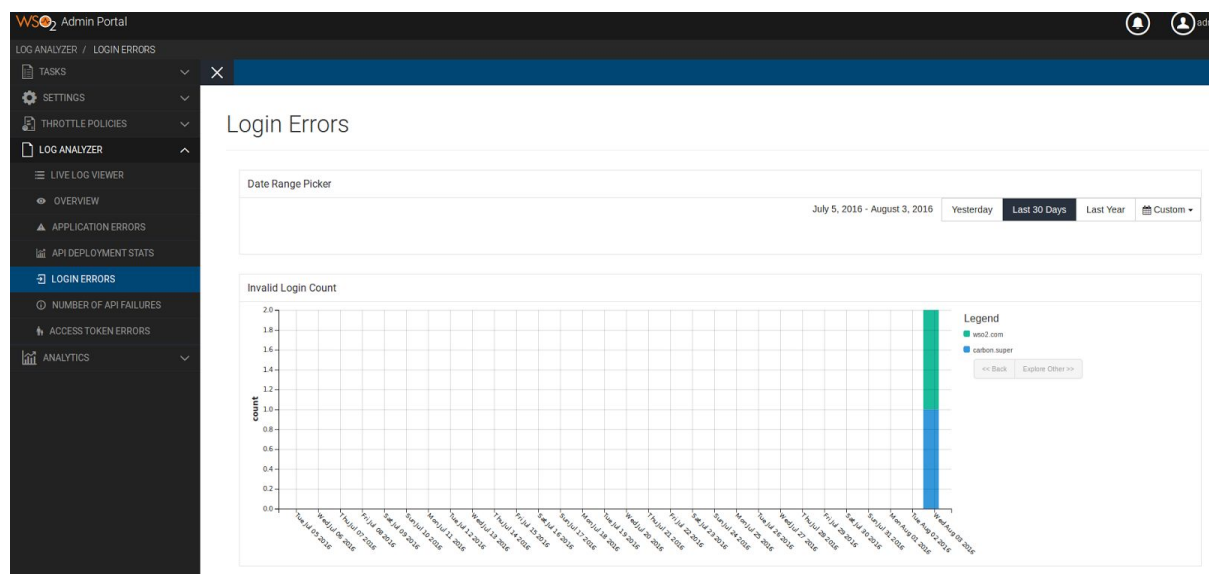
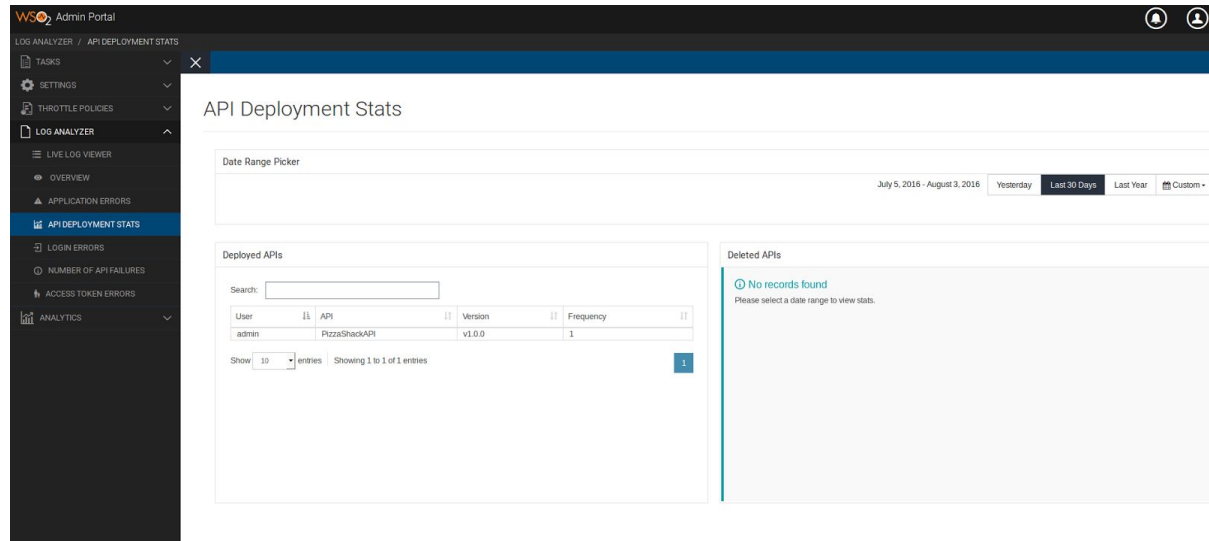
Configure WSO2 API Manager and WSO2 API Manager Analytics

Note: The wso2am-2.0.0/repository/conf/api-manager.xml file and must have the <Enabled> element in <Analytics> section to true.

1. Open the <APIM_HOME>/repository/conf/log4j.properties file. Add the DAS_AGENT to the end of the root logger.
2. First start the WSO2 API Manager Analytics server and then start the WSO2 API Manager.

Generate logs

1. Try out some activities using the API Manager (Publish an API, log into the store using invalid credentials, etc.)
2. Log into the Admin Portal (<https://localhost:9443/admin/>) and select the **LOG ANALYZER** section to view different logs.



The logs get updated every 15min. To view logs without waiting you can manually execute the analytics scripts by following the steps below:

1. Log in to WSO2 API Manager Analytics server (<https://localhost:9444/carbon>).
2. Click **Main > Manage > Batch Analytics > Scripts**.
3. Under **Scripts** execute the **APIM_LOGANALYZER_SCRIPT**.

