RedHat

Apiman

An Opensource API Mangement (on-premise)

Table of Contents

Introduction

Application Programming Interface (API)

The crux of any business or personnel to grow is “Communication” between them. For fruitful communication, there has to be a sender, a receiver, medium and most importantly, an **agreed** messaging Interface. This interface should be crystal clear and concise to be easily understood by both sides (sender and receiver).

API is an interface between two communicating software applications. It is the means (way) provided for multiple software applications to talk to each other, handshake and collaborate. Together these applications can provide more efficient output in less time by **re-using** one another’s APIs (**methods**).

The handshaking has to follow the **rules** or **protocols** by engaging in a **contract**. The contract specifies pre-defined clear and concise messages that one application (receiver/consumer) will adhere to use the other (sender/provider).

With the revolution that is ‘the Internet’, communication through APIs has reached its peak. These are called Web Services. The most popular ones are the REST(ful) services. By the year 2020, it is forecasted that there will be 20 Billion IoT (Internet of Things) ‘Connected Devices’. The usage of APIs is also growing to increase exponentially. Within years **Open Source** Softwares have also gained popularity, which helps everyone to experiment and add on new functionalities to existing software for **free**. As these changes become stable, they can be then used by enterprises and overall IT business to grow. This is further scaled with the help of a cloud platform. With **Cloud** platform (Centralised repository on the Internet) one can access the APIs as service (Software as Service [SaaS]) from any of their devices across the globe.

API Management

When we talk about exponential growth in terms of number of APIs, its usage, and scalability, there arises the need for **API management**. The following reasons make it inevitable:

- Securing the confidential data by controlling access to APIs
- Creating a central platform with a common gateway for visibility of APIs
- Analysing the usage of APIs
- Charging the usage of popular APIs (monetization)
- Controlling the network traffic by limiting the total number of hits to popular APIs
- Creating centralised API creation, maintenance and configuring strategies for API developers
- Automating the tasks of managing APIs

APIMan (Redhat API Management System)

Apiman will provide a one-stop solution for this full life cycle of API management for free as its an **Open Source** API Management Software.

Getting started with APIMan

Following chapter will help to the Install, configure and get apiman running for API providers and API consumers.

One can run apiman on any operating system that supports java software development.

Software Requirements

1) Java Development Kit (JDK) version 1.8 or higher
2) Maven build tool for building an example provided with apiman directory.
3) Git
4) JBoss Wildfly 10 J2EE application server

Downloading required software on Windows OS

1) JDK 8 or above
   Download link :
   https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

2) Maven
   Download link:
   http://maven.apache.org/download.cgi?Preferred=ftp://mirror.reverse.net/pub/apache/
   Install maven by unzipping the folder **apache-maven-3.6.1-bin.zip**
   Set System Environment Variable MAVEN_HOME= directory path till maven installation folder
   Example: D:\Study\Java\Java_software\apache-maven-3.6.1-bin\apache-maven-3.6.1

3) Git
   Download link:
   https://git-scm.com/downloads
   Install Git by running **Git-2.22.0-64-bit.exe**

4) WildFly 10

    1. Open link http://www.apiman.io/latest/

Following web page will appear

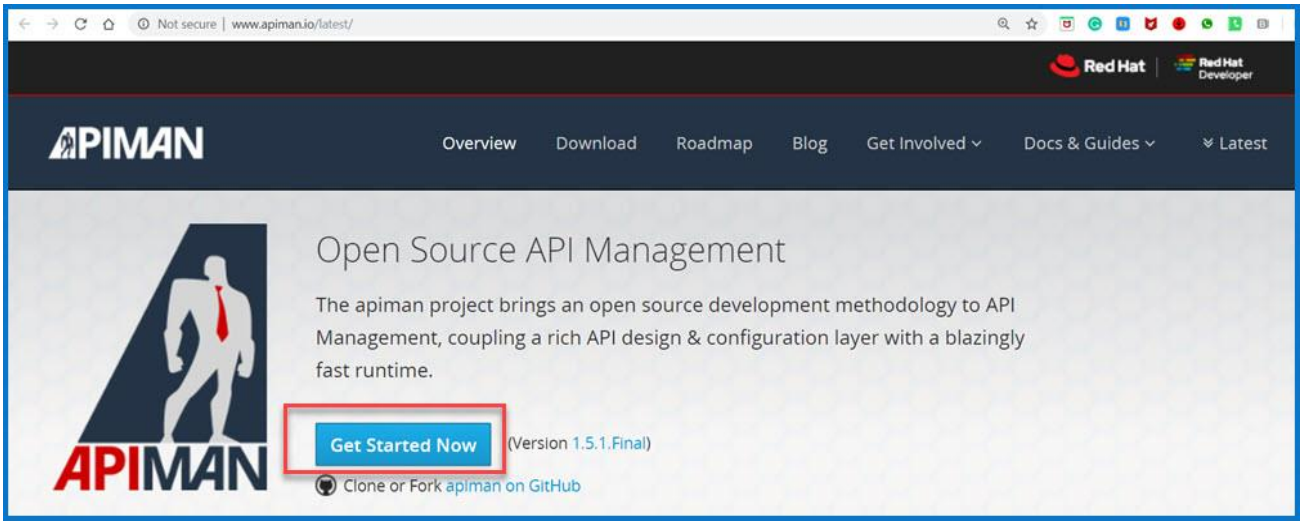


Figure 1: APIamn download link

2. Click on **Get Started Now** to display below the page

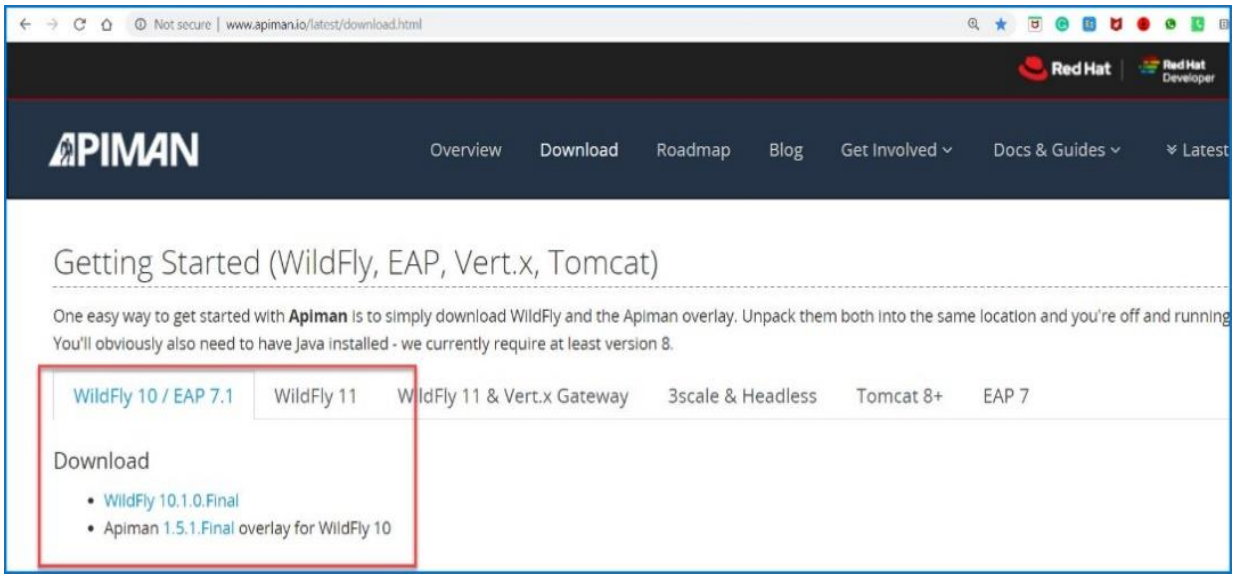


Figure 2:APIMAN download page

3. Click the respective tab to download a specific version of WildFly (Example WifldFly 10/EAP7.1)
4. Click on the second link **10.1.0 Final** specifying version of Wildfly out of **WildFly 10.1.0 Final** for the **Save As a** window to open.

   [Alternatively, one can click on first link **WildFly** and click the **Downloads** tab

   for the following web page to open

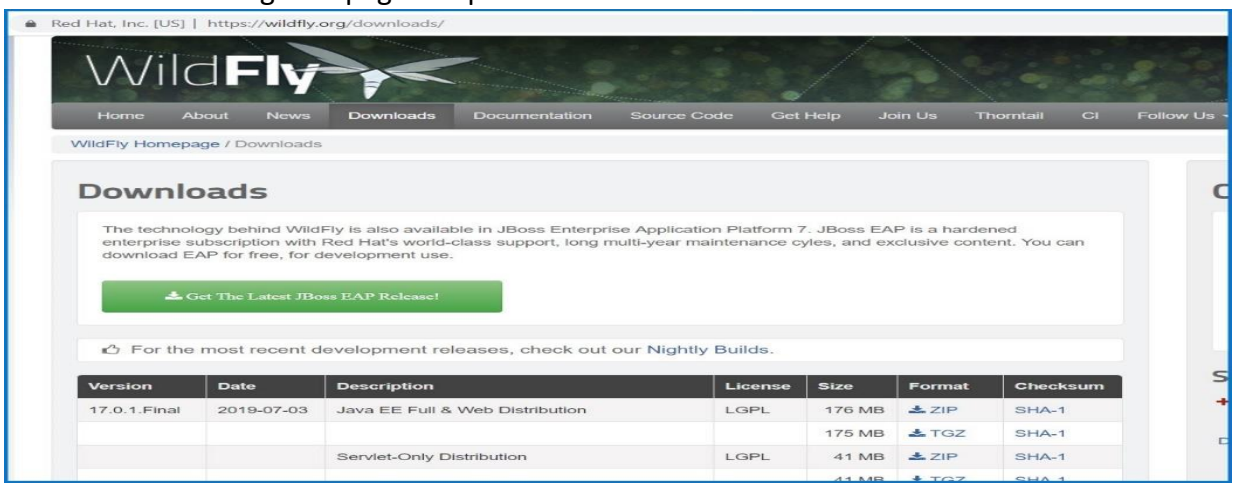


Figure 3: Table listing the wildflly version to download

Scroll down till Version 10.1.0 Final is reached and then click on Zip icon for **Java EE Full & Web Distribution.]**

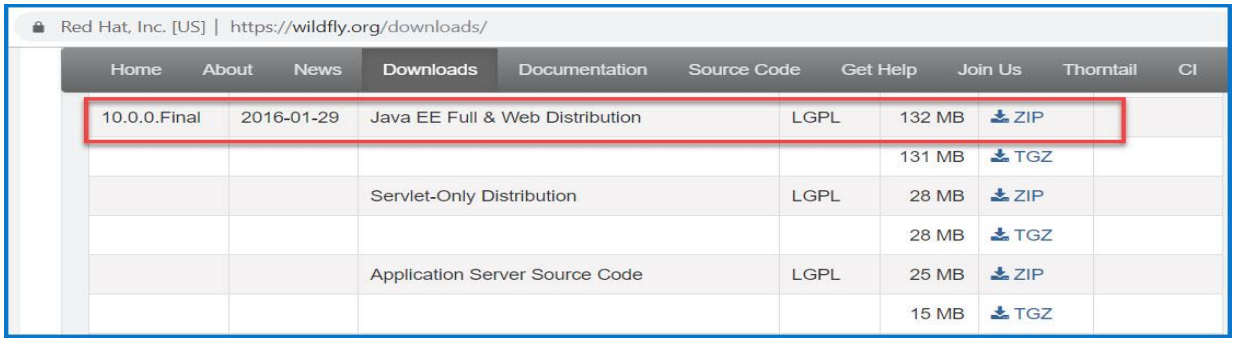


Figure 4: Wildfly 10 zip icon/link

5. Give the windows directory path to save the **wildfly-10.1.0.Final.zip** file**.**
6. Example : D:\Myapiman_softwares\
7. Click OK for the zip file to get downloaded.
8. You are back to the web page shown in figure 2 above

5) APIman

1. On web page is shown in figure 2 above, click on version link **1.5.1.Final** of **Apiman 1.5.1.Final overlay for WildFly 10** for **Save As** the window to open.
2. Give the same path as in step 5 to download **apiman-distro-wildfly10-1.5.1.Final-overlay.zip**[1]**.**
3. Verify the downloads in the directory shown in figure 5 below

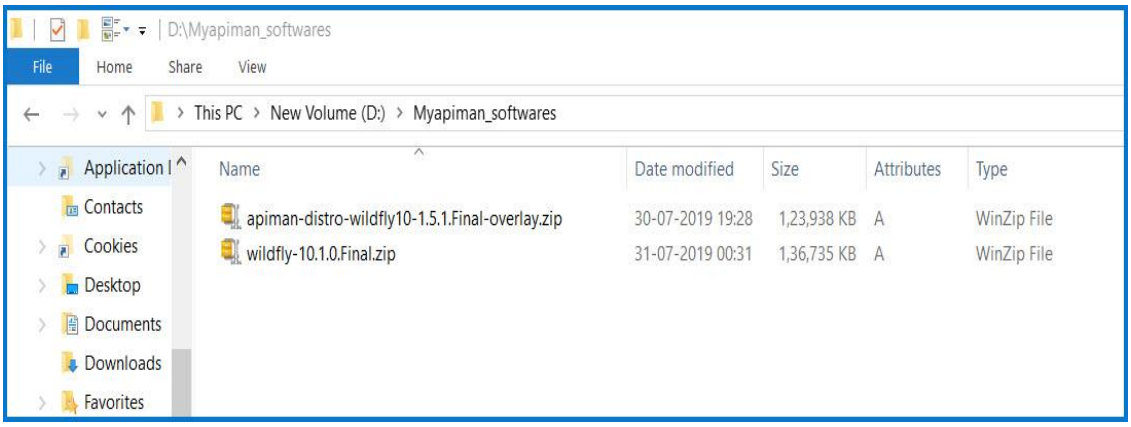


Figure 5: Apiman download directory

Note: The apiman zip (war[2]) file is an **overlay**[3] file

[1] apiman WildFly Overlay - In this distribution, apiman is packaged in a zip file that is installed over a JBoss WildFly (http://wildfly.org/) server.

[2] A .war file (Web application Archive) is a ZIP file containing all the files and folders required to launch a Java based website in a servlet container/web server/application server (WildFly).

[3] An overlay of a WAR file, is simply a Maven project that uses another project's WAR output as a dependency, rather than a project's JAR. When the overlay project is built, the underlying project's WAR file is exploded and files in the overlay project added to it. If an overlay project has a file with the same path and name as a file in the underlying WAR it will replace it. Thus the overlay project is literally that, it is a set of files that are laid over the original WAR. Some files may be new, others will replace what was already in the WAR file.

Installing the required software

Installation on Windows Operating System

1) JDK 8

1. Install JDK by running **jdk-8u221-windows-x64.exe**
2. Set System Environment Variable JAVA_HOME=directory path till JDK installation folder Go to Start->Control Panel->System&Security-> System->Advance System settings(in left navigation bar)
3. Example: C:\Program Files\Java\jdk1.8.0_221
4. Add new entry in system path : %JAVA_HOME%/bin

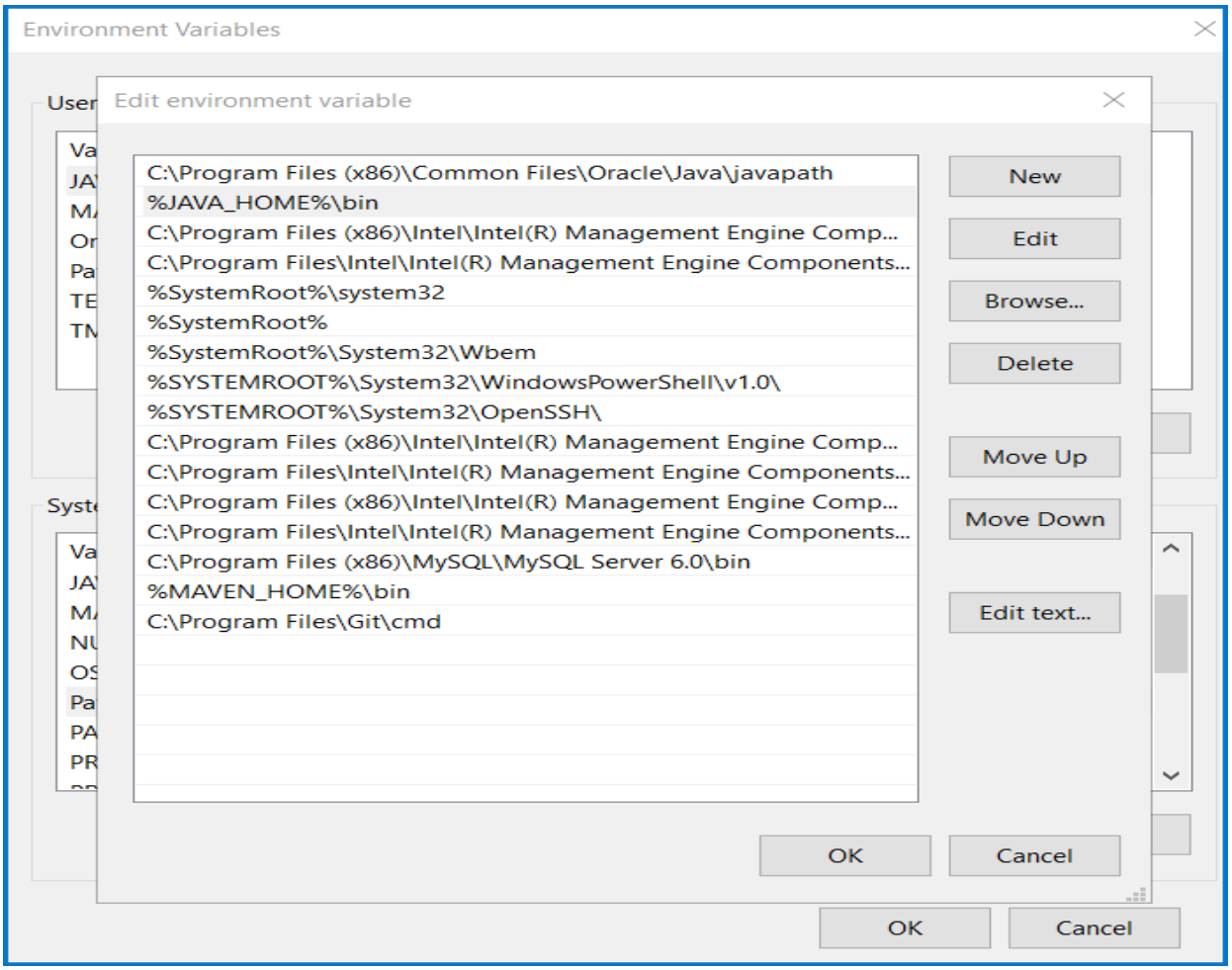


Figure 6: Windows system environment variable setting

2) Maven

1. Install maven by unzipping the folder **apache-maven-3.6.1-bin.zip**
2. Set System Environment Variable MAVEN_HOME= directory path till maven installation folder Refer JDK installation
3. Example: D:\Study\Java\Java_software\apache-maven-3.6.1-bin\apache-maven-3.6.1

