Practical Exercise: Creating and Publishing an API through the Publisher

Training Objective

Learn how to create an API, add API documentation, and publish it to the Developer Portal and deploy to gateway using the Publisher.

Business Scenario

After setting up the API-M, the API is created, published and deployed through the API Publisher in order to make it discoverable and subscribable from the store (Developer Portal).

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 Publisher
- Add documentation
- Deploy to gateway
- Publish the APIs

Detailed Instructions

Adding the PizzaShack API to the Publisher

Now that we set up the API-M and added users, we are ready to publish the API the PizzaShack application requires.

To add the API to the publisher, follow these steps:

1. Open the API Publisher web application from https://localhost:9443/publisher.
2. Log in using the user with the creator role you defined previously (creator).
3. Go to **APIs** in the Left Navigation.
4. Select **Start From Scratch** under REST APIs.
5. Provide information on the API as per the table below and click **Create**.

Field	Value	Description
Name	**PizzaShack**	Name of API as you want it to appear in the API store (Developer Portal)
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)
Endpoint	**https://localhost:9443/am/sample/pizzashack/v1/api/**	The endpoint that you add is automatically added as the production and sandbox endpoints.

6. Select the **Basic Info** tab under **Portal Configurations** and provide the following information and click **Save**.

Field	Value	Description
Publisher Access Control	All	The ability to protect an API to be managed only by users with specific roles
Developer Portal Visibility	Public	Whether this API is visible to all or restricted to certain roles in the Developer Portal.
Thumbnail Image	Download a PizzaShack logo image and upload it. You can get this Logo here : Link	Icon to be displayed in the Developer Portal (can be jpeg, tiff, png format).
Description	PizzaShackAPI: Allows to manage pizza orders (create, update, retrieve orders). Add this by clicking on **Edit Description** button close to the Thumbnail image of the API	High level description of API functionality.
Tags	pizza, order, pizza-menu	One or more tags. Tags are used to group/search for APIs (Press **Enter** after each tag)

API Categories	No API Categories defined by default. Hence leave this field blank. The categories should be defined by an Admin user through the Admin Portal.	API categories allow API providers to categorize APIs that have similar attributes. When a categorized API gets published to the Developer Portal, its categories appear as clickable links to the API consumers. The API consumers can use the available API categories to quickly jump to a category of interest.
GitHub URL	You can leave this field blank	This GitHub URL will be available in the API overview page in developer portal
Slack URL	You can leave this field blank	This Slack Channel URL will be available in the API overview page in developer portal
Mark the API as third party	No	Indicates if an API is a third party API. You can use third party APIs to expose an externally published API through API Manager.
Make this the default version	No	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.

7. Select the **Runtime** tab under **API Configurations** and use the following information to configure them and click **Save**.

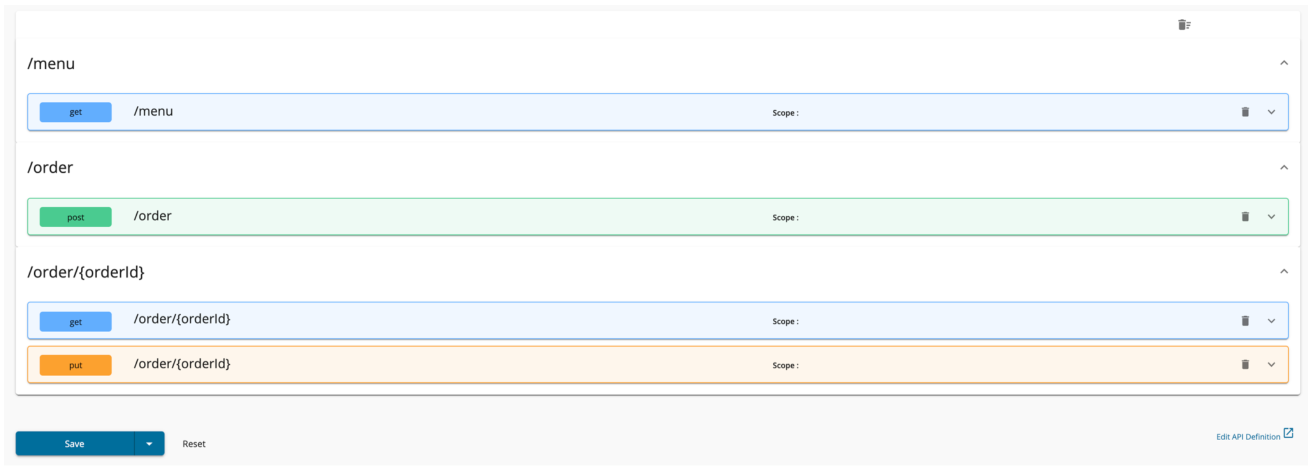
Field	Value	Description
Transport Level Security	HTTP/HTTPS/Mutual SSL	APIs can be exposed in HTTP and/or HTTPS transport: The transport protocol on which the API is exposed. Both HTTP and HTTPS transports are selected by default. If you want to limit API availability to only one transport (e.g., HTTPS), un-check the other transport. In mutual SSL the server validates the identity of the client so that both parties trust each other. If the Mutual SSL option is selected you can upload the certificates as necessary.