Lab: Using Published APIs

Training Objective

In this section you will learn how to invoke, manage and control published APIs through the terminal using cURL.

Business Scenario

PizzaShack Limited needs to improve their delivery time in order to provide a better service. They want to use the Google Directions API to assist the delivery team to identify routes with traffic, find the best possible route and reduce delays in finding the customer's location. They have also decided to improve their PizzaShack web portal in order to call the Google Direction API via API Manager to show the best route information.

High Level Steps

- Create published API
- Publish new API
- Create new application
- Create new subscription

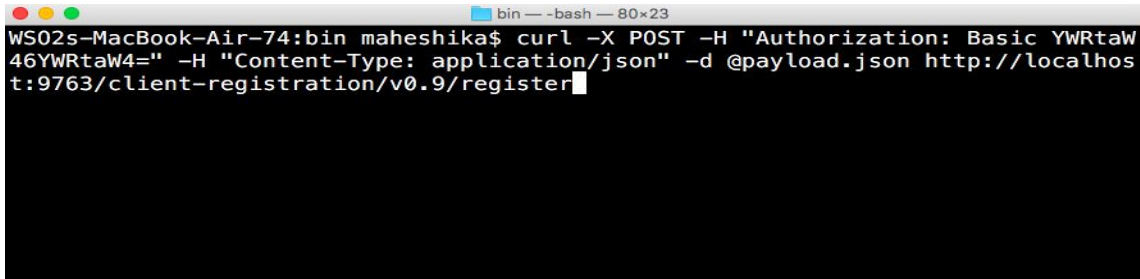
Detailed Instructions

Create Published API

1. Create the payload.json file in the [APIM_HOME]/bin folder with the following text and save.

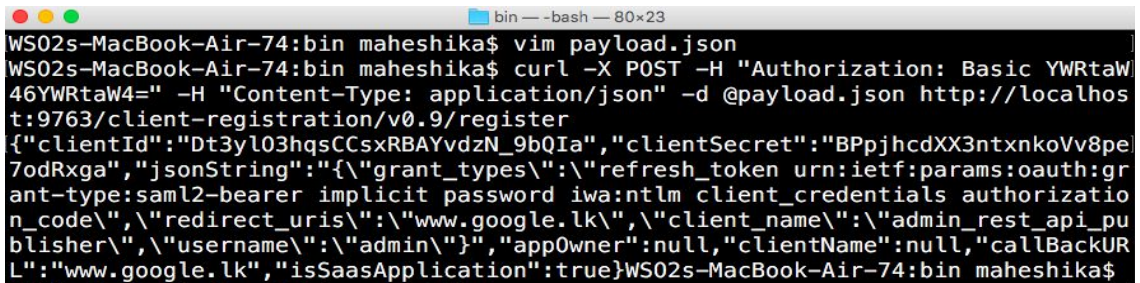
```
{
  "callbackUrl": "www.google.lk",
  "clientName": "rest_api_publisher",
  "tokenScope": "Production",
  "owner": "admin",
  "grantType": "password refresh_token",
  "saasApp": true
}
```
2. Open a Command Line Interface.
3. Navigate to the {APIM_HOME/bin} folder using the command.
4. Give the cURL command for client registration. (Make sure the API Manager server is running before doing this).

```
curl -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H
"Content-Type: application/json" -d @payload.json
http://localhost:9763/client-registration/v0.09/register
```



```
WS02s-MacBook-Air-74:bin maheshika$ curl -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H "Content-Type: application/json" -d @payload.json http://localhost:9763/client-registration/v0.09/register
```

The following response is displayed.



```
WS02s-MacBook-Air-74:bin maheshika$ vim payload.json
WS02s-MacBook-Air-74:bin maheshika$ curl -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H "Content-Type: application/json" -d @payload.json http://localhost:9763/client-registration/v0.09/register
{"clientId":"Dt3yl03hqsCCsxRBAYvdzN_9bQIa","clientSecret":"BPpjhdXX3ntxnkoVv8pe7odRxga","jsonString":{"grant_types":["refresh_token","urn:ietf:params:oauth:grant-type:saml2-bearer","implicit","password","iwa","ntlm","client_credentials","authorization_code","redirect_uri":["www.google.lk"],"client_name":["admin_rest_api_publisher"],"username":["admin"],"appOwner":null,"clientName":null,"callbackURL":"www.google.lk","isSaasApplication":true}}
```

5. Copy the clientId and clientSecret from the console and encode them to generate a key using <https://www.base64encode.org/> or any other encoder.

The screenshot shows a web browser at <https://www.base64encode.org>. The page has a green header with 'Decode' and 'Encode' tabs. The 'Encode' tab is active. Below the header, the text 'Encode to Base64 format' is displayed, followed by 'Simply use the form below'. A large text input field contains the string: 'Dt3yIO3hqsCCsxRBAYvdzN_9bQla:BPpjhcdXX3ntxnkoVv8pe7odRxga'. Below the input field, there is a green button labeled '> ENCODE <' and a dropdown menu set to 'UTF-8'. To the right of the dropdown, it says '(You may also select output charset.)'. Below these, the encoded output is shown in a text box: 'RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpCUHBqaGNkWFgzbnR4bmtvVnY4cGU3b2RSeGdh'.

Note : Provide a colon (:) between the clientId and clientSecret on Base64.