3) WildFly 10

Unzip the wildfly-10.1.0.Final.zip in the directory**:** D:\Myapiman_softwares\
to get wildfly-10.1.0.Final folder created**.**
Open wildfly-10.1.0.Final folder to get the following directory structure

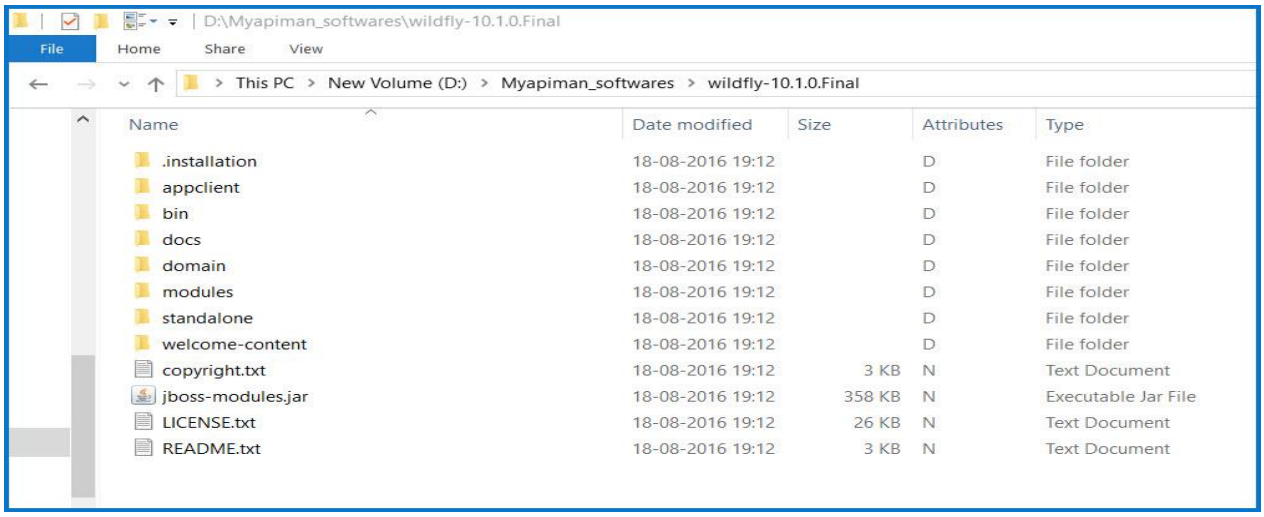


Figure 7: WildFly instllaiton directory structure

4) APiman

1. Copy **apiman-distro-wildfly10-1.5.1.Final-overlay.zip file** in **WildFly** directory created in step 3 above
2. Unzip **apiman-distro-wildfly10-1.5.1.Final-overlay.zip** directly in **wildfly-10.1.0.Final** directory
   Note : don't extract into folder apiman-distro-wildfly10-1.5.1.Final-overlay
3. While unzipping windows **might** ask to overwrite existing folders. If so, click **YES**[4]**.**

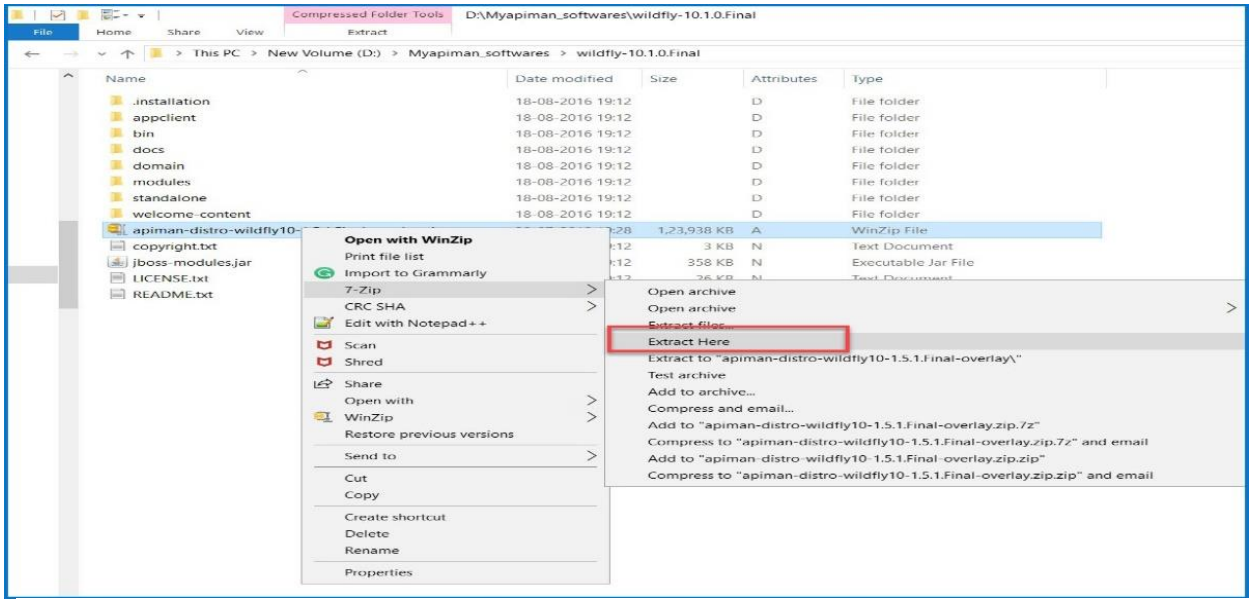


Figure 8: Unziping apiman overlay zip file

4. The following **wildfly-10.1.0.Final** directory structure is created .
   Note : A new folder **apiman** is created.

[4] The overlay zip file has few folders common and few new like **apiman** folder. We are supposed to overwrite common ones

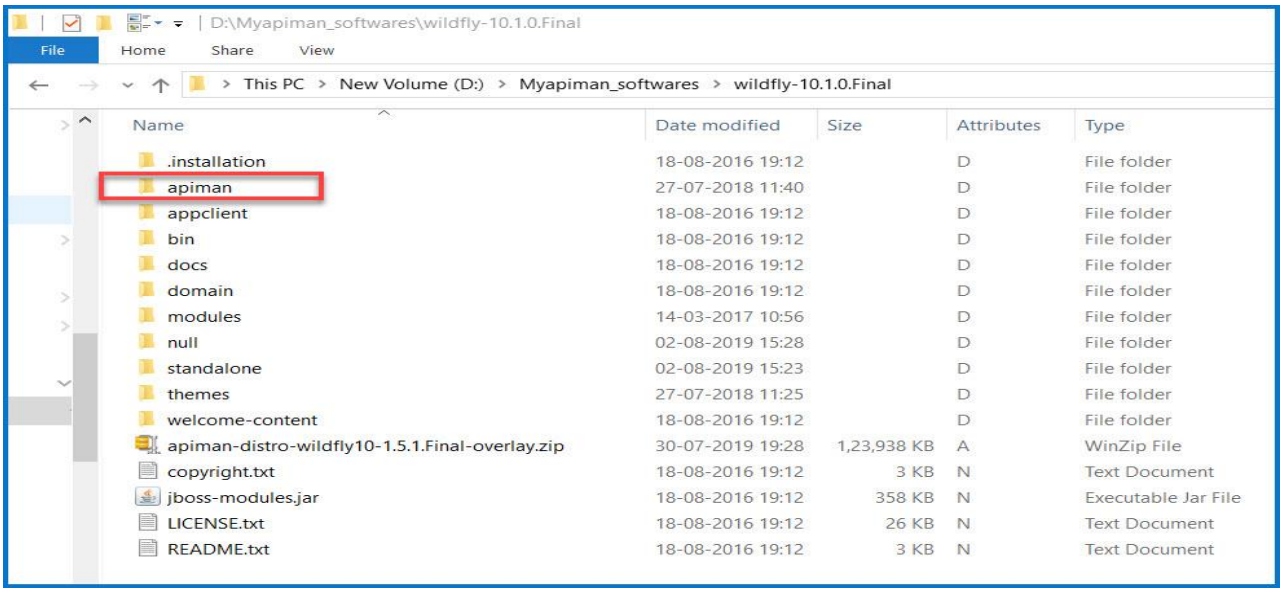


Figure 9: Apiman extracted directory structure in Wildfly folder

Installation on RedHat Linux Operating System

1) Got to http://www.apiman.io/

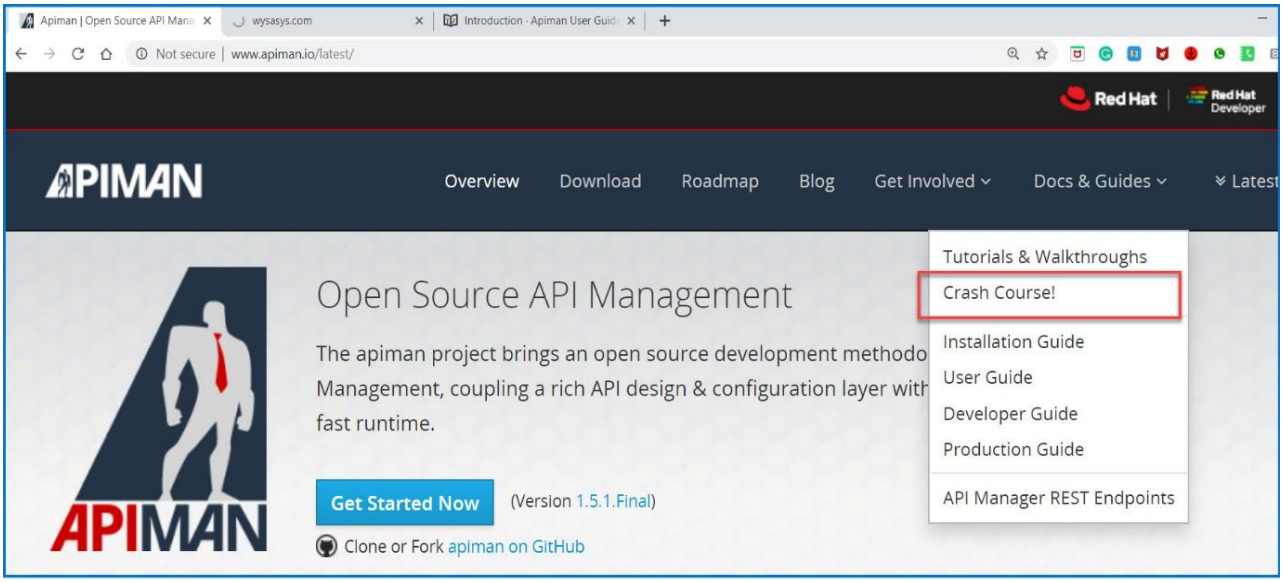


Figure 10: Apiman installation on linux guide

2) Click the **Docs & Guides** tab**.**
3) Click Crash Course
4) Goto Chapter 7. Section 6.r – Getting Up and Running with apiman in 10 minutes
5) Follow the steps given in this chapter to install apiman on Linux.

To get the APIman running on-premise, we have to deploy it on web server **wildfly**.

Following steps will

- Start Wildfly web server
- Deploy apiman on server

  With single command

Starting Wildfly Server on Windows OS with APIman config file

1) Open Windows Powershell from windows start menu

2) Go to **wildfly-10.1.0.Final** directory and then bin folder

```
>cd D:\Myapiman_softwares\wildfly-10.1.0.Final
>cd bin
```

3) Type following command:

```
>./standalone.ps1 -c standalone-apiman.xml
```

In above command **./standalone.ps1** will start Wildfly server and -c option takes parameters from **standalone-apiman.xml** configuration file inside the standalone directory to deploy apiman on the server.

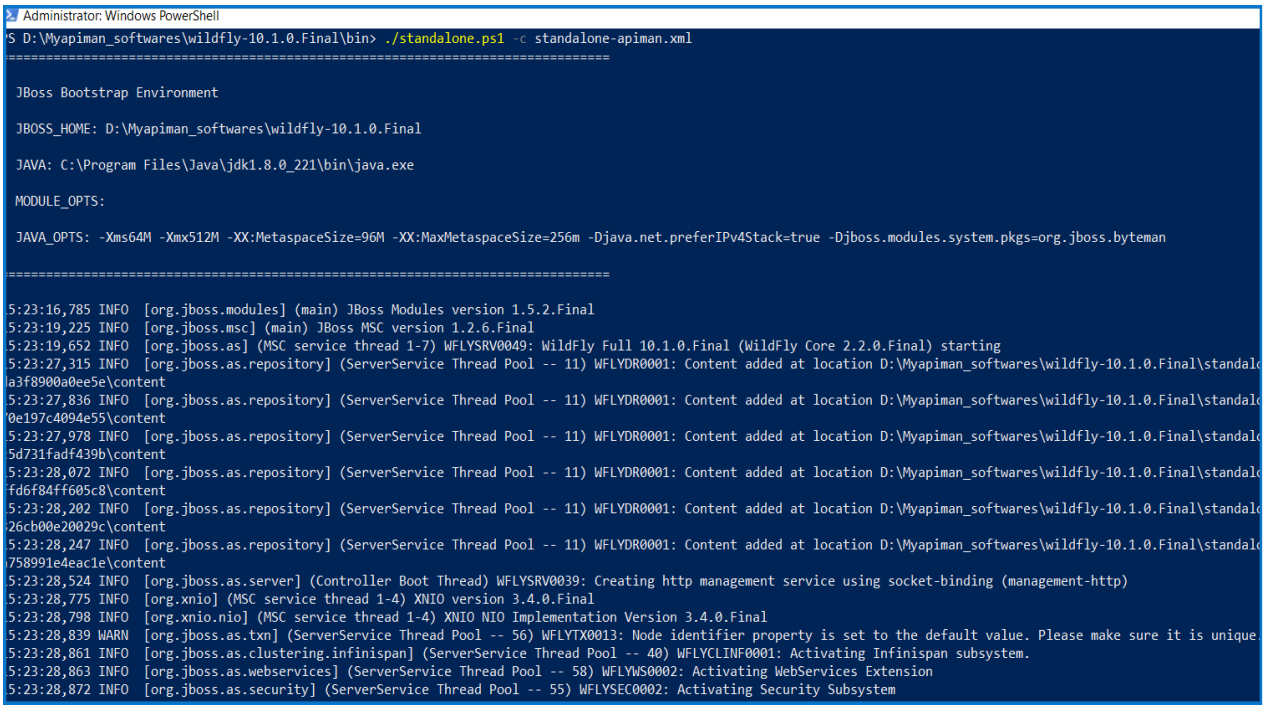


Figure 11: WildFLy server start

4) Wait till server starts. It will take a few seconds. Once sever starts following server started message will be shown on power shell. One can also notice all the war files that got deployed after the server being started.

Note : The **Deployed** messages are displayed for the apps being deployed on Fildfly server.
One of them is **apimanui** for launching apiman.