Application Level Security	OAuth2/Basic/ApiKey	APIs published on WSO2 API Gateway can be secured by OAuth 2.0. This is the default security available for all APIs. Basic authentication is a simple HTTP authentication scheme in which the request will contain an authorization header with a valid base64 encoded username and password. API Manager uses a self-contained JSON Web Token (JWT) as the API key. You can enable all these authentication mechanisms at the same time if required.
Key Manager Configuration	Allow all/Allow selected	Resident Key Manager is the default key manager available. You can plug in multiple key managers if required. **Allow selected** option gives you the capability to allow token generation from only a selected set of key managers
CORS Configuration	Disabled	Enable CORS for the API
Schema Validation	Disabled	Enables the request and response validation against the open API definition if selected.
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.

8. Select **Resources** tab under **API Configurations** and define the following resources and click **Save**.

For PizzaShackAPI, we will be defining 4 resources as defined below.

Resource URI Pattern	HTTP Verb
menu	GET
order	POST
order/{orderid}	GET
order/{orderid}	PUT


/menu

get /menu Scope :

/order

post /order Scope :

/order/{orderId}

get /order/{orderId} Scope :

put /order/{orderId} Scope :

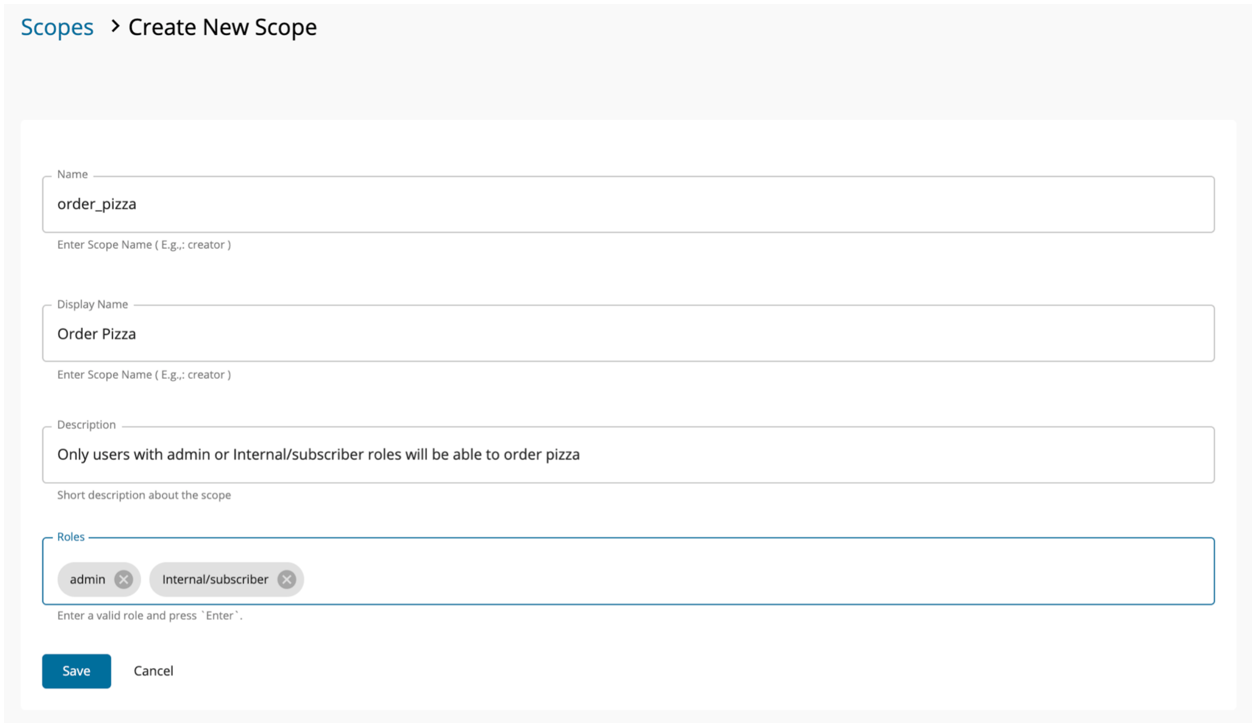
9. Select **Local Scopes** tab and add a scope with the information as explained below.