6. Type the following authorization invocation cURL command on the terminal with the encoded clientId and clientSecret for the Authorization Basic value and scope=apim_create as the scope.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope
=apim:api_create" -H "Authorization: Basic
RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpCUHBqaGNkWFgzbnR4
bmtvVnY4cGU3b2RSeGdh" https://127.0.0.1:8243/token
```

The screenshot shows a terminal window on a Mac. The command executed is: `curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:api_create" -H "Authorization: Basic RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpCUHBqaGNkWFgzbnR4bmtvVnY4cGU3b2RSeGdh" https://127.0.0.1:8243/token`. The output is a JSON object: `{"access_token":"d0194b3631fd447b4c3e36f8eda10c0","refresh_token":"f39a92ced2f731ac33234e59dfc6b235","scope":"apim:api_create","token_type":"Bearer","expires_in":3600}`. The terminal prompt is `WS02s-MacBook-Air-74:bin maheshika$`.

Note : The scope is given depending on the requirement .A new token is required each time a different scope is used and each token is valid only for 1 hour

7. Create the data.json file in the [APIM_HOME]/bin folder.

8. Add the following code to data.json and save

```
{
  "sequences": [],
  "tiers": [
    "Bronze",
    "Gold"
  ],
  "thumbnailUrl": null,
  "visibility": "PUBLIC",
  "visibleRoles": [],
  "visibleTenants": [],
  "cacheTimeout": 300,
  "endpointConfig":
    "{ \"production_endpoints\": { \"url\": \"http://maps.google.com/maps/api/directions/\", \"config\": null }, \"endpoint_type\": \"http \" },
    \"subscriptionAvailability\": null,
    \"subscriptionAvailableTenants\": [],
    \"destinationStatsEnabled\": \"Disabled\",
    \"apiDefinition\":
      \"{ \"paths\": { \"/*\": { \"get\": { \"x-auth-type\": \"Application\", \"x-throttling-tier\": \"Unlimited\", \"responses\": { \"200\": { \"description\": \"OK\" } } } }, \"x-wso2-security\": { \"apim\": { \"x-wso2-scopes\": [] } }, \"swagger\": \"2.0\", \"info\": { \"title\": \"GoogleDirectionsAPI\", \"description\": \"Calculates directions between locations\", \"contact\": { \"email\": \"ApiPublisher@pizzashack.com\", \"name\": \"ApiPublisher\", \"version\": \"Beta\" } } }\",
    \"responseCaching\": \"Disabled\",
    \"isDefaultVersion\": true,
    \"gatewayEnvironments\": \"Production and Sandbox\",
    \"businessInformation\": {
      \"technicalOwner\": \"ApiCreator\",
      \"technicalOwnerEmail\": \"ApiCreator@pizzashack.com\",
      \"businessOwner\": \"ApiPublisher\",
      \"businessOwnerEmail\": \"ApiPublisher@pizzashack.com\"
    },
    \"transport\": [
      \"http\",
      \"https\"
    ],
    \"tags\": [
      \"phone\",
      \"multimedia\",
      \"mobile\"
    ],
    \"provider\": \"admin\",
    \"version\": \"Beta\",
    \"description\": \"Calculates directions between locations\",
```



```

    "name": "GoogleDirectionsAPI",
    "context": "/googledirections"
  }

```

- Run the following cURL command to create the api using data.json. Type the invoked access token as the Authorization Bearer value.

```

curl -H "Authorization: Bearer
9d0194b3631fd447b4c3e36f8eda10c0" -H "Content-Type:
application/json" -X POST -d @data.json
http://127.0.0.1:9763/api/am/publisher/v0.10/apis

```

```

WS02s-MacBook-Air-74:bin maheshika$ curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:api_create" -H "Authorization: Basic RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpcUHBqaGNkWFgzbnR4bmtvVnY4cGU3b2RSeGdh" https://127.0.0.1:8243/token
{"access_token":"9d0194b3631fd447b4c3e36f8eda10c0","refresh_token":"f39a92ced2f731ac33234e59dfc6b235","scope":"apim:api_create","token_type":"Bearer","expires_in":3600}WS02s-MacBook-Air-74:bin maheshika$
WS02s-MacBook-Air-74:bin maheshika$
WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer 9d0194b3631fd447b4c3e36f8eda10c0" -H "Content-Type: application/json" -X POST -d @data.json http://127.0.0.1:9763/api/am/publisher/v0.9/apis

```

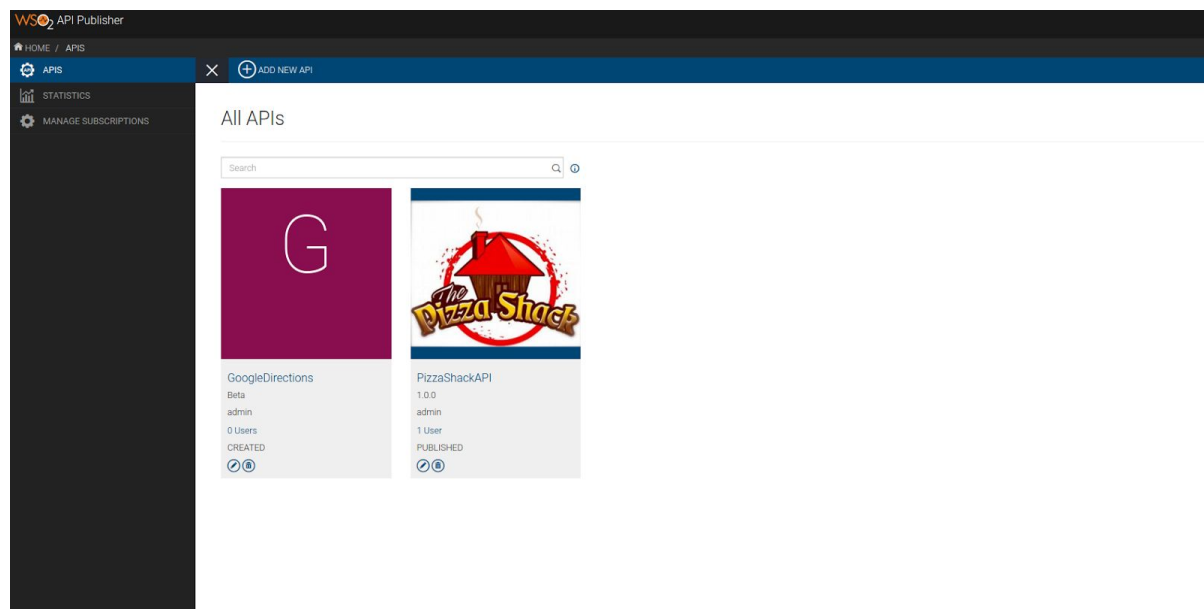
- The response which includes the id of the API will be displayed.