To manage the server, the admin URL is: http://127.0.0.1:9990/

```
5:51:28,893 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) Detected Elasticsearch transport tcp port : 19300
5:51:31,959 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-02T15:51:31,958][INFO ][o.e.c.s.ClusterService   ] [Hlau1TW]
eKXmNS_6tBcOzjroO8w}{127.0.0.1}{127.0.0.1:19300}, reason: zen-disco-elected-as-master ([0] nodes joined)[, ]
5:51:32,223 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-02T15:51:32,223][INFO ][o.e.g.GatewayService     ] [Hlau1TW]
5:51:32,317 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-02T15:51:32,317][INFO ][o.e.h.n.Netty4HttpServerTransport] [H
_addresses {127.0.0.1:19200}, {[::1]:19200}
5:51:32,317 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) Detected Elasticsearch http port : 19200
5:51:32,320 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-02T15:51:32,317][INFO ][o.e.n.Node               ] [Hlau1TW]
5:51:32,327 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (ServerService Thread Pool -- 79) ElasticSearch started...
5:51:35,287 INFO  [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-02T15:51:35,287][INFO ][o.e.c.r.a.AllocationService] [Hlau1TW
ELLOW] (reason: [shards started [[apiman_gateway][2]] ...]).
5:51:35,423 INFO  [stdout] (ServerService Thread Pool -- 79) ----------------------------
5:51:35,425 INFO  [stdout] (ServerService Thread Pool -- 79) apiman-es started!
5:51:35,426 INFO  [stdout] (ServerService Thread Pool -- 79) ----------------------------
5:51:35,430 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -- 79) WFLYUT0021: Registered web context: /apiman-es
5:51:35,468 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apimanui.war" (runtime-name : "apimanui.war")
5:51:35,468 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 60) WFLYSRV0010: Deployed "keycloak-server.war" (runtime-name : "keycloak-server.war")
5:51:35,469 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman.war" (runtime-name : "apiman.war")
5:51:35,479 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-gateway.war" (runtime-name : "apiman-gateway.war")
5:51:35,483 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-gateway-api.war" (runtime-name : "apiman-gateway-api.war")
5:51:35,483 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-es.war" (runtime-name : "apiman-es.war")
5:51:35,489 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-ds.xml" (runtime-name : "apiman-ds.xml")
5:51:35,601 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0060: Http management interface listening on http://127.0.0.1:9990/management
5:51:35,602 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin console listening on http://127.0.0.1:9990
5:51:35,611 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: WildFly Full 10.1.0.Final (WildFly Core 2.2.0.Final) started in 30125ms - Started 993 of 13
-demand)
```

Figure 12: Wildfly Server started with apiman deployed

Launching apiman on your browser

1) Open Web browser and goto sever welcome page http://localhost:8080

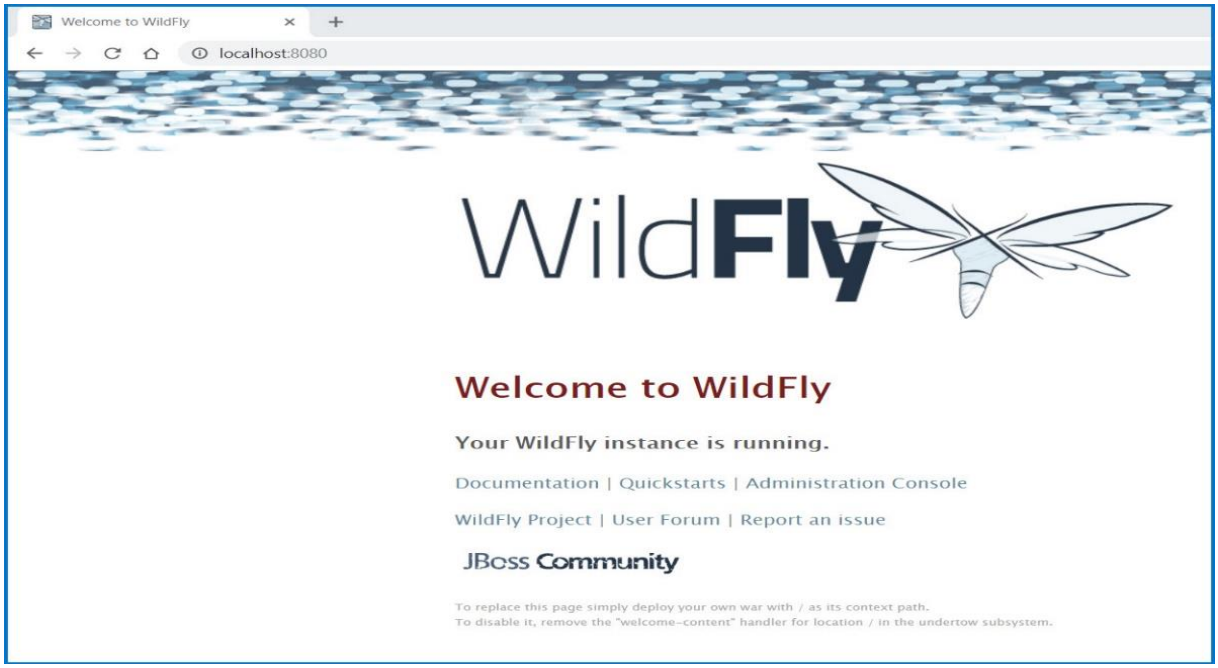


Figure 13: Wildfly Server home page

If one gets an error in opening the above page, it means an error had occurred in step 3. One has to troubleshoot.

2) Open **apiman UI** from a browser : http://localhost:8080/apimanui

The APIMAN Realm page is displayed

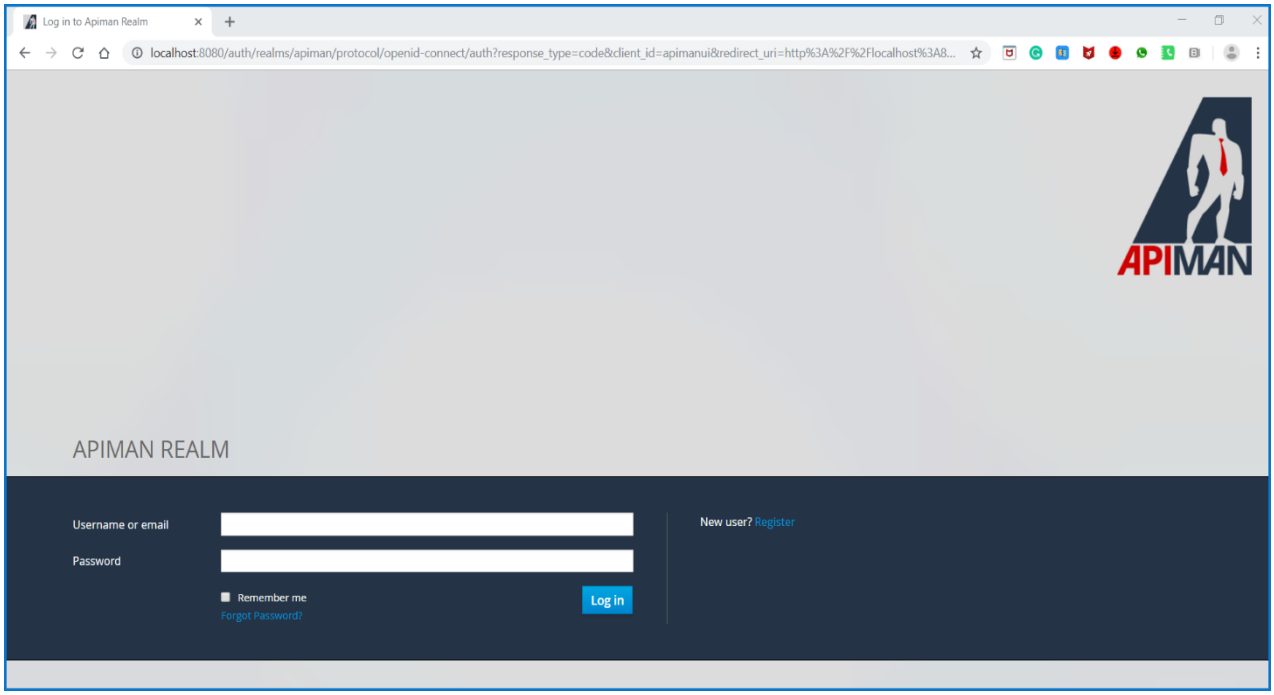


Figure 14: APiman Login(Realm) page

3) Create a new user by clicking **Register.**

4) Login as a user to get in apiman User Interface **Home** .

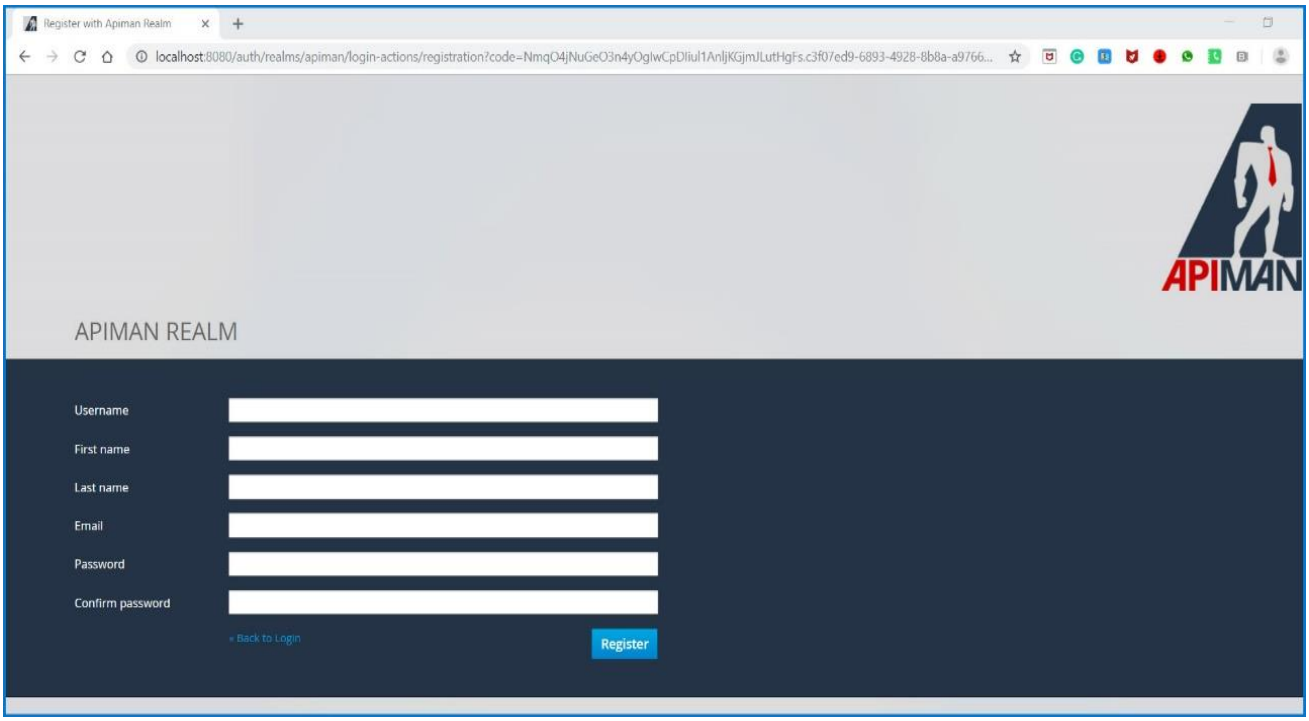


Figure 16: User resgistration page

APIMAN UI

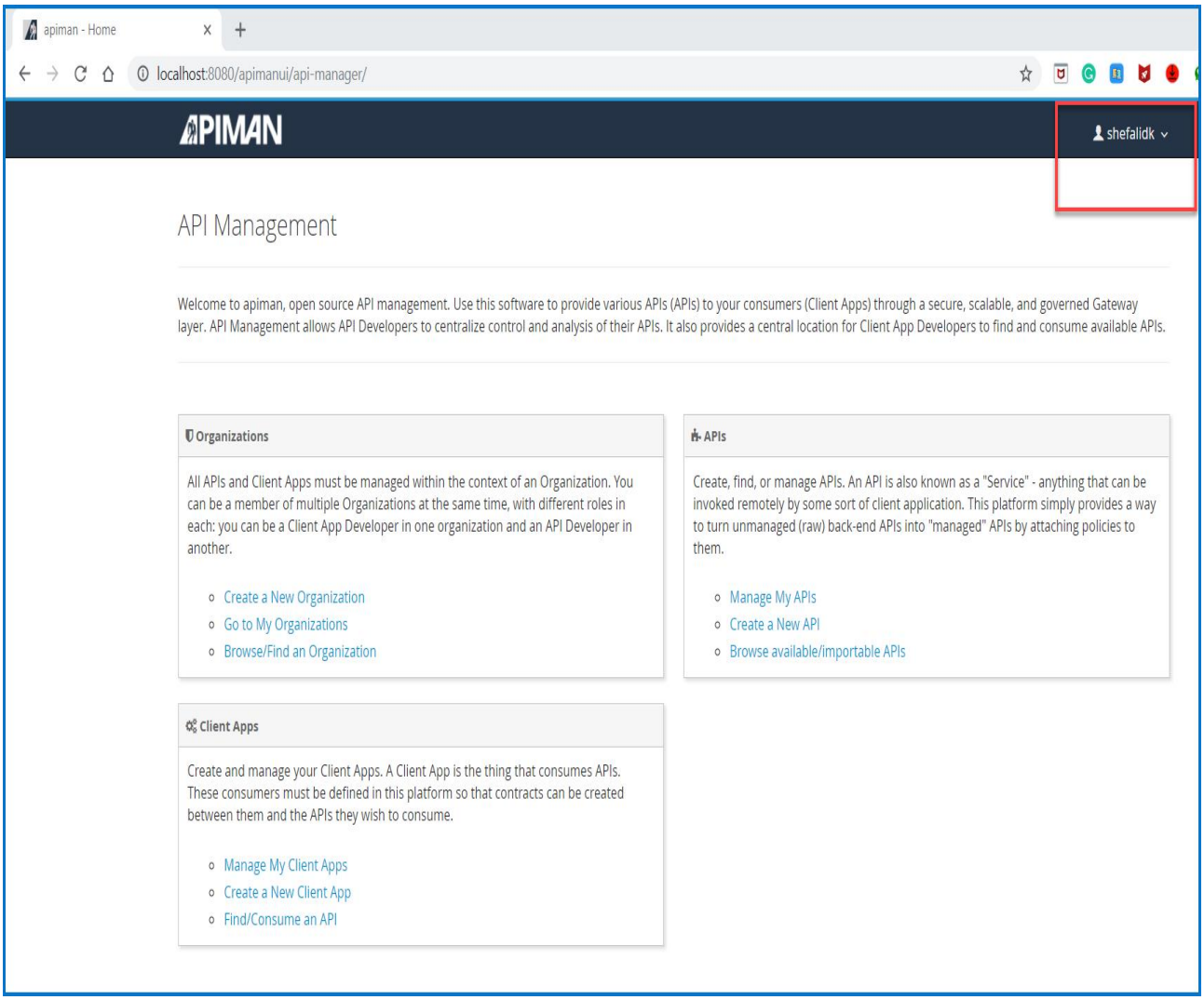
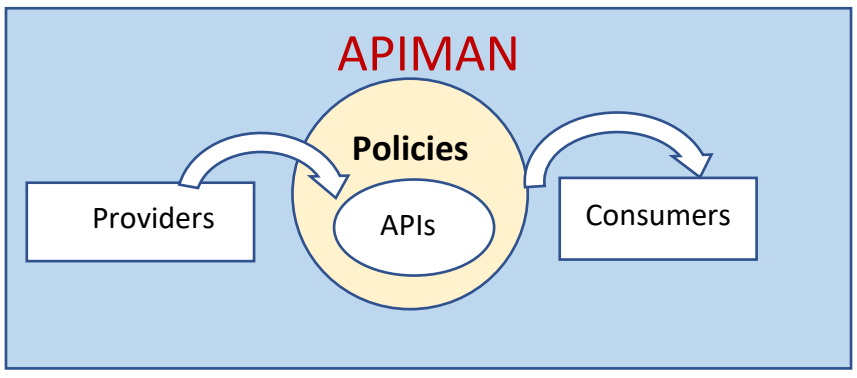


Figure 17: APIMan (API Manger UI) Home page

Working With Apiman

To manage API, one has to create an API. In API Management System the Four key objects are

1) **API**
2) **API Provider**
3) **API Consumer**
4) **Policies (Rules)**



As mentioned earlier through apiman one can

- Provide APIs
- Manage APIs configuring policies (rules)
- Discover/Search APIs
- Consume APIs
- Develope/Create API

Echo-Service QuickStart

The API (echo-service web service) performs the simple task of echoing back (in the **HTTP response)** the metadata contained in the HTTP request that it receives.

The following section will demonstrate the above functionalities of apiman with a sample **echo-service** API

Running the unmanaged (without using apiman) echo-service API on the browser

1) Download sample code

The source code for the example service is contained in a git repo (http://git-scm.com) hosted at github (https://github.com/apiman). To download a copy of the example service, navigate to the the directory in which you want to build the service and execute this git command:

```
>git clone git@github.com:apiman/apiman-quickstarts.git
```

Alternatively, you can go to https://github.com/apiman/apiman-quickstarts and download the source code by clicking **Clone or download** and **Download ZIP**

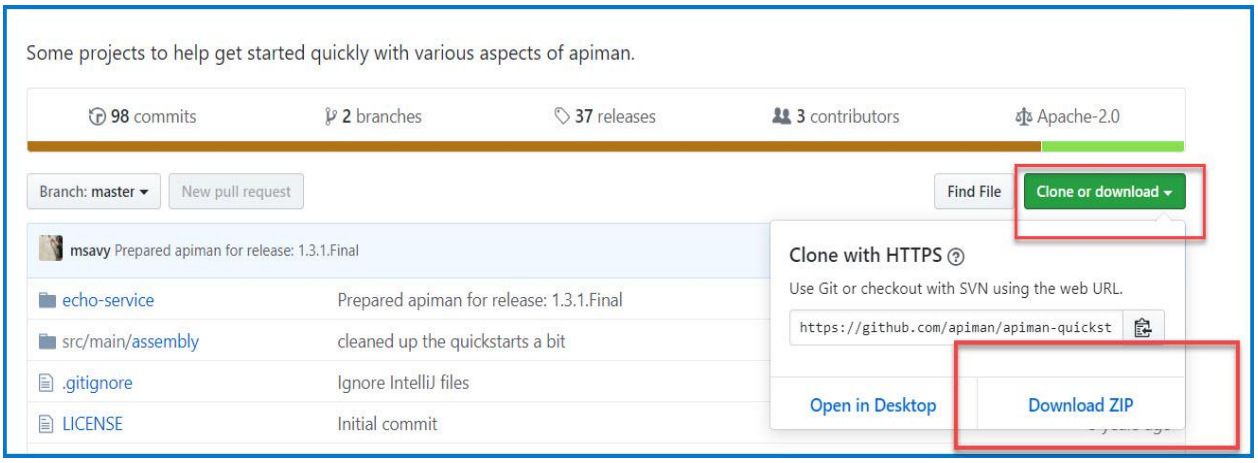


Figure 18: Git page for download echo-service

2) Unzip the file to get the **apiman-quickstarts-master** folder created.