   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 only to the users with the admin 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, their OAuth2 bearer token cannot grant access to any API resource beyond its associated scopes.

Field	Description
Scope Name	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.
Roles	The user role(s) that are allowed to obtain a token against this scope. E.g., manager, employee. This field is optional.

To invoke an API protected by scopes, you need to get an access token via the Token API by directly calling it or the tokens with specified scopes need to be generated from the API Devportal.



Name: order_pizza

Enter Scope Name (E.g,.: creator)

Display Name: Order Pizza

Enter Scope Name (E.g,.: creator)

Description: Only users with admin or Internal/subscriber roles will be able to order pizza

Short description about the scope

Roles: admin, Internal/subscriber

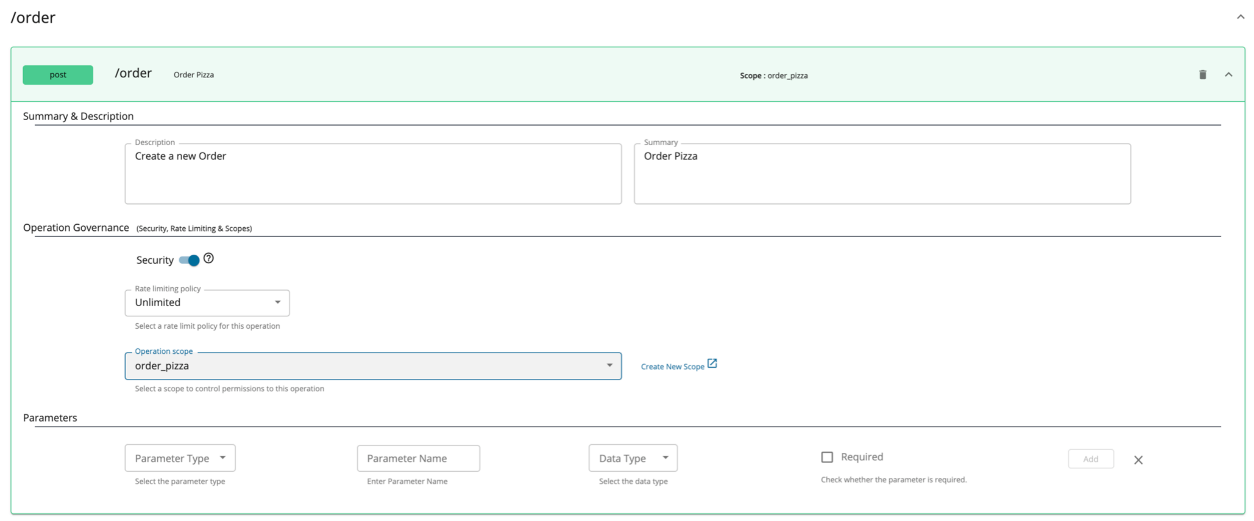
Enter a valid role and press `Enter`.

Field	Value
Name	order_pizza
Display Name	Order Pizza
Description	Only users with admin role and internal/subscriber role can order
Roles	admin, Internal/subscriber (Press enter after typing each role)

Since the scope is defined now, we need to assign that scope to the appropriate API resources.