```

bin -- bash -- 99x23
~/Documents/products/APIM/wso2am-1.10.0/bin -- java -sh wso2server.sh
bnR4bmtvVnY4cGU3b2RSe6dh" https://127.0.0.1:8243/token
{"access_token":"9d0194b3631fd447b4c3e36f8eda10c0","refresh_token":"f39a92ced2f731ac33234e59dfc6b23
5","scope":"apim:api_create","token_type":"Bearer","expires_in":3600}WS02s-MacBook-Air-74:bin mahes
hika$
WS02s-MacBook-Air-74:bin maheshika$
WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer 9d0194b3631fd447b4c3e36f8eda10c0
" -H "Content-Type: application/json" -X POST -d @data.json http://127.0.0.1:9763/api/am/publisher/
v0.9/apis
{"name":"GoogleDirectionsAPI","context":"/googledirections","id":"4d0b513e-71d4-489e-9681-31a9178b
e18","status":"CREATED","description":"Calculates directions between locations","provider":"admin",
"version":"Beta","tags":["multimedia","phone","mobile"],"transport":["http","https"],"sequences":[]
,"thumbnailUrl":null,"visibility":"PUBLIC","visibleRoles":["],"visibleTenants":["],"cacheTimeout":300
,"endpointConfig":{"production_endpoints":{"url":"http://maps.google.com/maps/api/directions/
","config":null},"endpoint_type":"http"},"subscriptionAvailability":null,"subscriptionAvai
lableTenants":["],"destinationStatsEnabled":"Disabled","tiers":["Bronze","Gold"],"apiDefinition":{"
paths":{"/*":{"get":{"x-auth-type":"Application"},"responses":{"200":{"description
":"OK"},"x-throttling-tier":"Unlimited"}}},"x-wso2-security":{"apim":{"x-wso2-scopes":
[]},"swagger":{"2.0"},"info":{"contact":{"name":"xx","email":"xx@ee.com"},"descript
ion":"Calculates directions between locations","title":"GoogleDirectionsAPI","version":"B
eta"},"responseCaching":"Disabled","isDefaultVersion":true,"gatewayEnvironments":"Production and
Sandbox","businessInformation":{"technicalOwner":"ApiCreator","technicalOwnerEmail":"ApiCreator@pi
zzashack.com","businessOwner":"ApiPublisher","businessOwnerEmail":"ApiPublisher@pizzashack.com"}}}WS0
2s-MacBook-Air-74:bin maheshika$