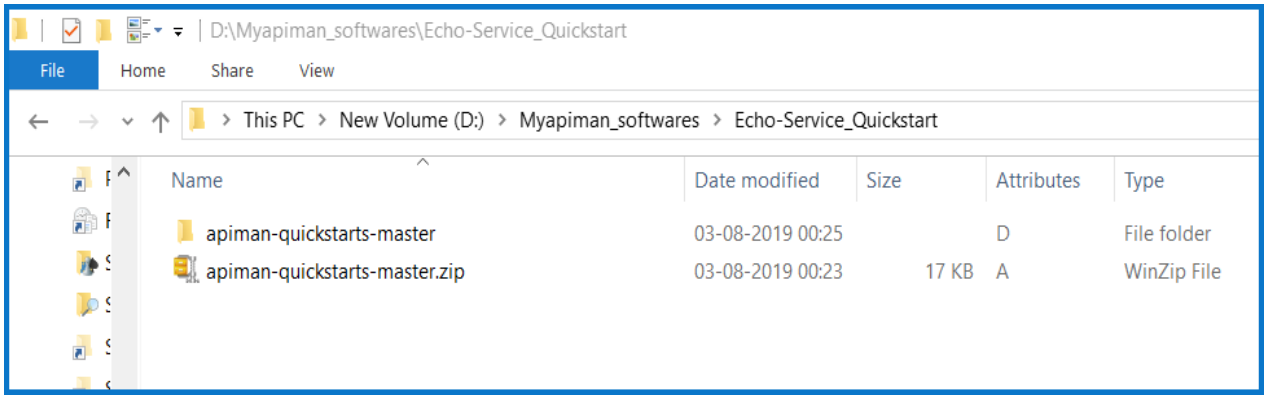


Figure 19: Unzipped apiman-quckstart folder

The directory structure of the folder is as shown in the figure.

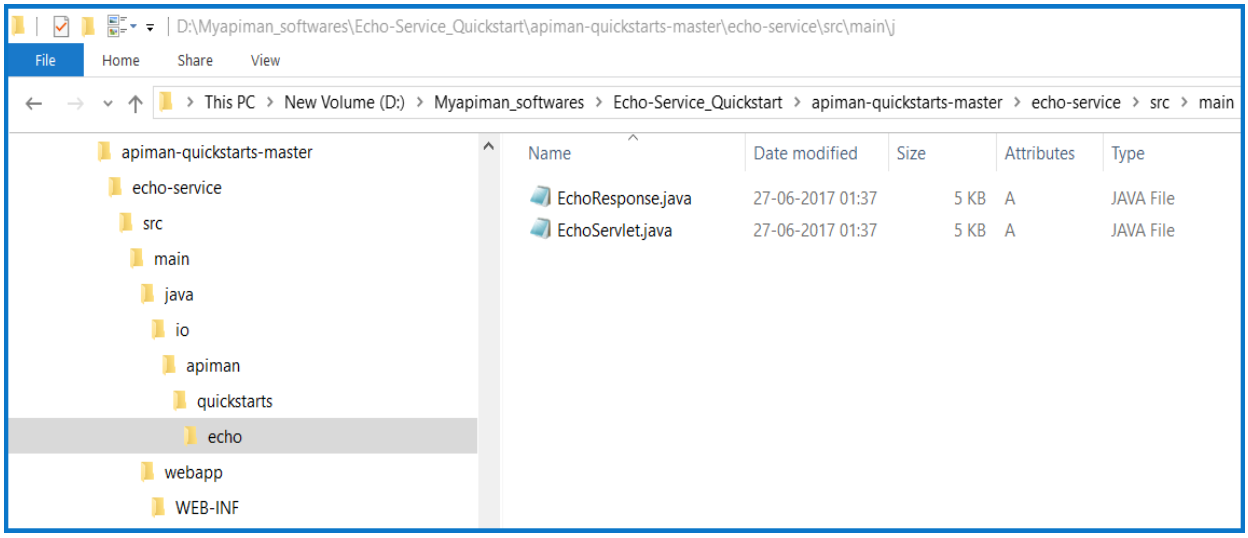


Figure 20: Directory structure of apiman-quickstart (echo-service) API

3) Creating a WAR (Web Archive) file using Widows PowerShell

Build (create war file) the echo-service as a web application using a maven build tool:

```
>cd  D:\Myapiman_softwares\Echo-Service_Quickstart\apiman-quickstarts-master\echo-service
>mvn package
```

A **BUILD SUCCESS** message is displayed, as shown below in figure 20 once **.war** file gets successfully created.

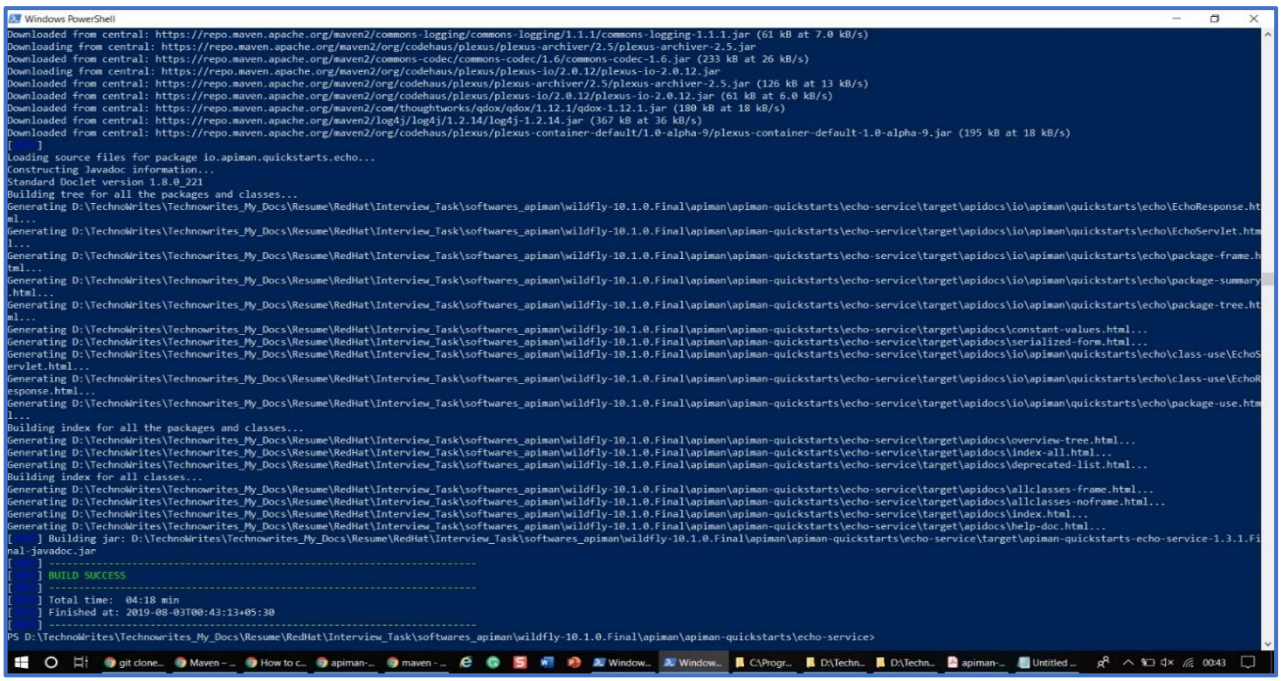


Figure 21: Echo-service build on maven to create war file

4) Deploying war file

Copy the **apiman-quickstarts-echo-service-1.3.1.Final.war** to Wildfly's deployments directory

Note: Wildfly server is already running. If not, first start the same

A Deployed message will be displayed for war file on Wildfly server which is already running on

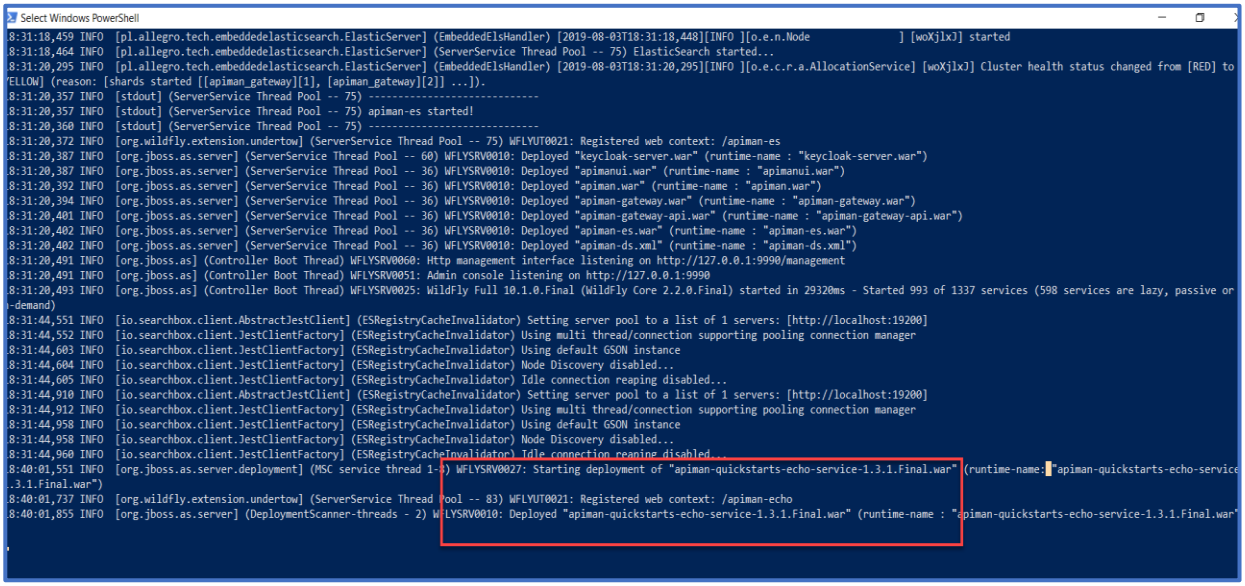


Figure 22: Wildfly server started with echo-service deployed

Windows PowerShell

One can see the URL for accessing service in the above figure as **/apiman-echo.**

Alternatively one can go to deployments folder to verify apiman-quickstarts-echo-service-1.3.1.Final.war file is deployed, as shown in figure 21 below.

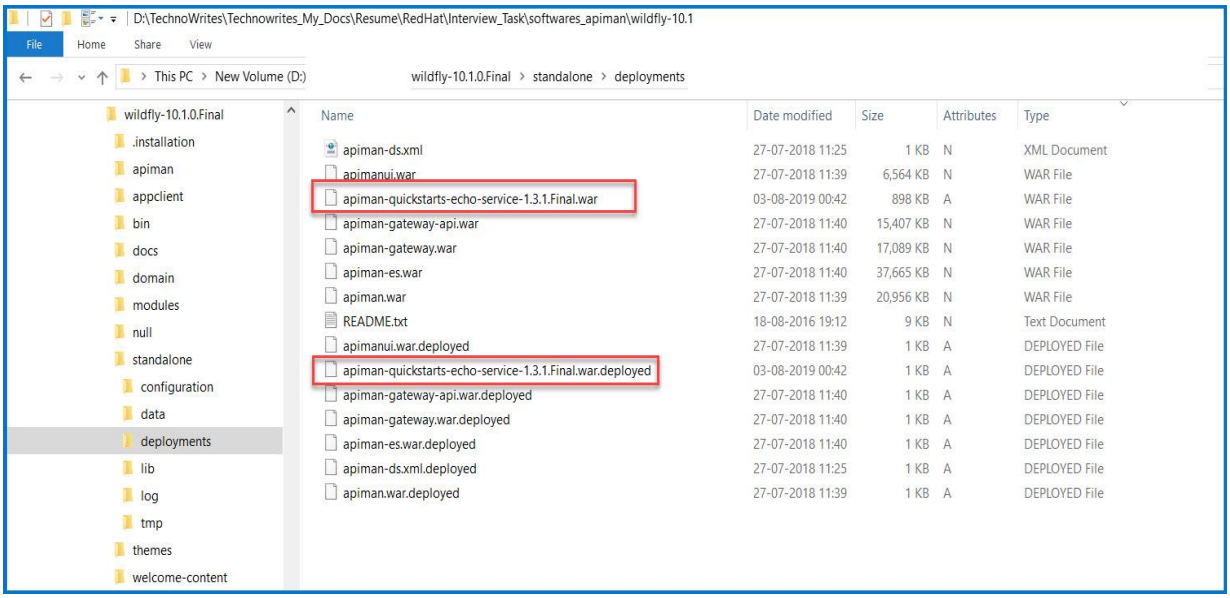


Figure 23: Echo-service deployed on Wildfly

5) Accessing the Echo-service API through a Client (browser)

The URL to access the echo-service through the browser is: http://localhost:8080/apiman-echo

Note: Remember, however, that this is the URL of the deployed example API if we access it directly. We'll refer to this as the "unmanaged API" as we are able to connect to the API directly, without going through the API Gateway. The URL to access the API through the API Gateway ("the managed API") at runtime will be different when policies to access the API are applied by **apiman**.

The following response (output) will be displayed on the web browser

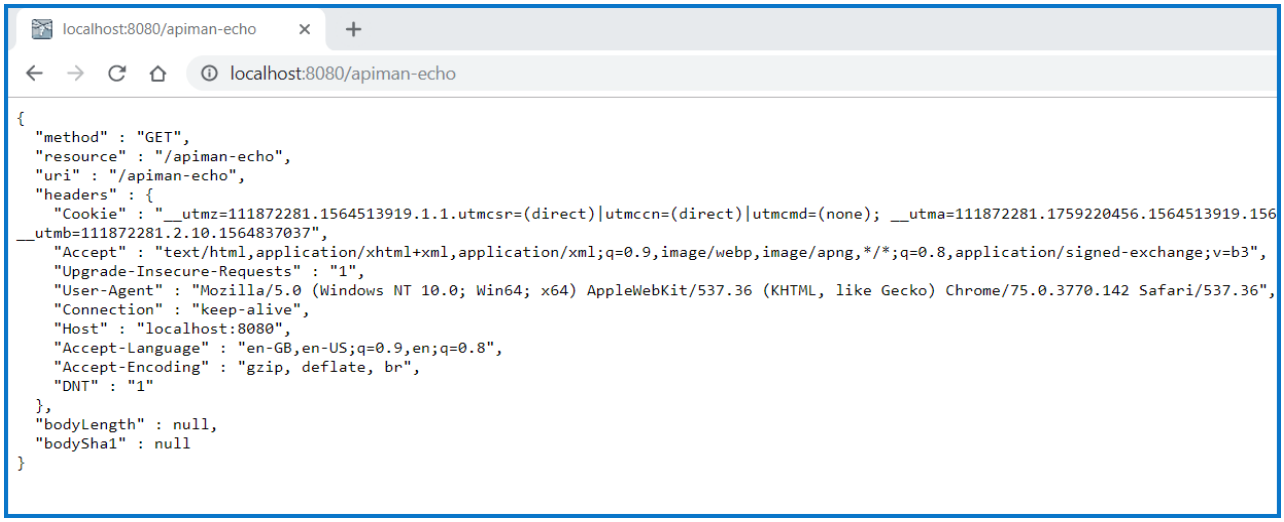


Figure 24: Echo-service Response without using API Gateway of apiman

Working with Managed echo-service API using APIMAN

To start using echo-service through APIMAN, one must follow the process as shown in the

process diagram below.

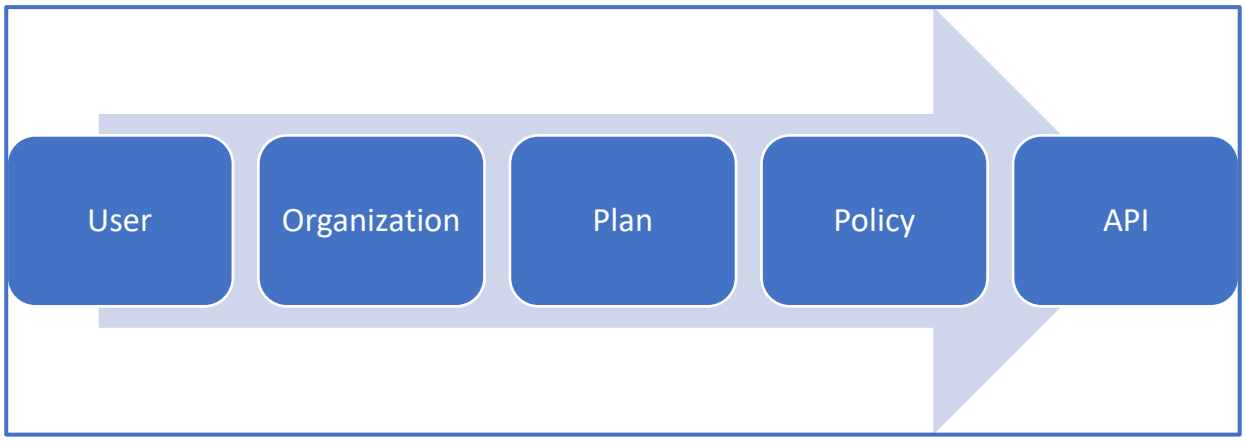


Figure 25: API Provider process flow

1) Creating two separate users for API provider and consumer

   (Refer to **Launching apiman on your browser** to access apiman login page and registering new users).