10. Select **Resources** tab again and select **POST /order** resource and add the following information and click **Save**.

Field	Value
Summary	Order Pizza
Description	Create a new Order
Operation Scope	Select “order_pizza” from the dropdown


/order

post /order Order Pizza Scope : order_pizza

Summary & Description

Description: Create a new Order

Summary: Order Pizza

Operation Governance (Security, Rate Limiting & Scopes)

Security

Rate limiting policy: Unlimited

Select a rate limit policy for this operation

Operation scope: order_pizza

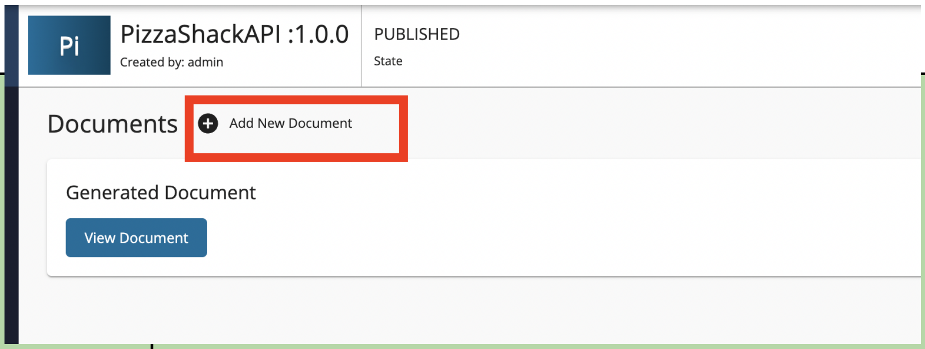
Select a scope to control permissions to this operation

Create New Scope

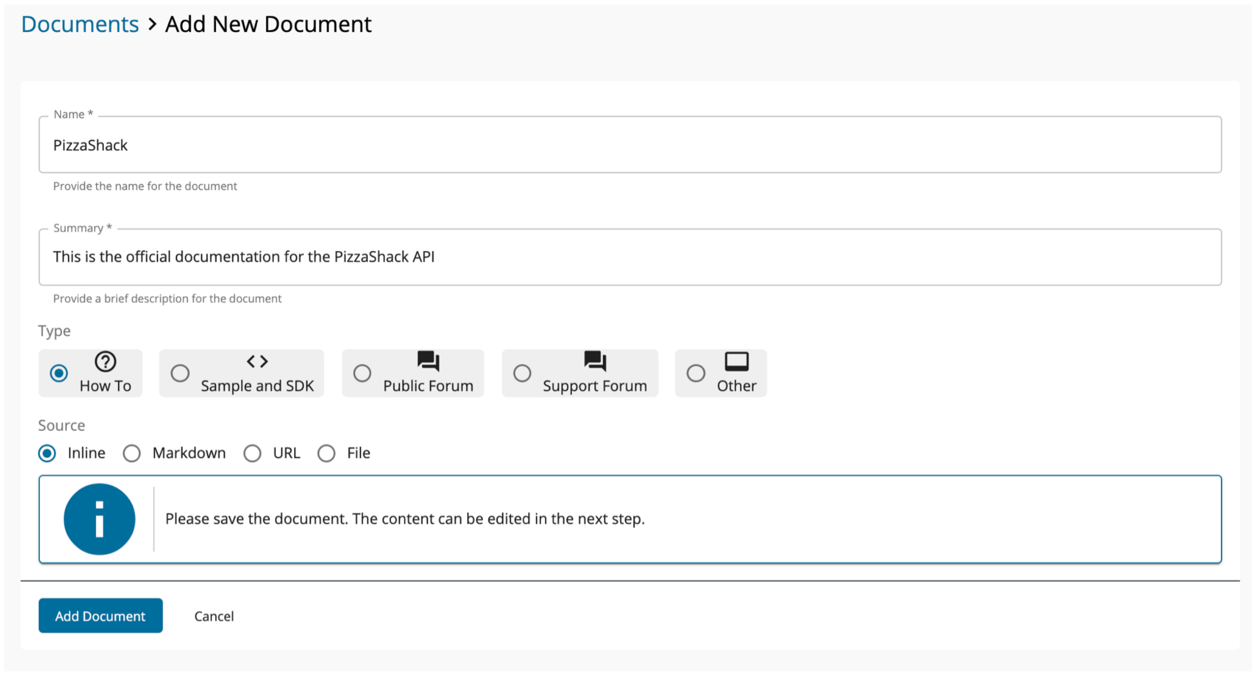
Parameters

Add Documentation

1. Select the **Documents** tab under **Portal Configurations**
2. Select **Add New Document** and enter the following details to add a document.