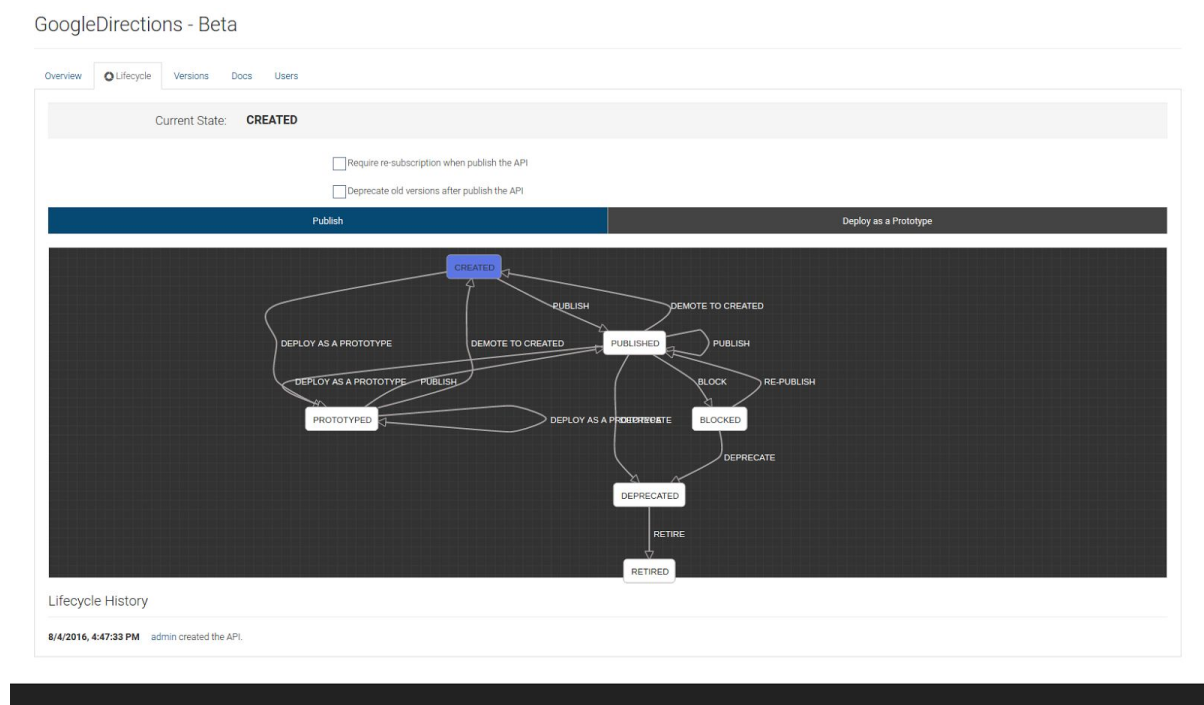
```

11. Refresh <https://localhost:9443/publisher/> and the GoogleDirectionAPI will be displayed on the dashboard.



Publish API

1. Log in as admin to <https://localhost:9443/publisher>.
2. Click on GoogleDirectionsAPI and select the **Lifecycle** tab. The lifecycle will be indicated as **Created**.



3. Go to the terminal and invoke a new authorization token using the following cURL command with the previously encoded string as the Authorization Basic value and scope=apim_publish.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope=apim:api_publish" -H "Authorization: Basic
RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpCUHBqaGNkWFgzbnR4bmtvVnY4cGU3b2
RSeGdh" https://127.0.0.1:8243/token
```

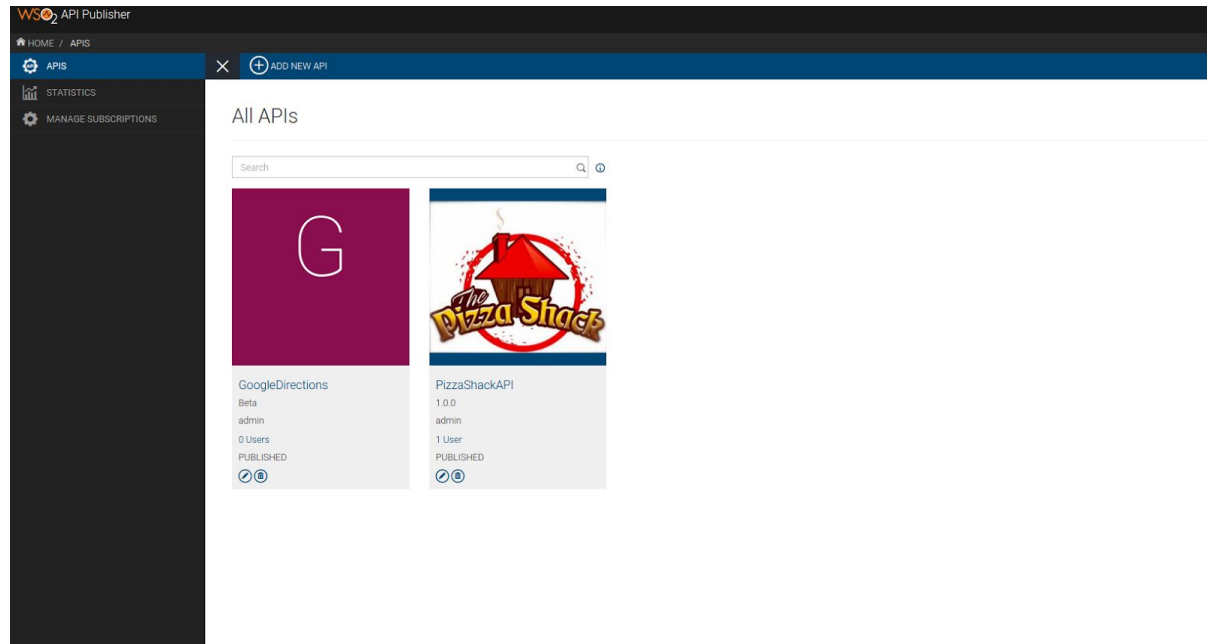
```
WS02s-MacBook-Air-74:bin maheshika$ curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:api_publish" -H "Authorization: Basic RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpCUHBqaGNkWFgzbnR4bmtvVnY4cGU3b2RSeGdh" https://127.0.0.1:8243/token
```

4. Type the following cURL command to publish the API. Modify the Authorization Bearer and apild as required.

```
curl -H "Authorization: Bearer aefcf3e9efb6bc09d34451a0824ed6e8" -X POST "http://127.0.0.1:9763/api/am/publisher/v0.10/apis/change-lifecycle?apiId=4d0b513e-71d4-489e-9681-31a9178bc189&action=Publish"
```