Working with API Provider

2) Login to apiman UI as **API provider user** from APIMAN Realm Page.
   (Refer APIMAN Login(Realm) Screen shown in Figure 16 above).

3) ON apiman UI page Click **Create New Organization** under **Organizations**

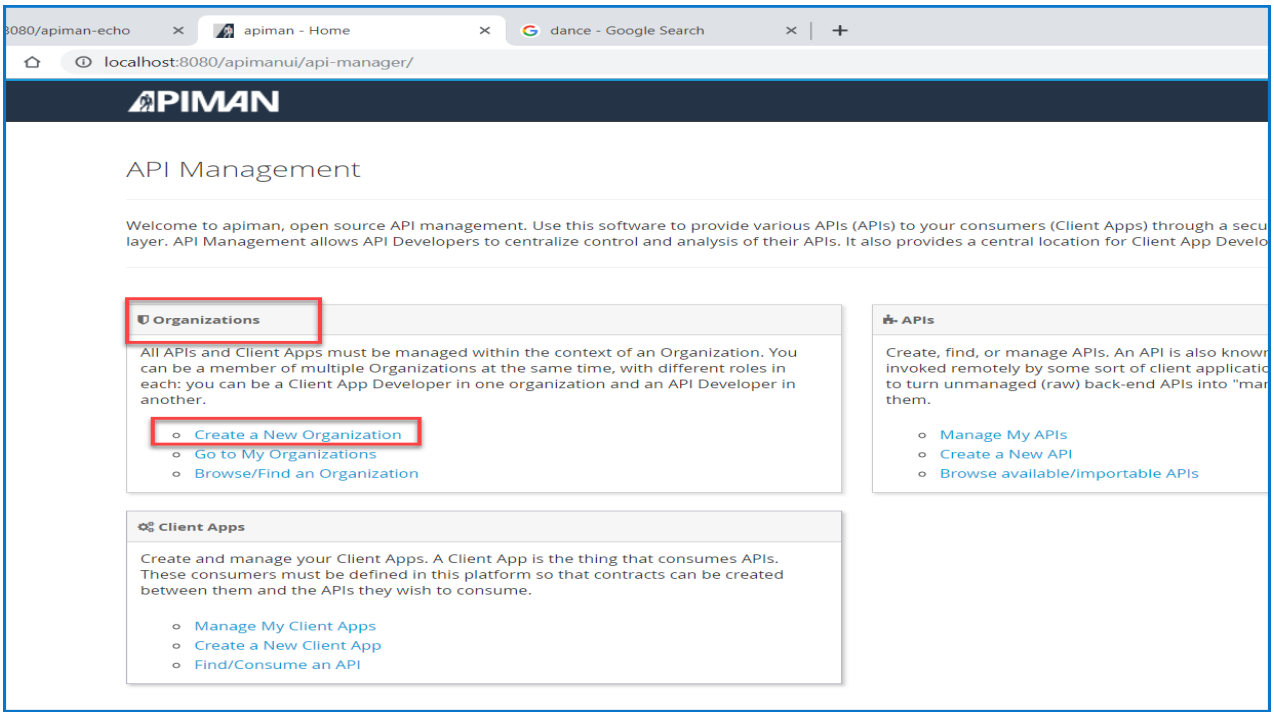


Figure 26: Create Organization link on APiman UI

Enter details of the organization

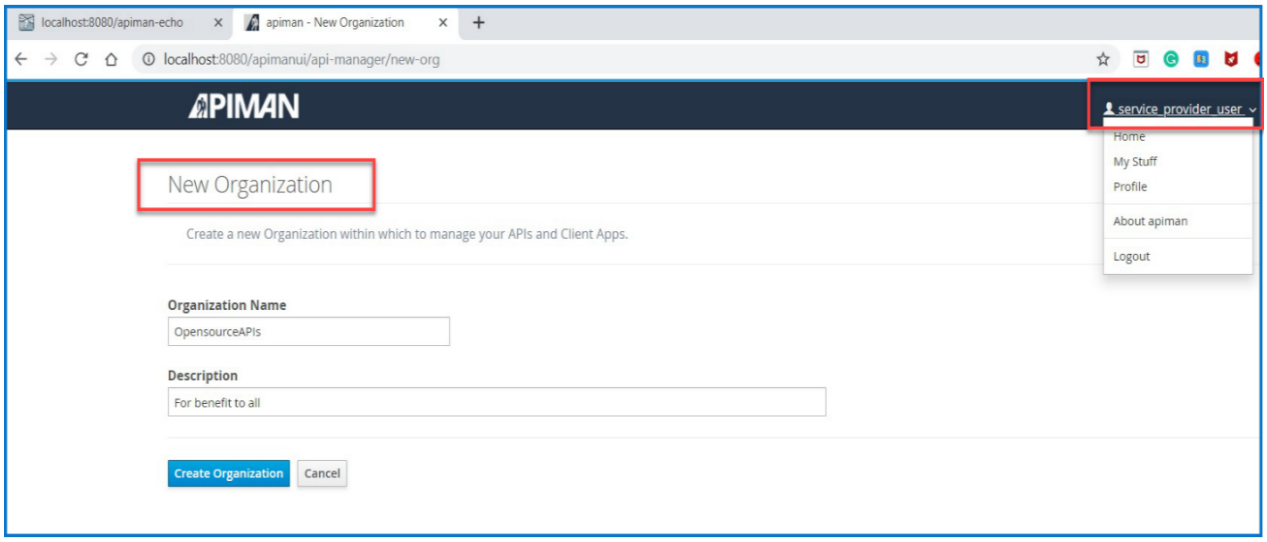


Figure 27: Create Organization for user

4) Creating Plans under the organization
A plan is a set of policies applied to an API.
In the organization, click on **Plans** tab and click **New Plan** as shown in figure 24 below

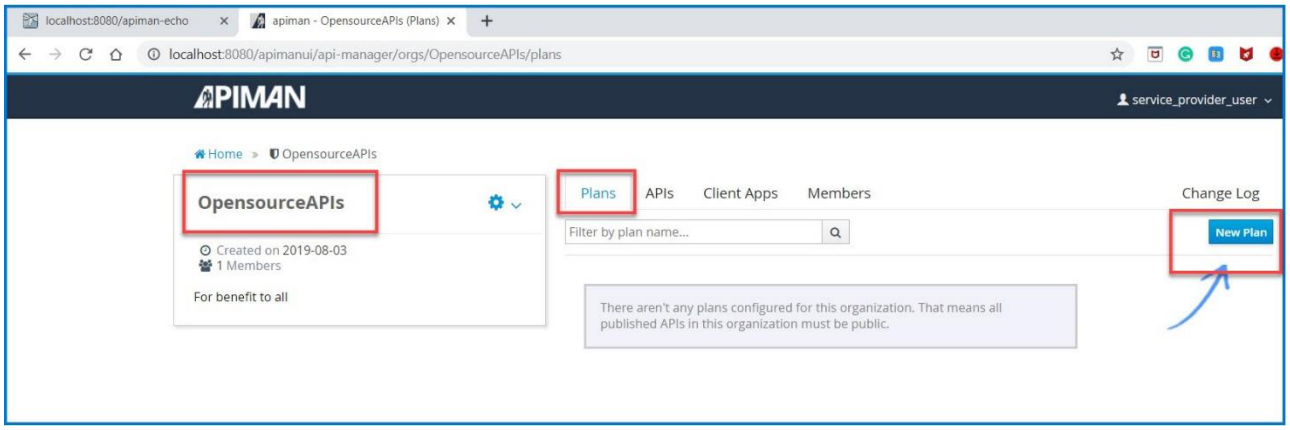


Figure 28: Create Plan

5) Adding policies to Plan

The core of APiman is Policy (Rules or functionalities). Every API in apiman can be configured with zero or more policies.

Whenever the API Gateway receives a request for an API (optionally on behalf of a specific Client App), it creates a chain of policies from those configured at the three levels and then applies that chain of policies to the request.

One can add policies to plan by clicking the **Policies** tab

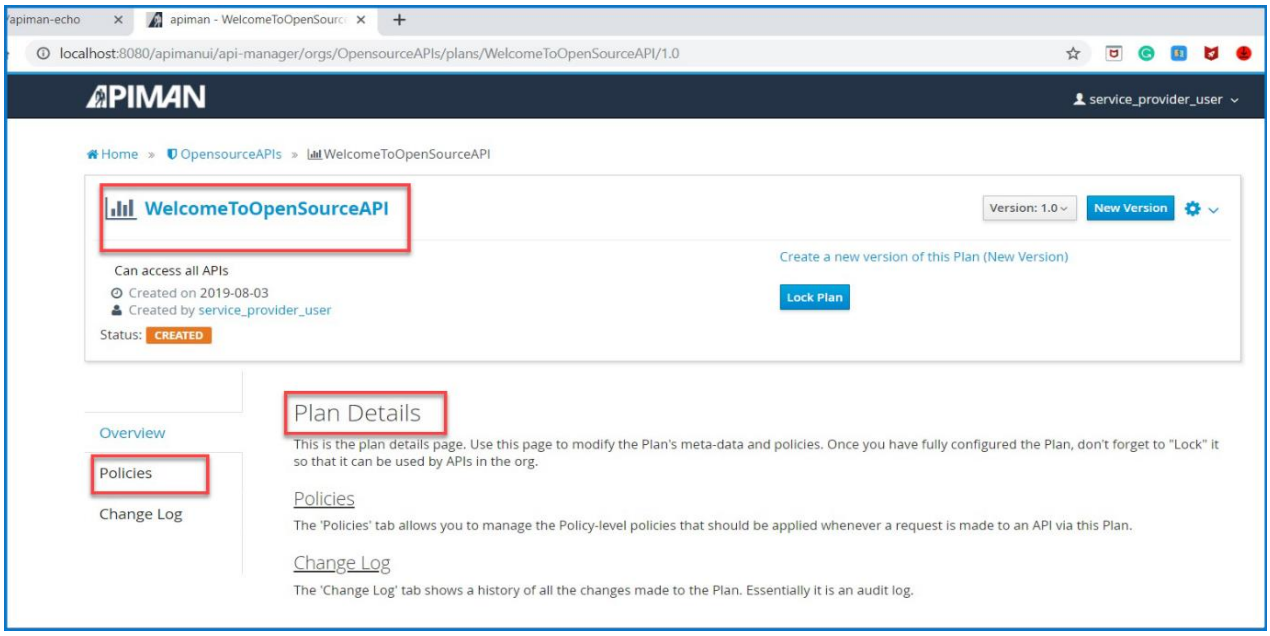


Figure 29: Plan with Policies tab

The policies page is displayed. You can add multiple policies for a single plan.

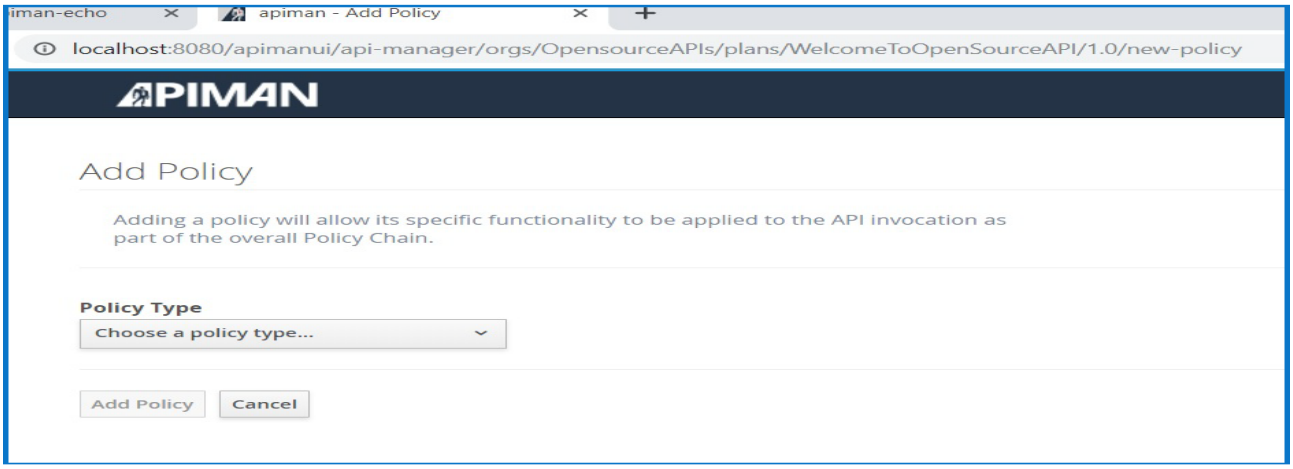


Figure 30: Add policy

In the figure shown above, select the specific **Policy Type** to be added to the plan.

The following policy types can be configured:

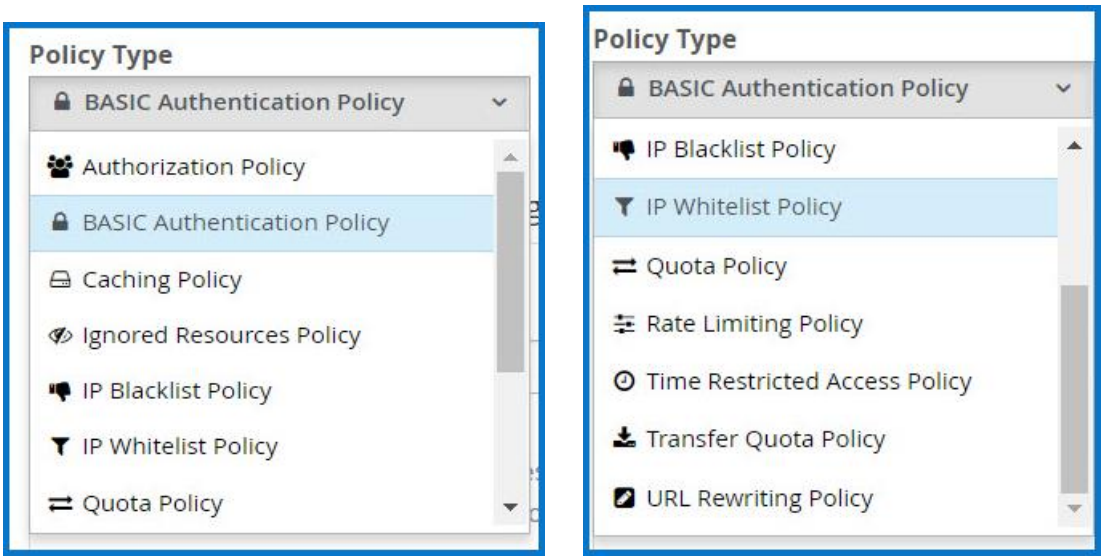


Figure 31: Policy Types

The following page is displayed for **Time Restricted Access** policy type

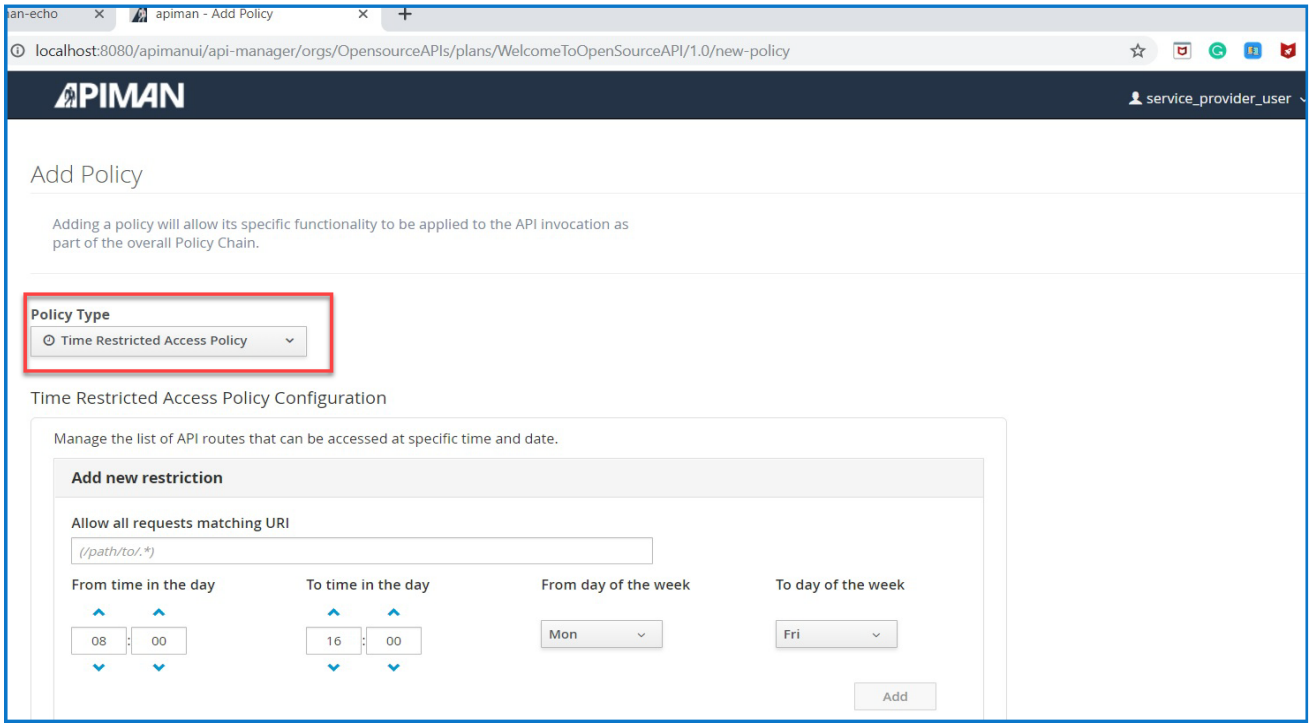


Figure 32: Time Restricted Policy

After adding multiple policies to plan, one has to Lock the plan by clicking the **Lock Plan** shown in figure 28 below. The plan **WelcomeToOpenSource** has two policies added**.**