Field	**Value**
Name	PizzaShack
Summary	This is the official documentation for the PizzaShack API
Type	How To
Source	Inline


Documents > Add New Document

Name *
PizzaShack
Provide the name for the document

Summary *
This is the official documentation for the PizzaShack API
Provide a brief description for the document

Type
[✓ How To] Sample and SDK Public Forum Support Forum Other

Source
[✓ Inline] Markdown URL File

Please save the document. The content can be edited in the next step.

Several document types are available:

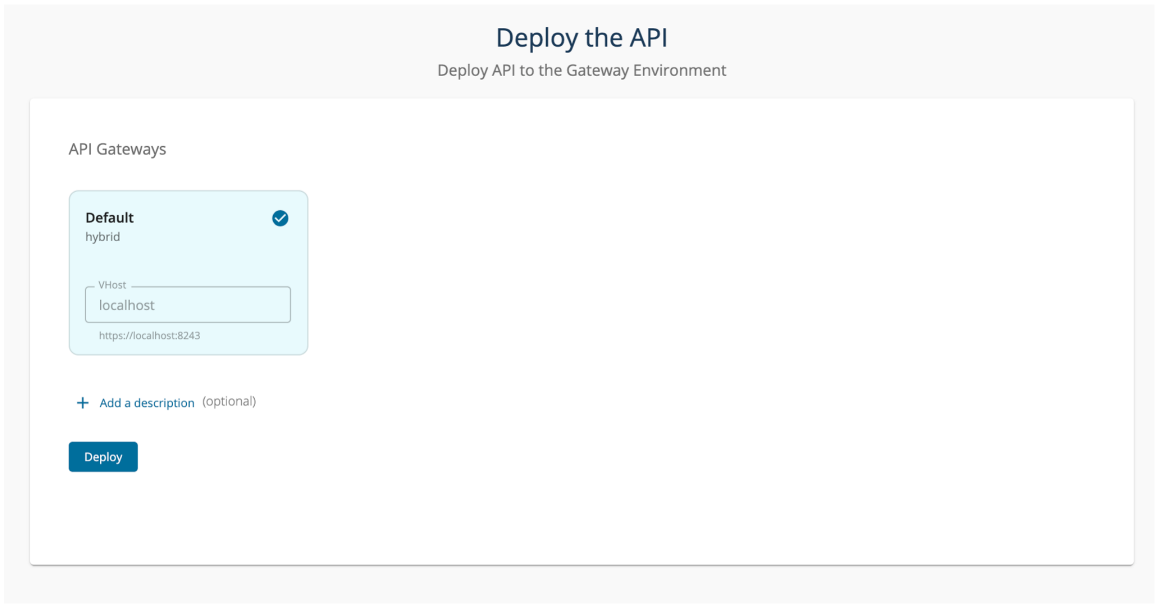
- How To
- Samples and SDK
- Public Forum
- Support Forum
- Other

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

Deploy the API

API Deploying is the process of making the API available for invocation via a gateway. You can deploy an API to a selected API Gateway environment via the Publisher Portal. To invoke an API, it needs to be PUBLISHED on the developer portal as well as DEPLOYED on a gateway environment. You need to create a revision of an API in order to deploy it.

Select **Deployments** tab under **Deploy**. Select the Gateway/s you need to deploy the API to and click on deploy. This will create the first revision and deploy it to the specified gateways.