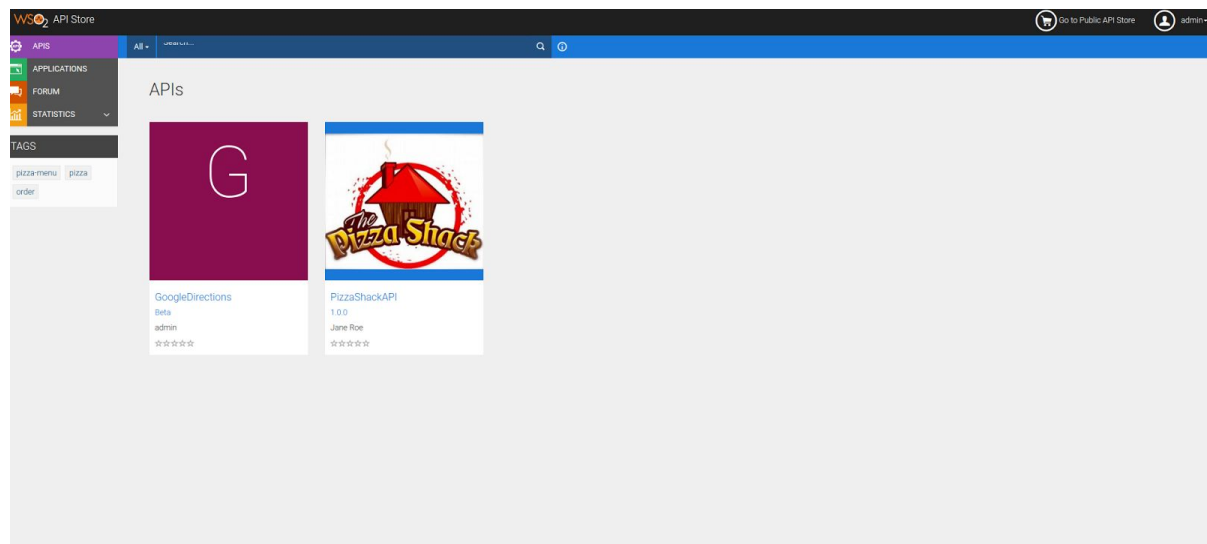
```
WS02s-MacBook-Air-74:bin maheshika$ curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:api_publish" -H "Authorization: Basic RHQzeWxPM2hxc0NDc3hSQkFZdmR6Tl85YlFJYTpCUHBqaGNkWFgzbnR4bmtvVnY4cGU3b2RSeGdh" https://127.0.0.1:8243/token
{"access_token":"aefcf3e9efb6bc09d34451a0824ed6e8","refresh_token":"fa698295033918d92ad9754c8716c1ade","scope":"apim:api_publish","token_type":"Bearer","expires_in":3252}
WS02s-MacBook-Air-74:bin maheshika$
WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer aefcf3e9efb6bc09d34451a0824ed6e8" -X POST "http://127.0.0.1:9763/api/am/publisher/v0.9/apis/change-lifecycle?apiId=4d0b513e-71d4-489e-9681-31a9178bc189&action=Publish"
WS02s-MacBook-Air-74:bin maheshika$
```

5. Go to the Publisher and refresh. Now the status of the GoogleDirectionsAPI will be displayed as **Published**.



Create New Application

1. Log in as admin to the Store ([https://localhost:9443/store/]) and click **carbon.super** on the dashboard. The GoogleDirectionsAPI icon is visible.



2. Open a Command Line Terminal.
3. Modify the [API HOME]/bin/payload.json file clientName to rest_api_store and save;

```
{
  "callbackUrl": "www.google.lk",
  "clientName": "rest_api_store",
  "tokenScope": "Production",
  "owner": "admin",
  "grantType": "password refresh_token",
  "saasApp": true
}
```

4. Open a Command Line Interface and type the following cURL command for client registration.

```
curl -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H
"Content-Type: application/json" -d @payload.json
http://localhost:9763/client-registration/v0.10/register
```

5. Encode the client ID and secret values as before.
6. Invoke the authorization token using the following cURL command. Replace the Authorization Basic value.

```
curl -k -d
"grant_type=password&username=admin&password=admin&scope=apim:subscribe" -H "Authorization: Basic
ZktWNWFJTxFkc1FDRHduV1NMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ0Fvd1YzdE
pBVzU5eThh" https://127.0.0.1:8243/token
```



```
WS02s-MacBook-Air-74:bin maheshika$ curl -X POST -H "Authorization: Basic YWRtaW46YWRtaW4=" -H "Content-Type: application/json" -d @payload.json http://localhost:9763/client-registration/v0.9/register
{"clientId":"fKV5aIMqdrQCDwnWSL8LontHNo8a","clientSecret":"jRomRGVPhW2uPgAowV3tJlAW59y8a","jsonString":{"grant_types":["refresh_token","urn:ietf:params:oauth:grant-type:saml2-bearer","implicit","password","iwa","ntlm","client_credentials","authorization_code","redirect_uri"],"client_name":"admin_rest_api_store","username":"admin"},"appOwner":null,"clientName":null,"callbackURL":"www.google.lk","isSaasApplication":true}
WS02s-MacBook-Air-74:bin maheshika$
WS02s-MacBook-Air-74:bin maheshika$ curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:subscribe" -H "Authorization: Basic ZktWNWFJTxFkcjFDRHduV1NM0ExvbnRITm84YTpqUm9tUkdWUGhXMmVQZ0Fvd1YzdEpBVzU5eThh" https://127.0.0.1:8243/token
{"access_token":"cbe53aefd029a4a7eaf79817308f4708","refresh_token":"ec8189452fffa284c0b7fbc991043fff","scope":"apim:subscribe","token_type":"Bearer","expires_in":3600}
WS02s-MacBook-Air-74:bin maheshika$
```