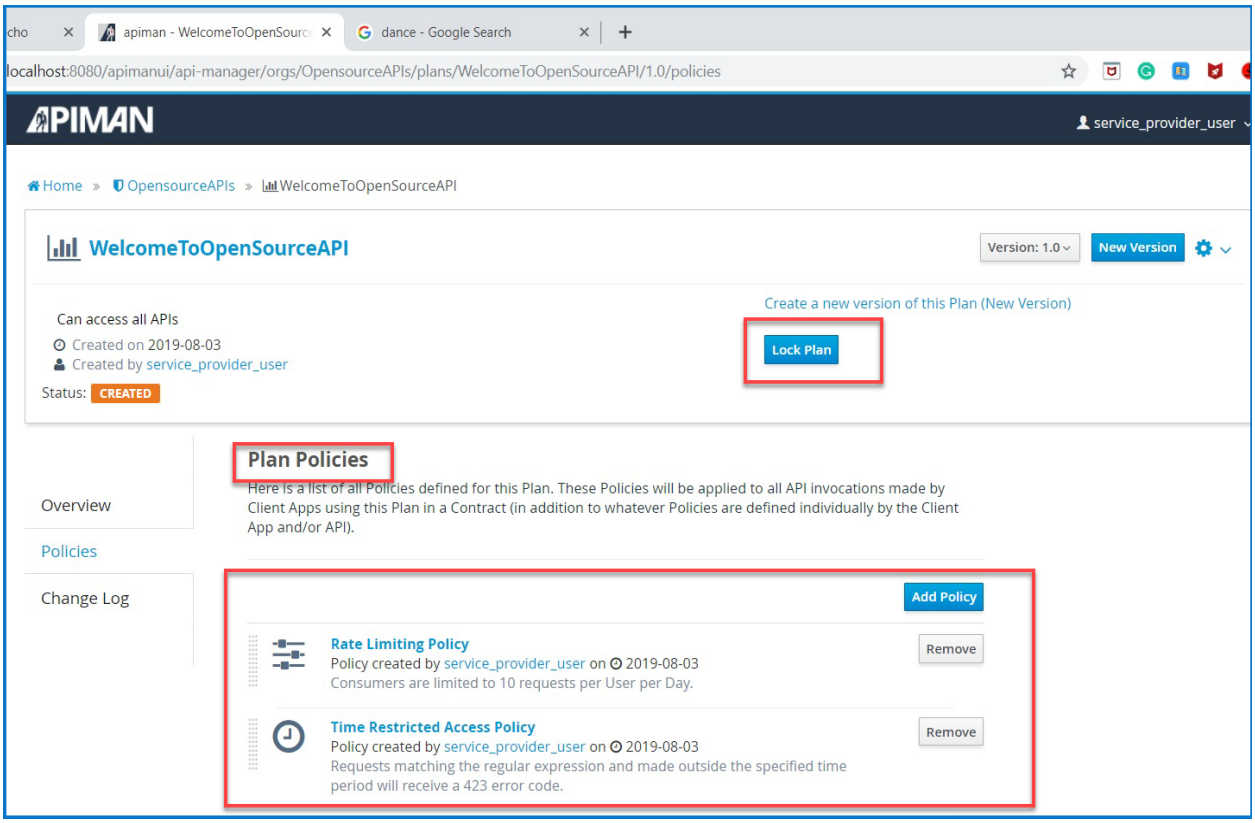


Figure 33: Locking the Plan

6) Define API

To use (consume) echo-service API by a consumer, the provider.has to define the same. In the organization, click on **APIs** tab and click **New API** as shown in figure 32 below

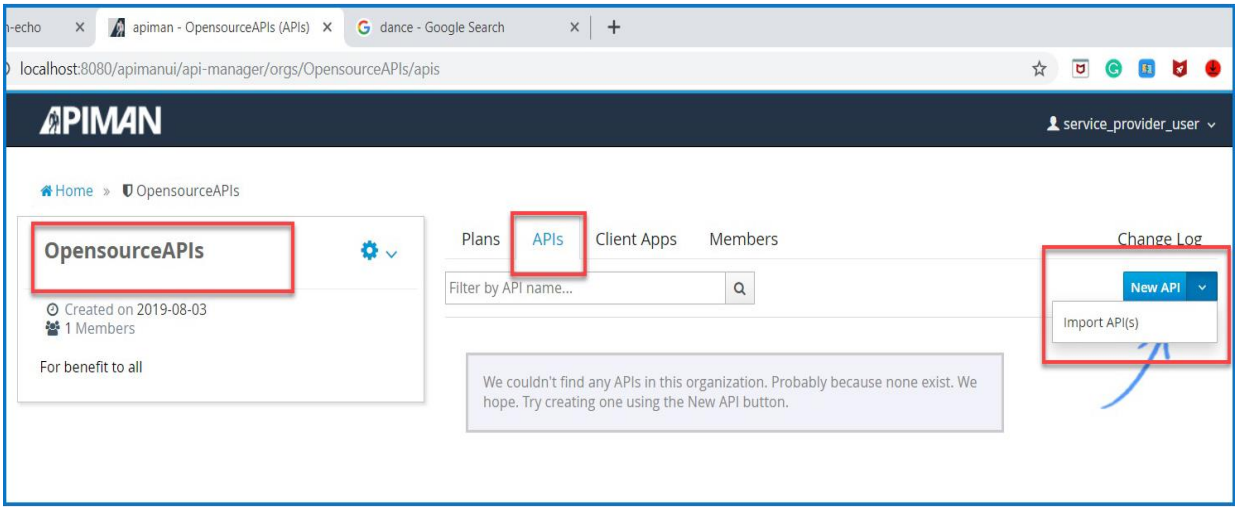


Figure 34: Define API inside Organization

Define the echo-service API here by naming the same.

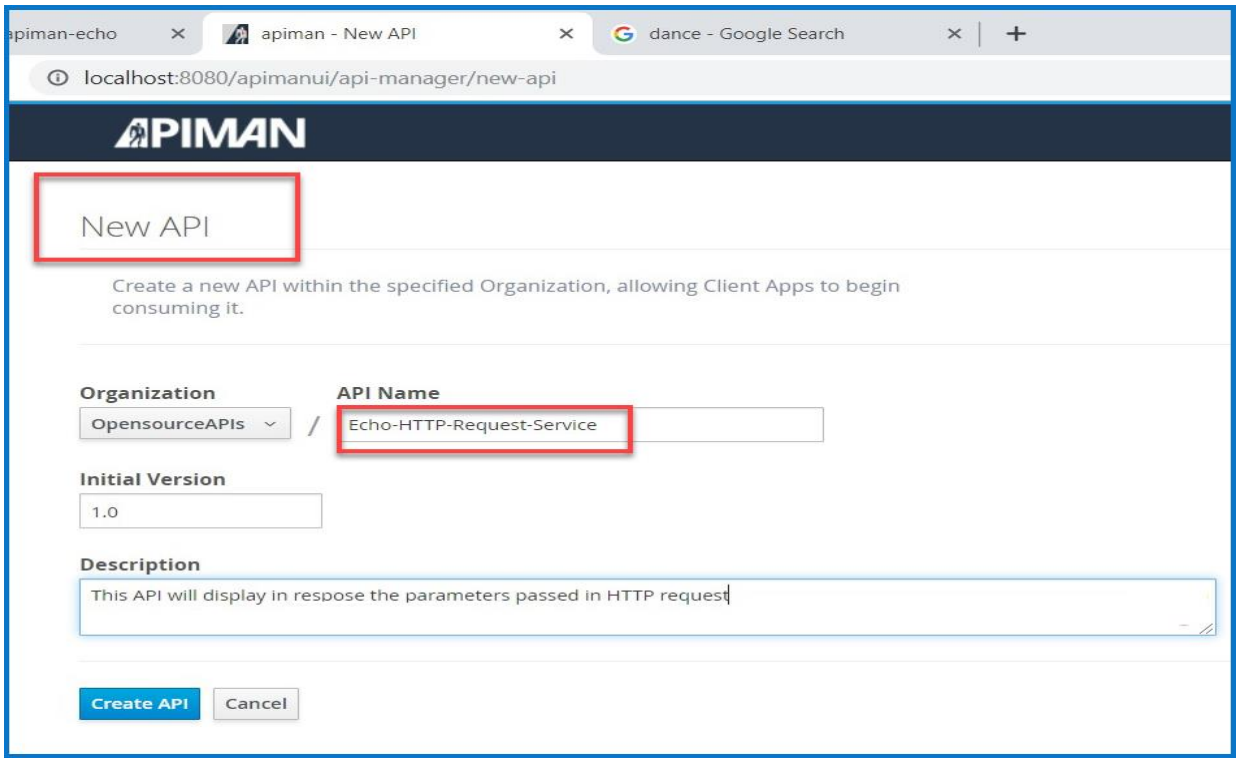


Figure 35: Define API page

7) Define API Implementation

After the API is defined, we have to define its implementation. In the context of the API Manager,

the API Endpoint is the API's direct URL. Remember that the API Gateway will act as a proxy for the API, so it must know the API's actual URL. In the case of our example API, the URL is: http://localhost:8080/apiman-echo

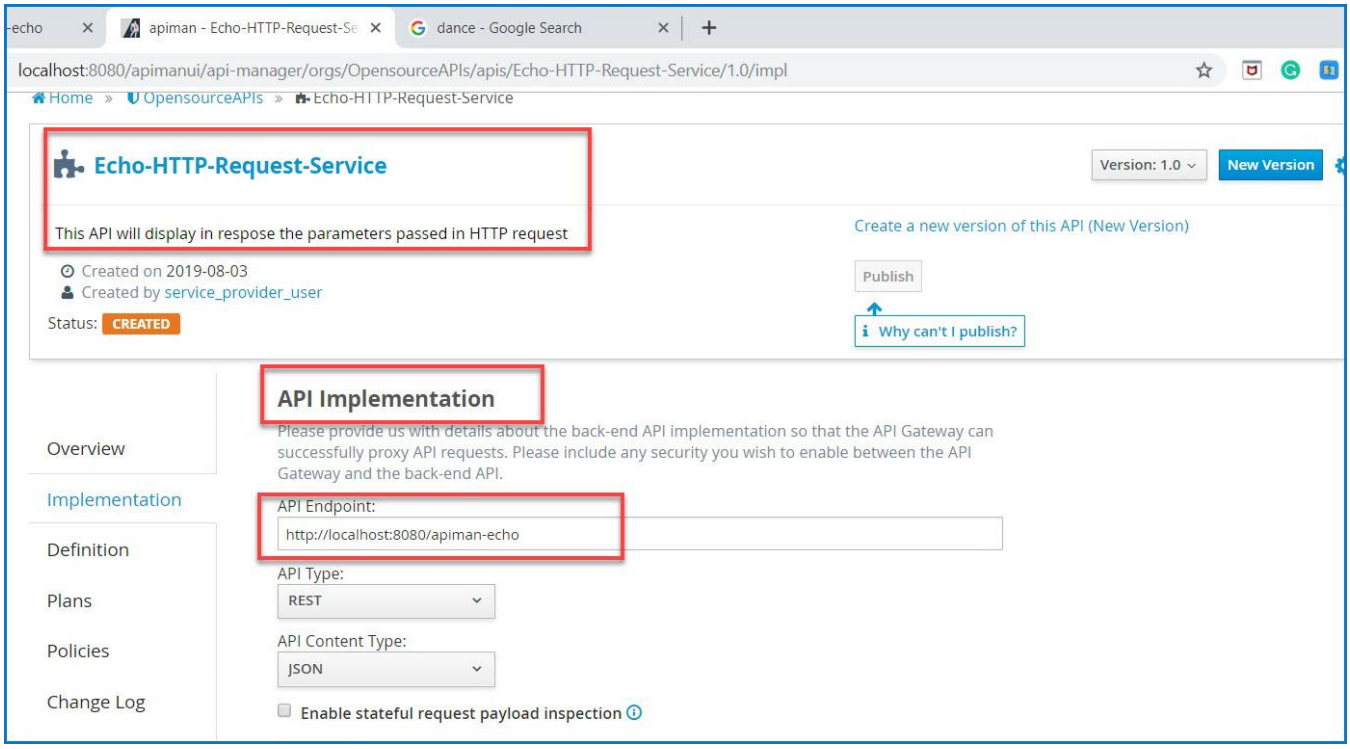


Figure 36: API Implementation

8) Assign the plans (policies) to API

Select the number of plans from the list and click **Publish**

Note: For sample example, two plans are selected Authentication and WelcomeToOpenSource.

The status of API is now turned to **Published,** as shown in figure 36 below.

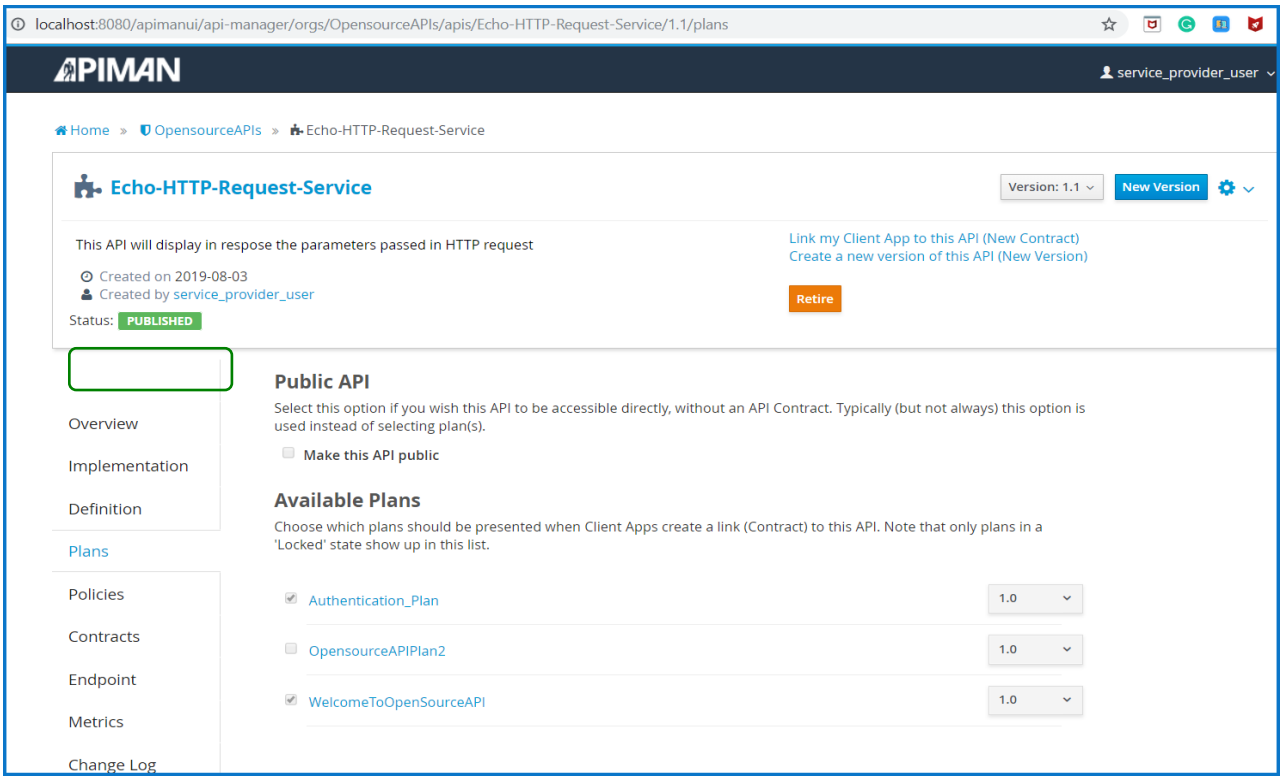


Figure 37: API published

The API provider has created, configured and published the API Echo-HTTP-Request-Service (alias of echo-service). Now it's time for the consumer to consume the API in its application.