Deploy the API

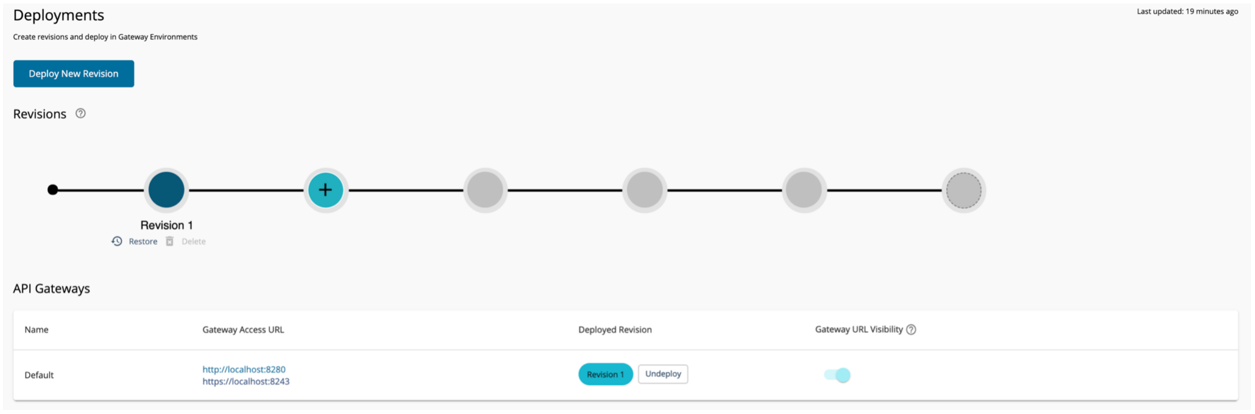
Deploy API to the Gateway Environment

API Gateways

Default ✓
hybrid

VHost: localhost

https://localhost:8243



Deployments

Create revisions and deploy in Gateway Environments

Last updated: 19 minutes ago

API Gateways

Name	Gateway Access URL	Deployed Revision	Gateway URL Visibility
Default	http://localhost:8280 https://localhost:8243	Revision 1 Undeploy	●

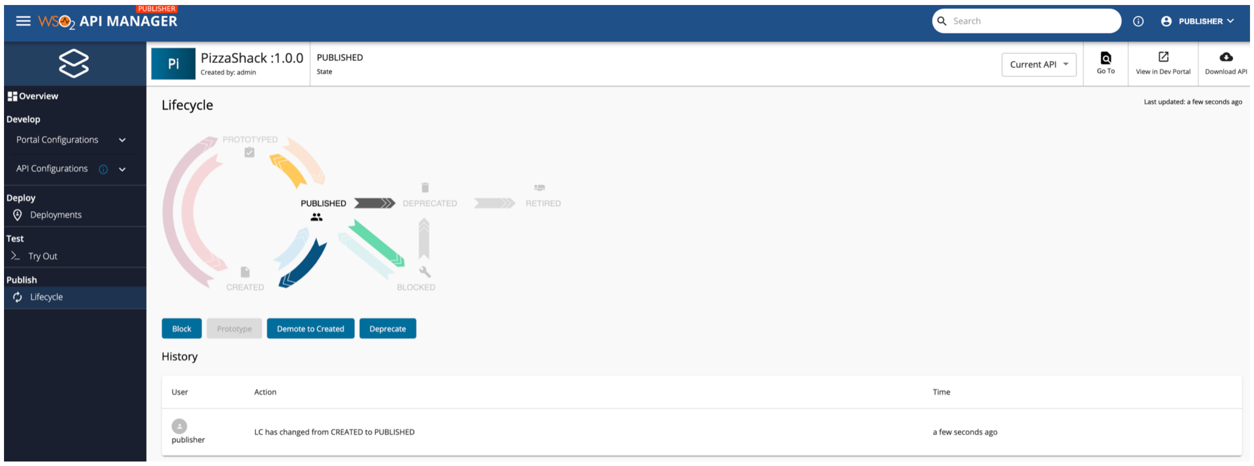
Publish the API

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 creator and login as publisher user.
2. Click on the API - You can see that an additional tab named **Lifecycle** is now available, allowing you to manage the API lifecycle.
3. To publish the API, select the **Publish** button.

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



User	Action	Time
publisher	LC has changed from CREATED to PUBLISHED	a few seconds ago

Expected Outcome

The PizzaShackAPI which manages pizza orders has been created and published and can be accessed through the API Developer Portal.