7. Remove the content in the [APIM_HOME]/bin/data.json file and add the following:

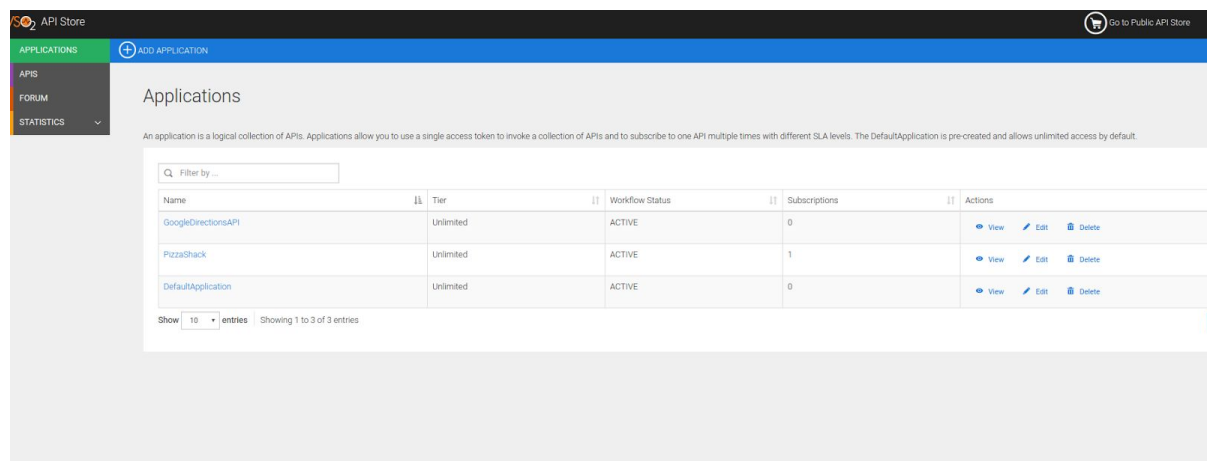
```
{
  "groupId": "",
  "subscriber": "admin",
  "throttlingTier": "Unlimited",
  "description": "GoogleDirectionsAPI App",
  "status": "APPROVED",
  "name": "GoogleDirectionsAPI"
}
```

8. Run the following cURL command to create an application. Replace the Authorization Bearer value with the access token generated above.

```
curl -H "Authorization: Bearer cbe53aefd029a4a7eaf79817308f4708"
-H "Content-Type: application/json" -X POST -d @data.json
"http://127.0.0.1:9763/api/am/store/v0.10/applications"
```

```
WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer cbe53aefd029a4a7eaf79817308f4708" -H "Content-Type: application/json" -X POST -d @data.json "http://127.0.0.1:9763/api/am/store/v0.9/applications"
{"name":"GoogleDirectionsAPI","keys":[],"status":"APPROVED","description":"GoogleDirectionsAPI App","applicationId":"b45c9838-7b74-4fd1-8b5a-2252909a0342","groupId":null,"callbackUrl":null,"subscriber":"admin","throttlingTier":"Unlimited"}
WS02s-MacBook-Air-74:bin maheshika$
```

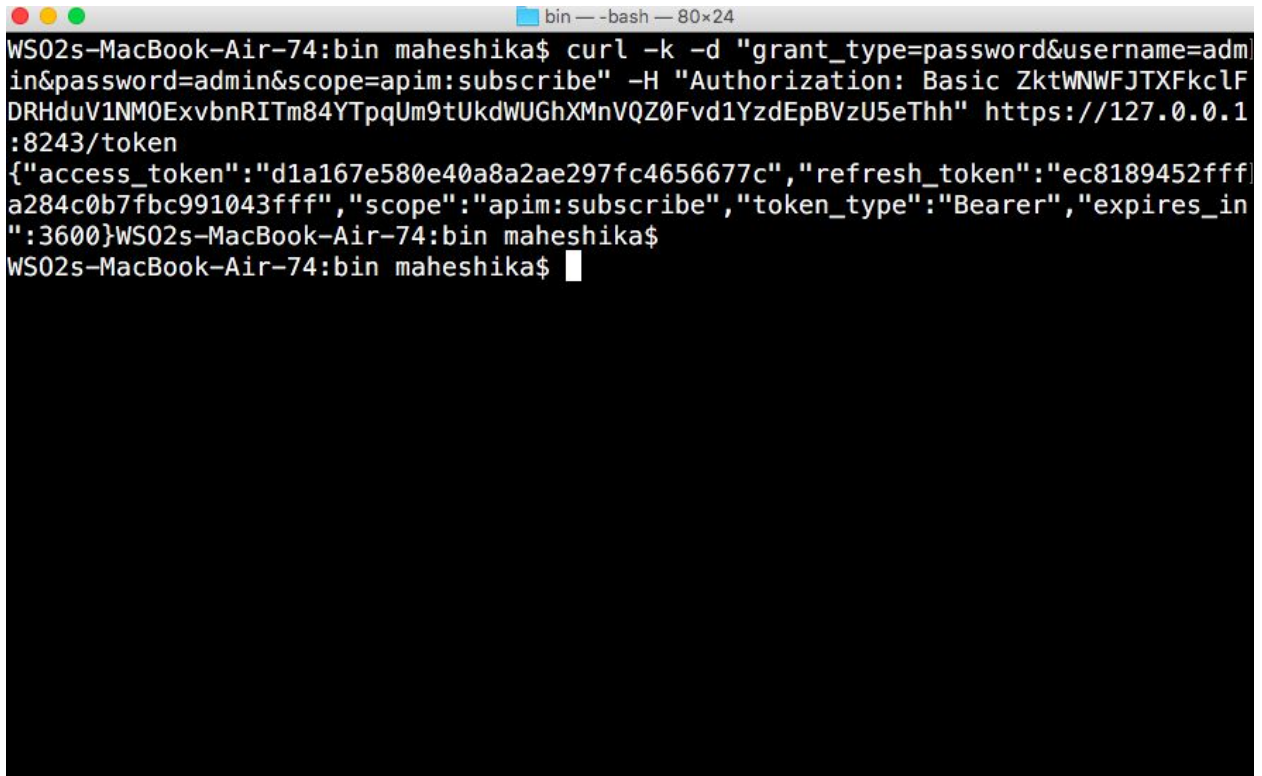
- The created application for GoogleDirectionsAPI will be listed under **Applications** in the Store.