Working with API Consumer

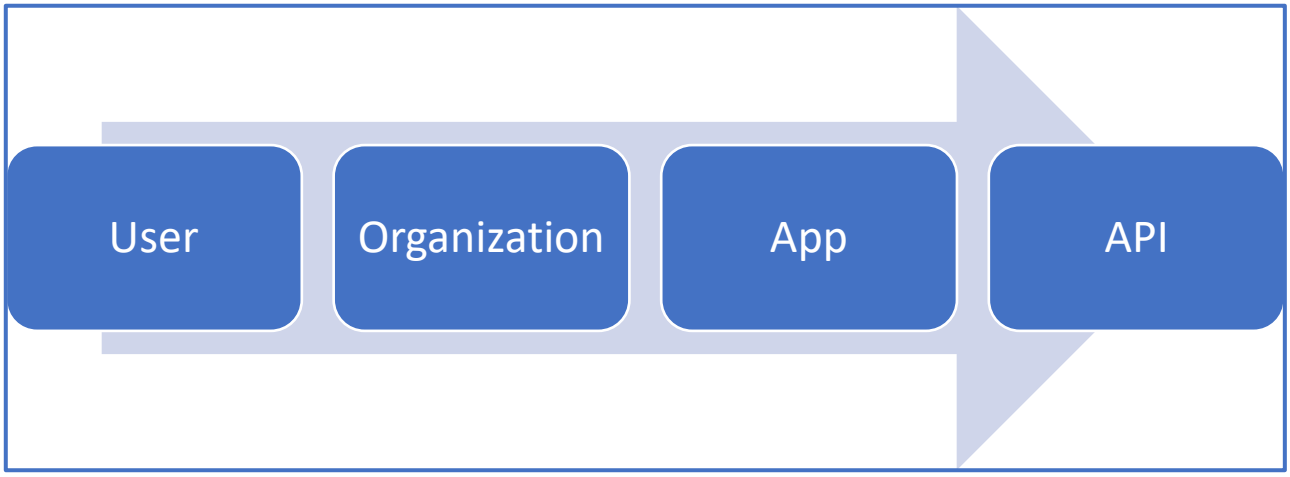


Figure 38: API consumer process flow

The API consumer is an Application Developer of an application which needs to use (consume) the API.

Following are the steps to be performed to achieve the same:

1) Login to apiman UI as **API consumer user** from APIMAN Realm Page.
   (Refer APIMAN Login(Realm) Screen shown in Figure 16 above).
2) ON apiman UI page Click **Create New Organization** under **Organization**
3) Create Application MyApp from apiman UI (API Manager Home page)

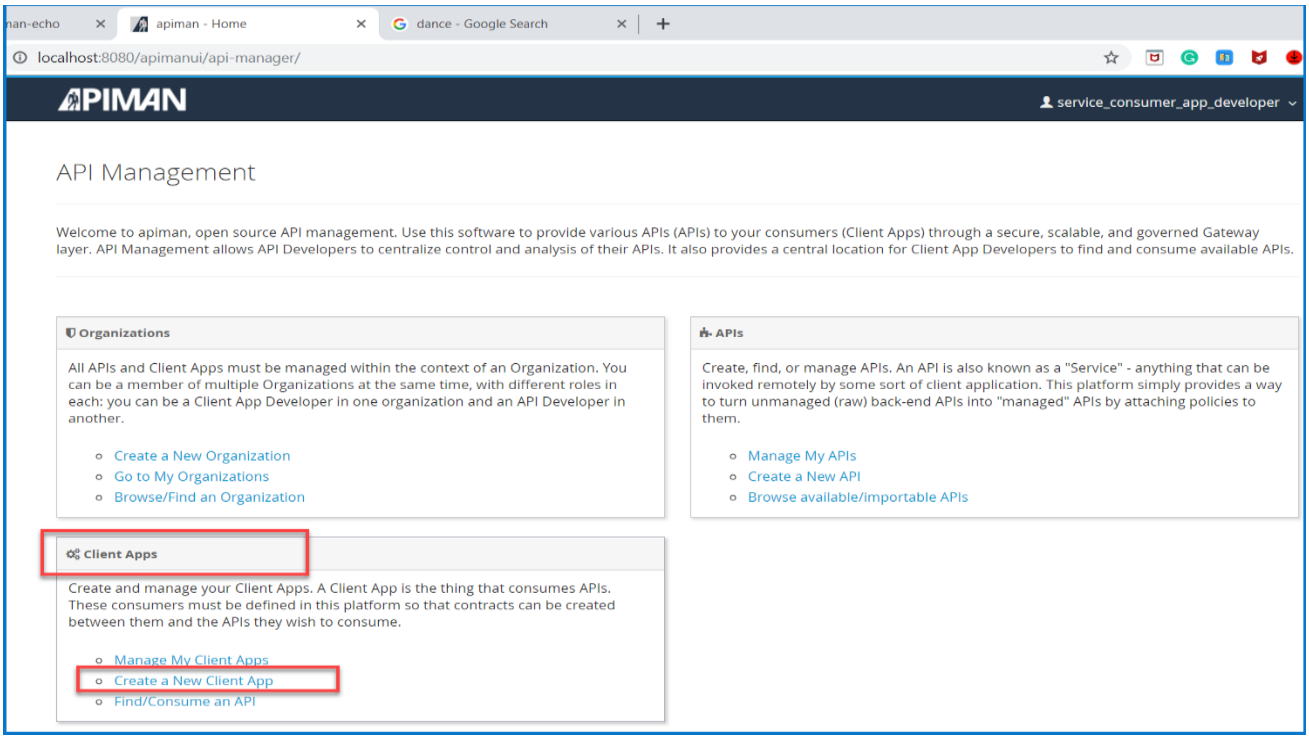


Figure 39: Create app

4) Search for Echo-HTTP-Request-Service API.

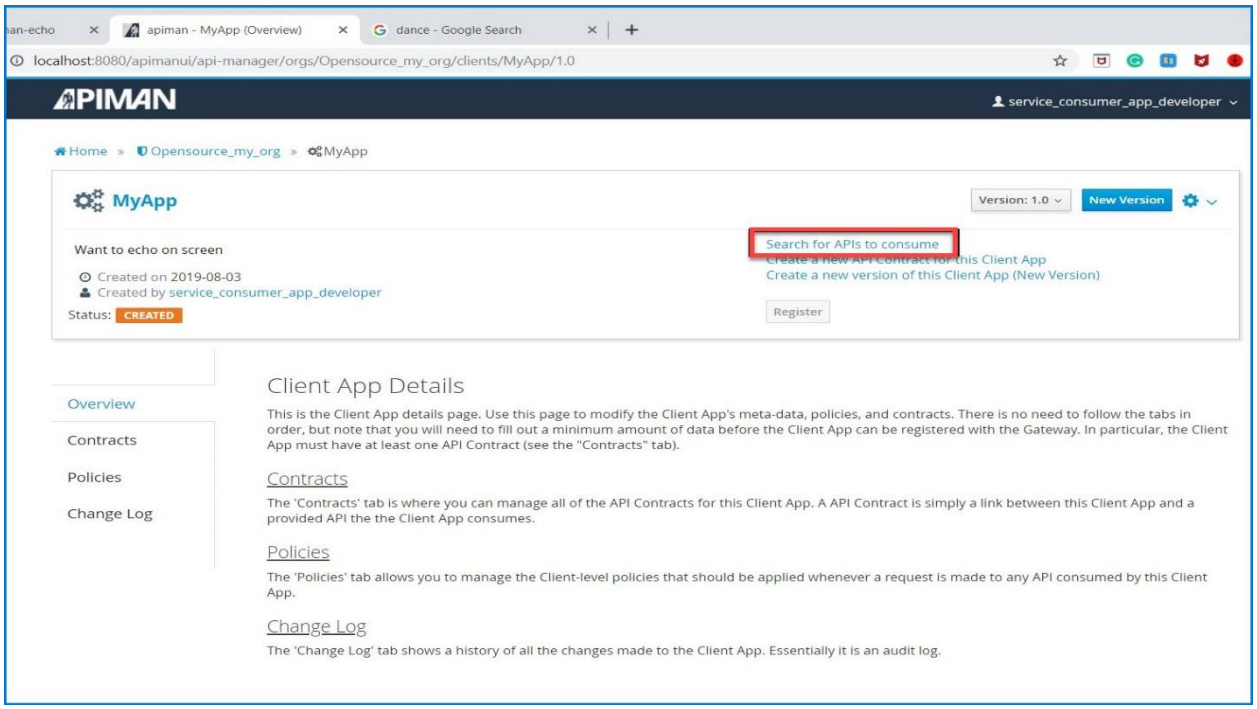


Figure 40: Search API to consume

5) Create a contract

This step is required to invoke the API through the app.
Remember, API is a contract between Provider and Consumer.
It binds both

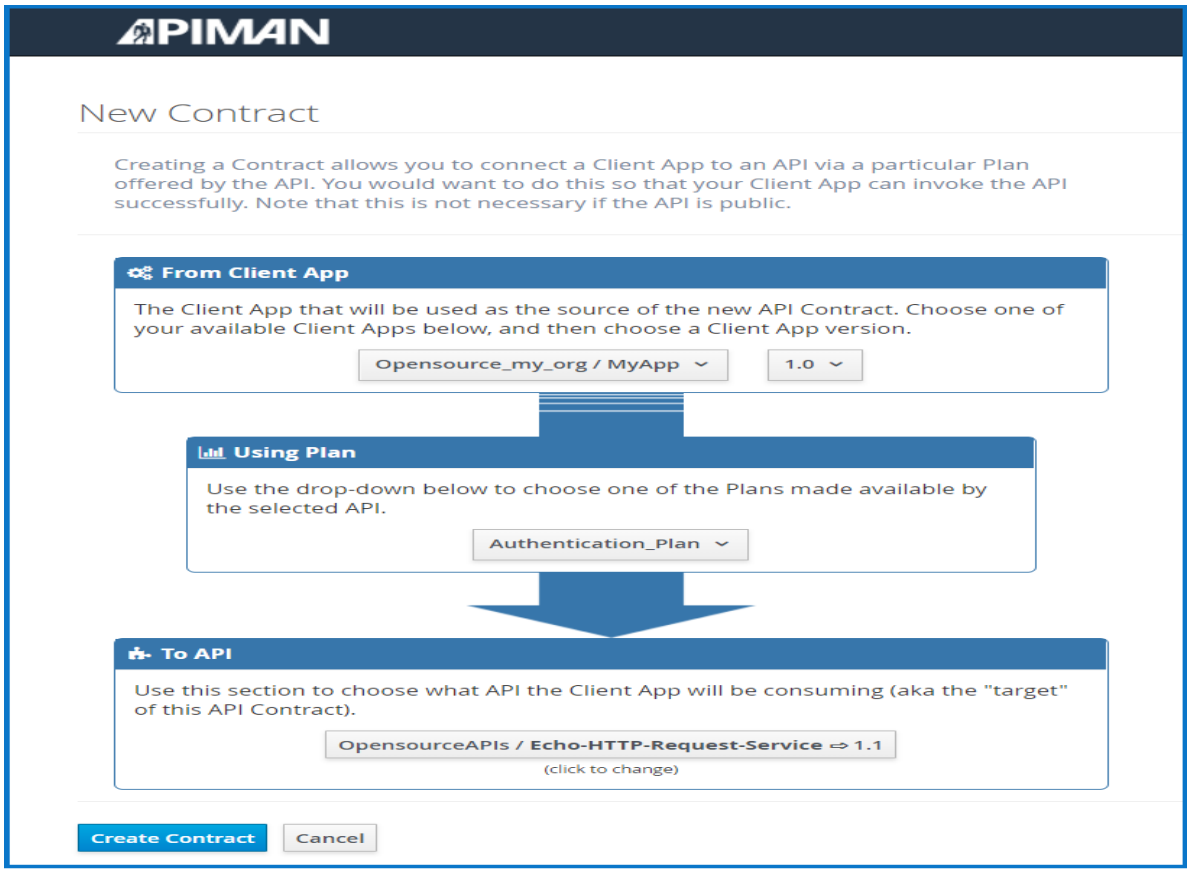


Figure 41: New contract

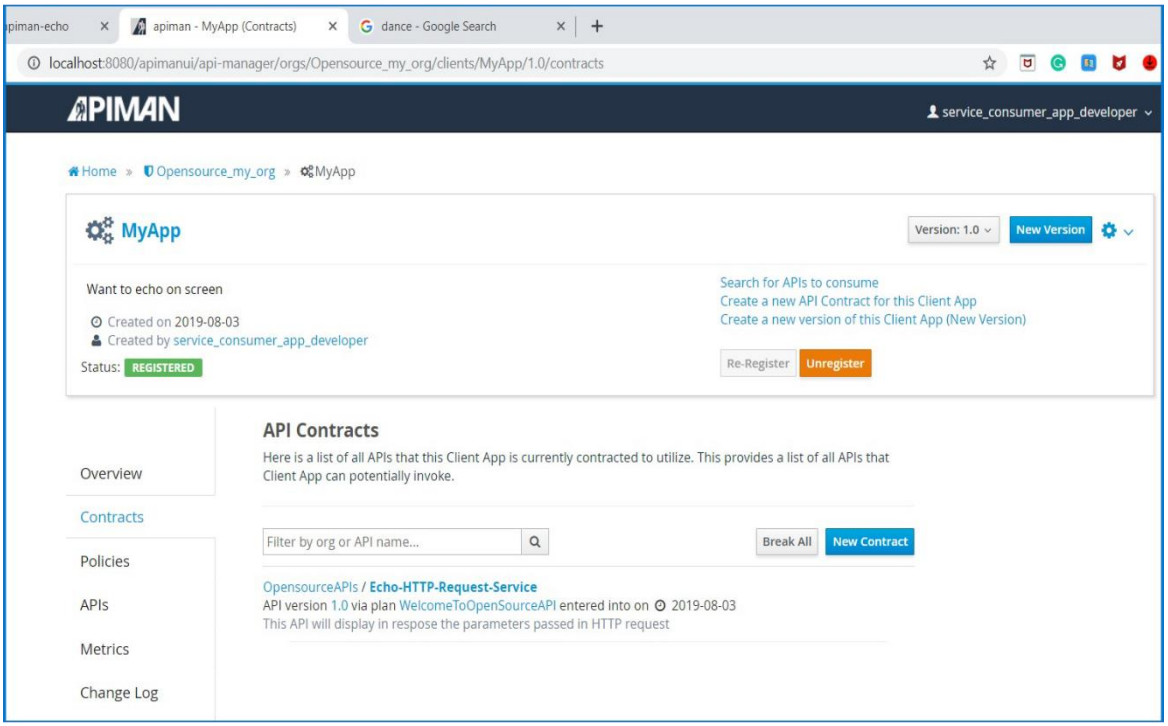


Figure 42: Contract created

6) Register App

The last step is to **register** the application with the API Gateway so that the gateway can act as a proxy for the API.

Voila !! Both API provider and Consumer are set up to use the Echo-HTTP-Request-Service.

Accessing managed echo-service API

Recall the note mentioning, in order to access the managed API through the API Gateway acting as a proxy for other API we have to obtain the managed API's URL. Refer to page no 15.

Follow the below steps to get the managed URL to run the echo-service

1) In the API Manager UI, on Home page click on **Manage My Client Apps** under **Client Apps**

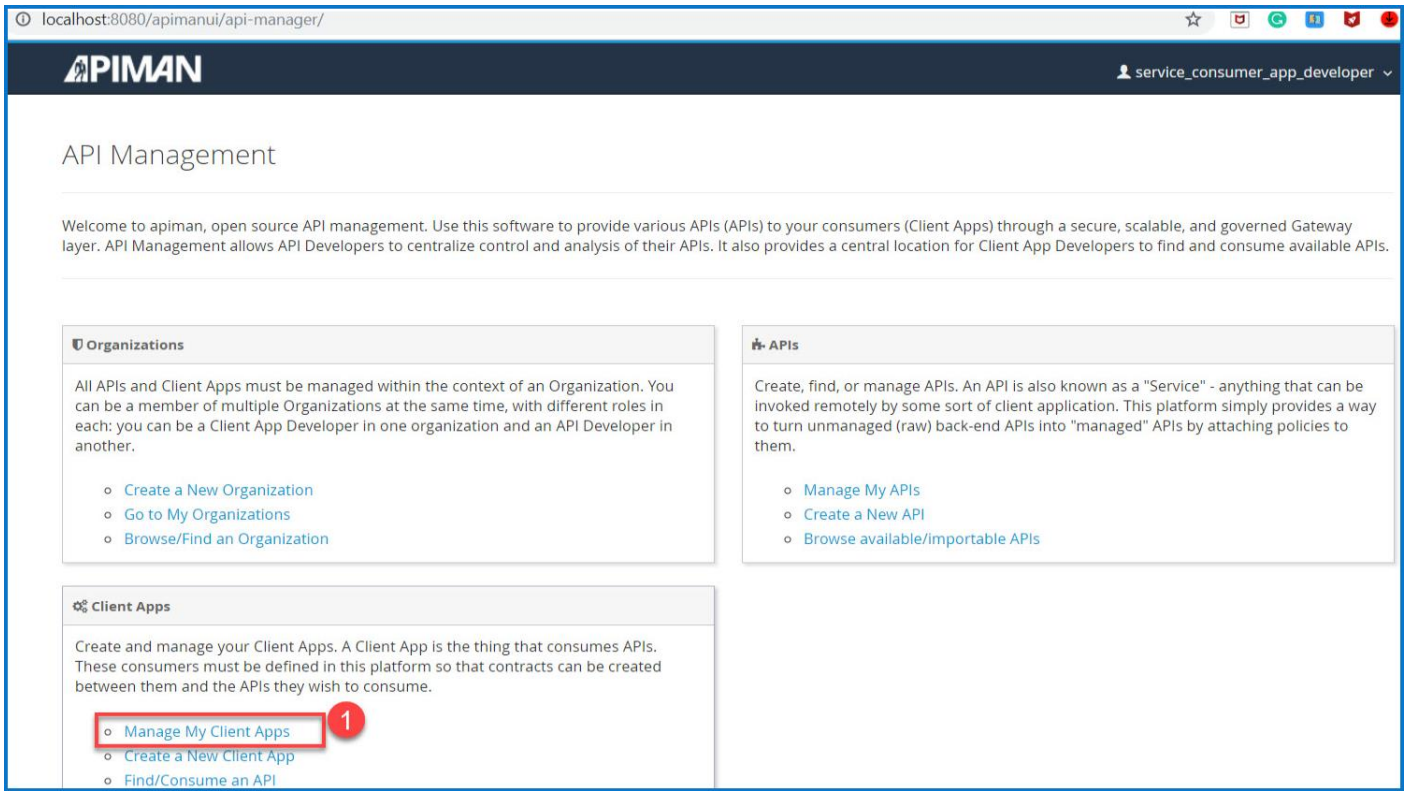


Figure 43: Manage Client App

2) Select the application

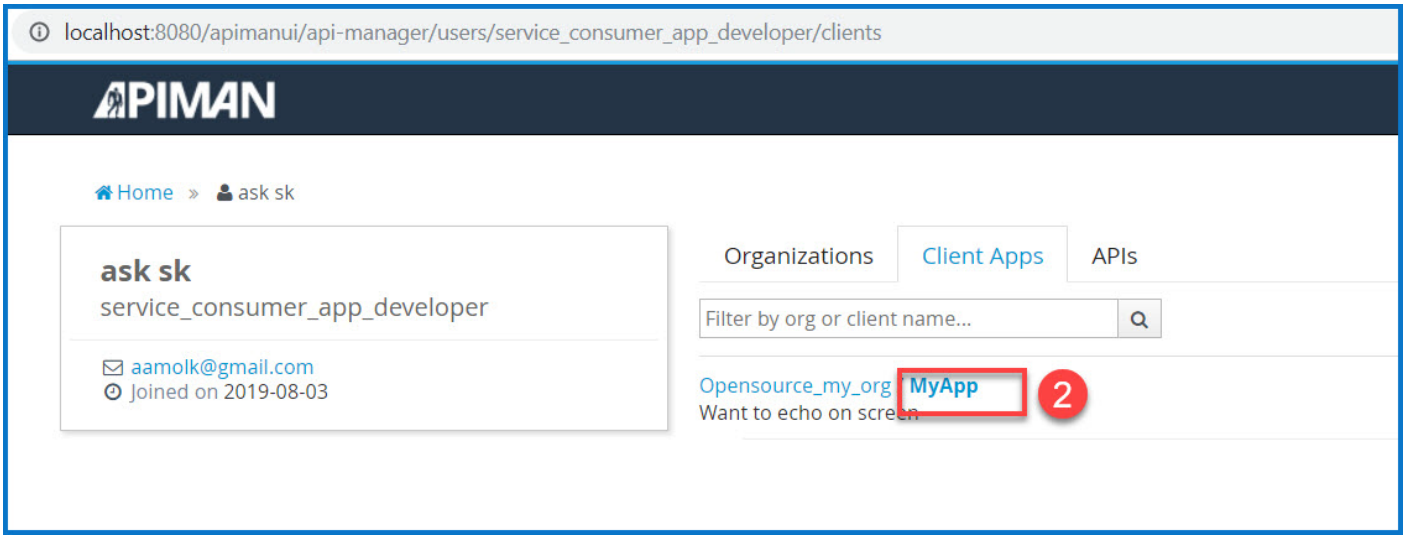


Figure 44: Select App

3) Select **APIs** tab in Myapp application page

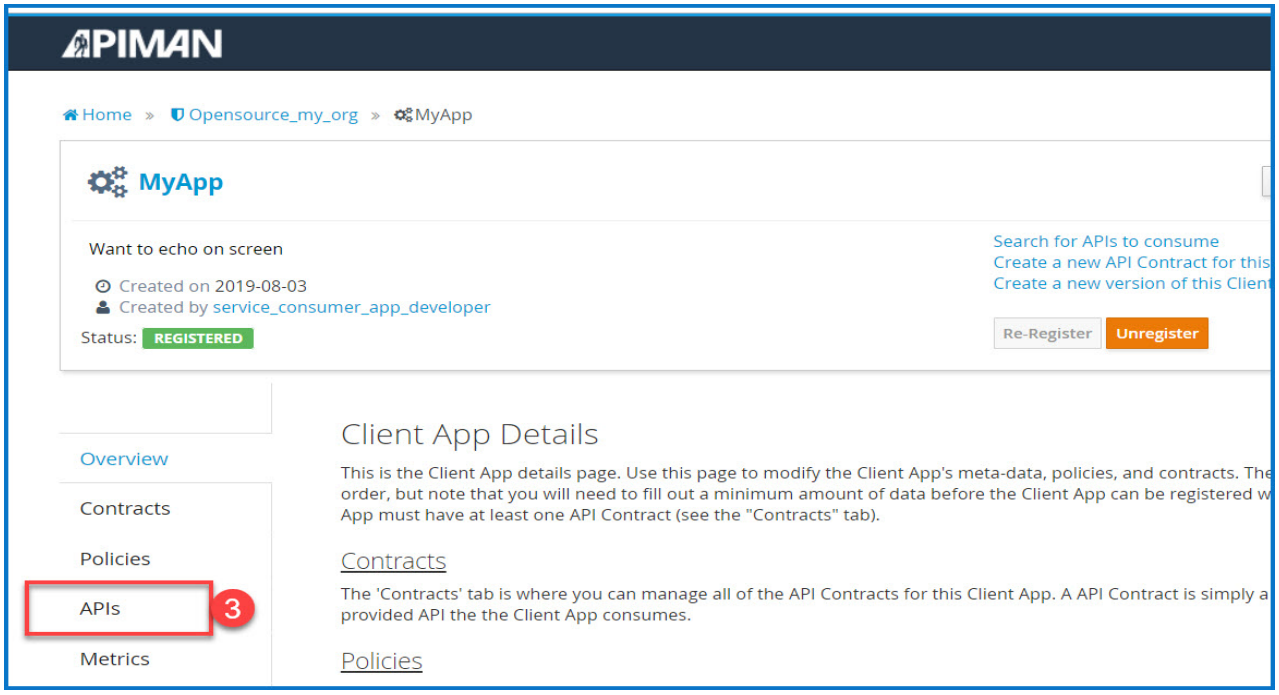


Figure 45: API tab in Application

4) Select the API and its endpoint URL
On the **Client App APIs** page, list of APIs will be displayed. Do get the URL of respective API, click on the **"i"** character to the right of the API name shown in figure 44 below.

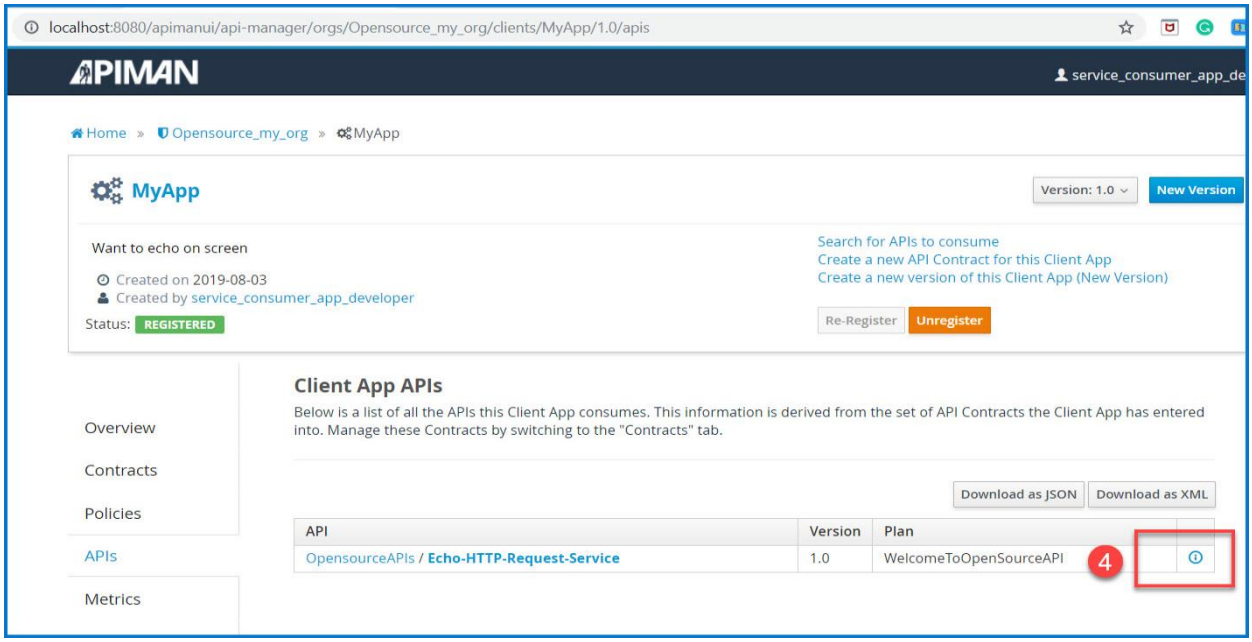


Figure 46: Get URL icon

This will expose the API Key and the API's HTTP endpoint in the API Gateway.

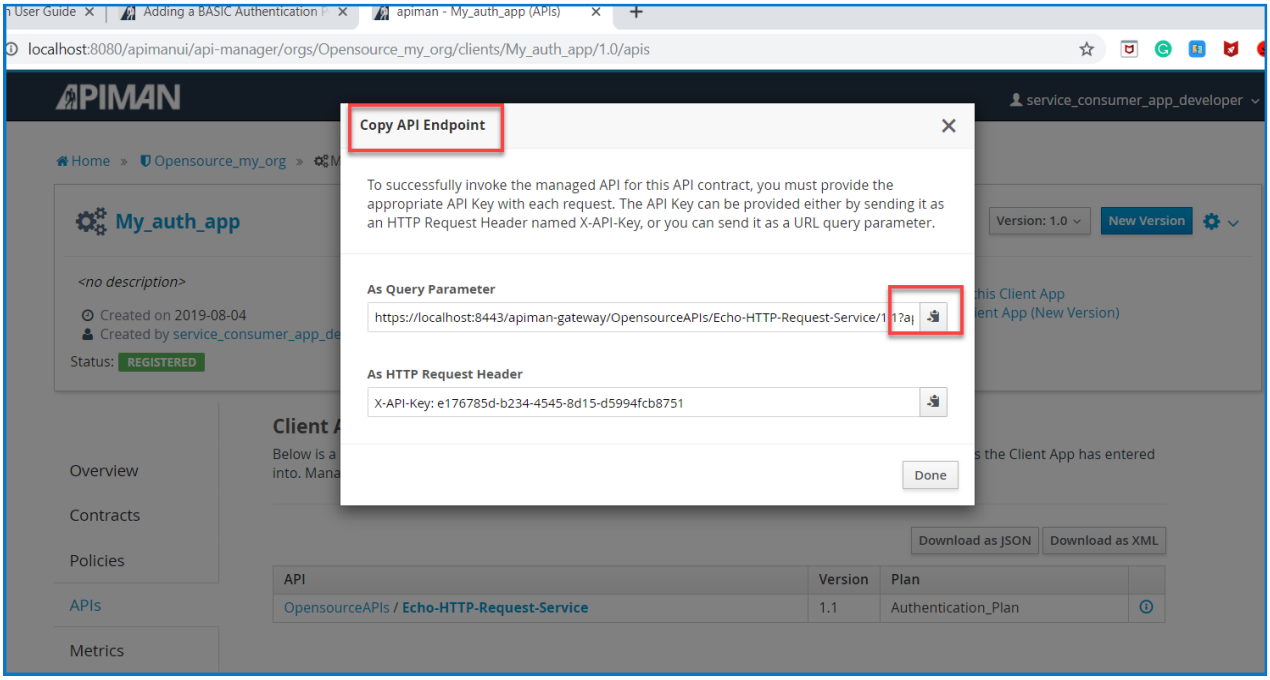


Figure 47: Copy API Endpoint

In order to be able to access the API through the API Gateway, we have to provide the API Key with each request. Combine the API Key and HTTP endpoint. The API Key can be provided either through an HTTP Header (X-API-Key) or a URL query parameter.

Copy the URL into the clipboard by clicking on a copy to clipboard button at right.

Endpoint for managed APIs here is :

https://localhost:8443/apiman-gateway/OpensourceAPIs/Echo-HTTP-Request-Service/1.1?apikey=e176785d-b234-4545-8d15-d5994fcb8751

5) Run the API from the browser

Copy the above URL in the browser

The API will ask for username and password (configured by the service provider as an authentication policy). This will be shared by the provider to the consumer

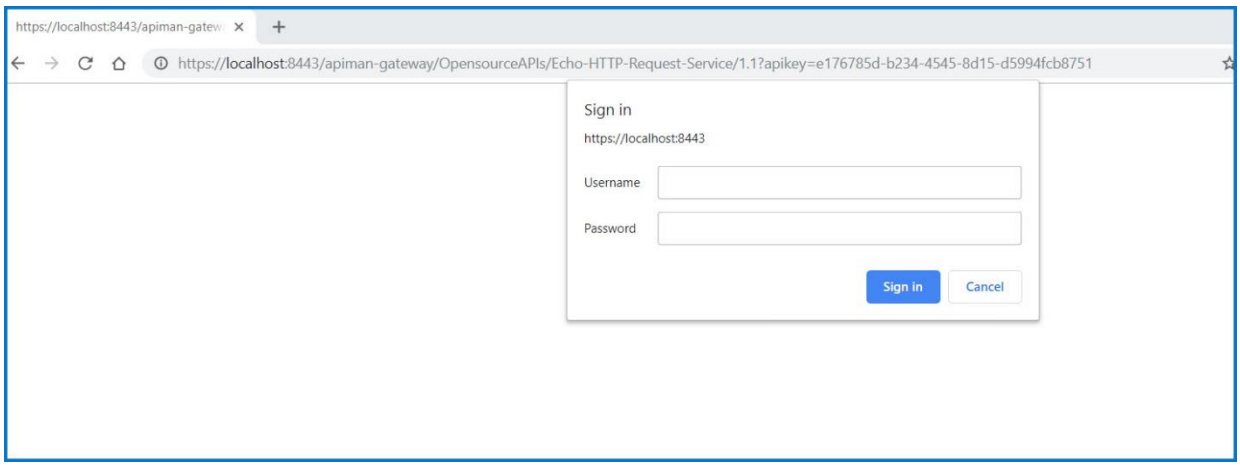


Figure 46: API Authentication

The API output (response) with authenticated user name will be seen as below.

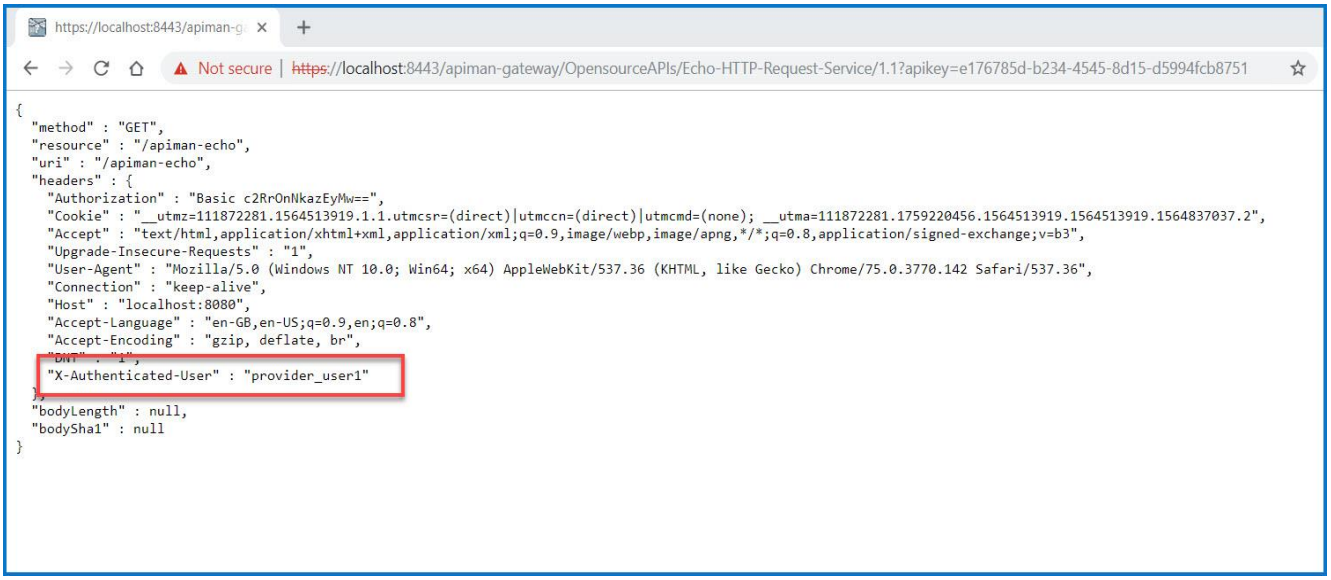


Figure 49: Managed echo-service API response

Congratulations !! we have successfully tested the Authentication policy applied to API through APIMAN.

Index