Create New Subscription

- Invoke a new authorization token using the following cURL command. Replace the Authorization Basic value with the encoded string.


```
curl -k -d
"grant_type=password&username=admin&password=admin&scope=
apim:subscribe" -H "Authorization: Basic
ZktWNWFJTxFkclFDRHduVlNMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ
0Fvd1YzdEpBVzU5eThh" https://127.0.0.1:8243/token
```



```
WS02s-MacBook-Air-74:bin maheshika$ curl -k -d "grant_type=password&username=admin&password=admin&scope=apim:subscribe" -H "Authorization: Basic ZktWNWFJTxFkclFDRHduVlNMOExvbnRITm84YTpqUm9tUkdWUGhXMnVQZ0Fvd1YzdEpBVzU5eThh" https://127.0.0.1:8243/token
{"access_token":"d1a167e580e40a8a2ae297fc4656677c","refresh_token":"ec8189452fffa284c0b7fbc991043fff","scope":"apim:subscribe","token_type":"Bearer","expires_in":3600}WS02s-MacBook-Air-74:bin maheshika$
WS02s-MacBook-Air-74:bin maheshika$
```

2. Retrieve the list of applications using the following cURL command. Replace the Authorization Bearer value.

```
curl -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c"
"http://127.0.0.1:9763/api/am/store/v0.10/applications"
```

```
WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c" "http://127.0.0.1:9763/api/am/store/v0.9/applications"
{"previous":"","list":[{"name":"DefaultApplication","status":"APPROVED","description":null,"applicationId":"27825afe-bfef-4b77-a828-c0385d9f62fa","groupId":"","subscriber":"admin","throttlingTier":"Unlimited"},{"name":"GoogleDirectionsAPI","status":"APPROVED","description":"GoogleDirectionsAPI App","applicationId":"645c9838-7b74-4fd1-8b5a-2252909a0342","groupId":null,"subscriber":"admin","throttlingTier":"Unlimited"}],"next":"","count":2}WS02s-MacBook-Air-74:bin maheshika$ clear
```

3. Retrieve the list of APIs using the following cURL command. Replace the Authorization Bearer value.

```
curl -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c"
http://127.0.0.1:9763/api/am/store/v0.10/apis
```

```

WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c" http://127.0.0.1:9763/api/am/store/v0.9/apis
{"previous":"","list":[{"name":"GoogleDirectionsAPI","context":"/googledirections/Beta","id":"4d0b513e-71d4-489e-9681-31a9178bc189","status":"PUBLISHED","version":"Beta","provider":"admin","description":"Calculates directions between locations"}, {"name":"PizzaAPI","context":"/pizzashack/1.0.0","id":"bfdeb39c-1552-484d-bc45-82a2881dd322","status":"PUBLISHED","version":"1.0.0","provider":"apicreator","description":"Pizza API: Allows to manage pizza orders (create, update, retrieve orders)"}], "next":"","count":2}
WS02s-MacBook-Air-74:bin maheshika$

```

4. Replace the test in the [APIM_HOME]/bin/data.json file with the following. Give the retrieved apilIdentifier and applicationId.

```

{
  "tier": "Gold",
  "apiIdentifier": "4d0b513e-71d4-489e-9681-31a9178bc189",
  "applicationId": "645c9838-7b74-4fd1-8b5a-2252909a0342"
}

```

5. Type the following cURL command to create a new subscription replacing the Authorization Bearer value.

```

curl -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c" -H
"Content-Type: application/json" -X POST -d @data.json
"http://127.0.0.1:9763/api/am/store/v0.10/subscriptions"

```

```

WS02s-MacBook-Air-74:bin maheshika$ curl -H "Authorization: Bearer d1a167e580e40a8a2ae297fc4656677c" -H "Content-Type: application/json" -X POST -d @data.json "http://127.0.0.1:9763/api/am/store/v0.9/subscriptions"
{"apiIdentifier":"admin-GoogleDirectionsAPI-Beta","status":"UNBLOCKED","applicationId":"645c9838-7b74-4fd1-8b5a-2252909a0342","tier":"Gold","subscriptionId":"fb0559f-908e-4777-8ab5-df56845a0b97"}
WS02s-MacBook-Air-74:bin maheshika$

```

6. Go to the Store and click **APPLICATIONS**.
7. Select the GoogleDirectionsAPI from the list. Select the **Subscriptions** tab. There will be a subscription tier field with a Gold Subscription.

