



RedHat

Apiman

**An Opensource API Management
(on-premise)**

Table of Contents

Introduction	1
<i>Application Programming Interface (API)</i>	<i>1</i>
<i>API Management</i>	<i>1</i>
<i>APIMan (Redhat API Management System)</i>	<i>1</i>
Getting started with APIMan	3
<i>Software Requirements.....</i>	<i>3</i>
<i>Downloading required softwares on Windows OS.....</i>	<i>3</i>
<i>Installing the required softwares</i>	<i>6</i>
Installing on Windows Operating System	6
Installating on RedHat Linux Operating System	8
<i>Starting Wildfly Server on Windows OS with APiman config file</i>	<i>9</i>
<i>Launching apiman on your browser.....</i>	<i>10</i>
<i>APIMAN UI</i>	<i>12</i>
Working With Apiman	12
<i>Echo-Service QuickStart</i>	<i>13</i>
Running the unmanaged (without using apiman) echo-service API on the browser.....	13
Working with Managed echo-service API using APIMAN	16
Working with API Provider	17
Working with API Consumer	23
Accessing managed echo-service API.....	26

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

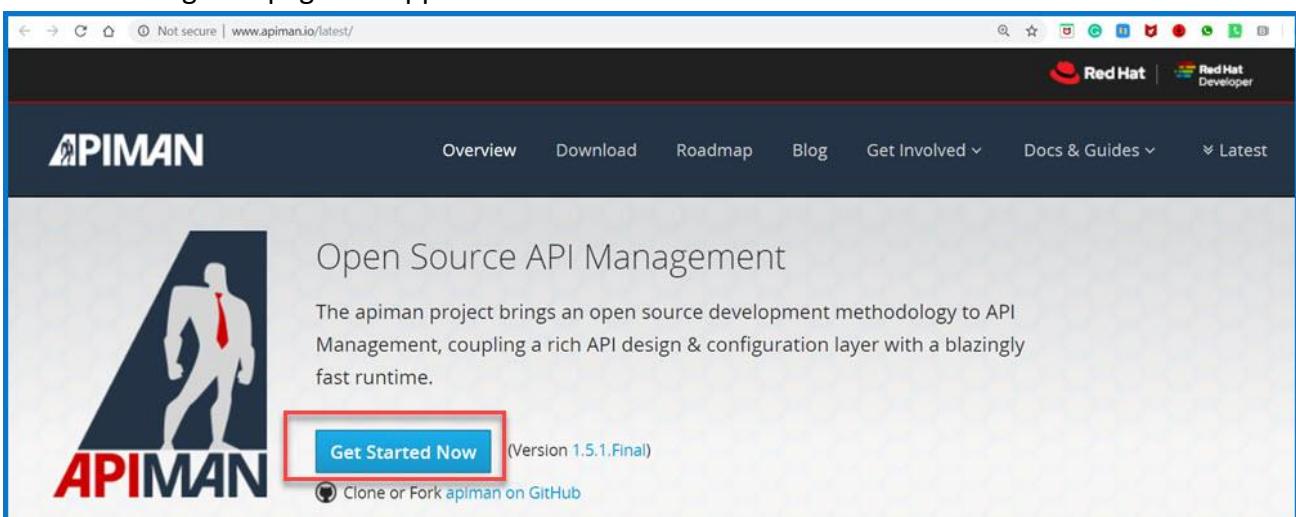


Figure 1: APIMan download link

Getting started with APIMan

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

The screenshot shows the APIMan download page. At the top, there are tabs for WildFly 10 / EAP 7.1, WildFly 11, WildFly 11 & Vert.x Gateway, 3scale & Headless, Tomcat 8+, and EAP 7. The WildFly 10 / EAP 7.1 tab is highlighted with a red border. Below the tabs, there is a section titled "Download" containing two links: "WildFly 10.1.0.Final" and "Apiman 1.5.1.Final overlay for WildFly 10".

Figure 2:APIMAN download page

3. Click the respective tab to download a specific version of WildFly (Example WildFly 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

The screenshot shows the WildFly Downloads page. At the top, there is a navigation bar with links for Home, About, News, Downloads, Documentation, Source Code, Get Help, Join Us, Thorntail, CI, and Follow Us. Below the navigation bar, there is a banner for WildFly 10.1.0.Final. The main content area is titled "Downloads" and contains a table of available distributions:

Version	Date	Description	License	Size	Format	Checksum
17.0.1.Final	2019-07-03	Java EE Full & Web Distribution	LGPL	176 MB	ZIP	SHA-1
		Servlet-Only Distribution	LGPL	41 MB	ZIP	SHA-1

Figure 3: Table listing the wildfly 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.**]

The screenshot shows the WildFly 10 distribution table. The table has columns for Version, Date, Description, License, Size, Format, and Checksum. The first row, which corresponds to the Java EE Full & Web Distribution, has its ZIP link highlighted with a red border.

Version	Date	Description	License	Size	Format	Checksum
10.0.0.Final	2016-01-29	Java EE Full & Web Distribution	LGPL	132 MB	ZIP	
		Servlet-Only Distribution	LGPL	131 MB	TGZ	
		Application Server Source Code	LGPL	25 MB	ZIP	
				15 MB	TGZ	

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

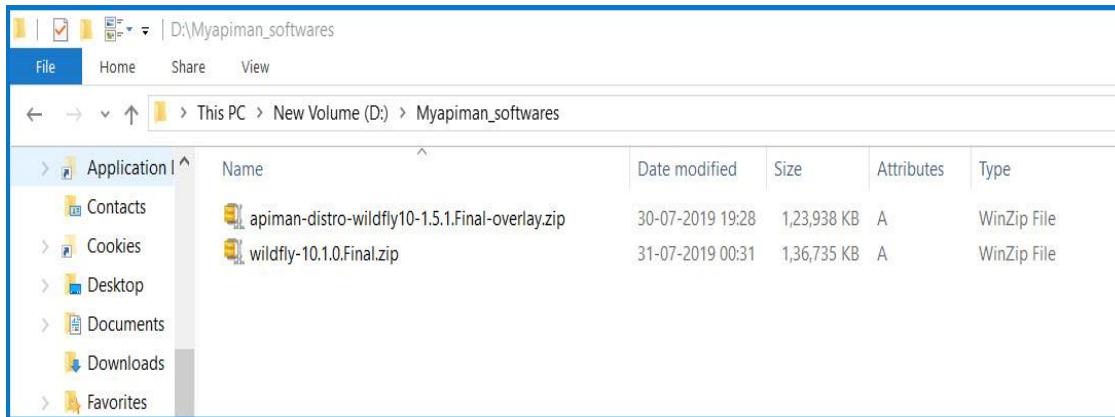


Figure 5: Apiman download directory

Note: The apiman zip (war²) file is an **overlay³** 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

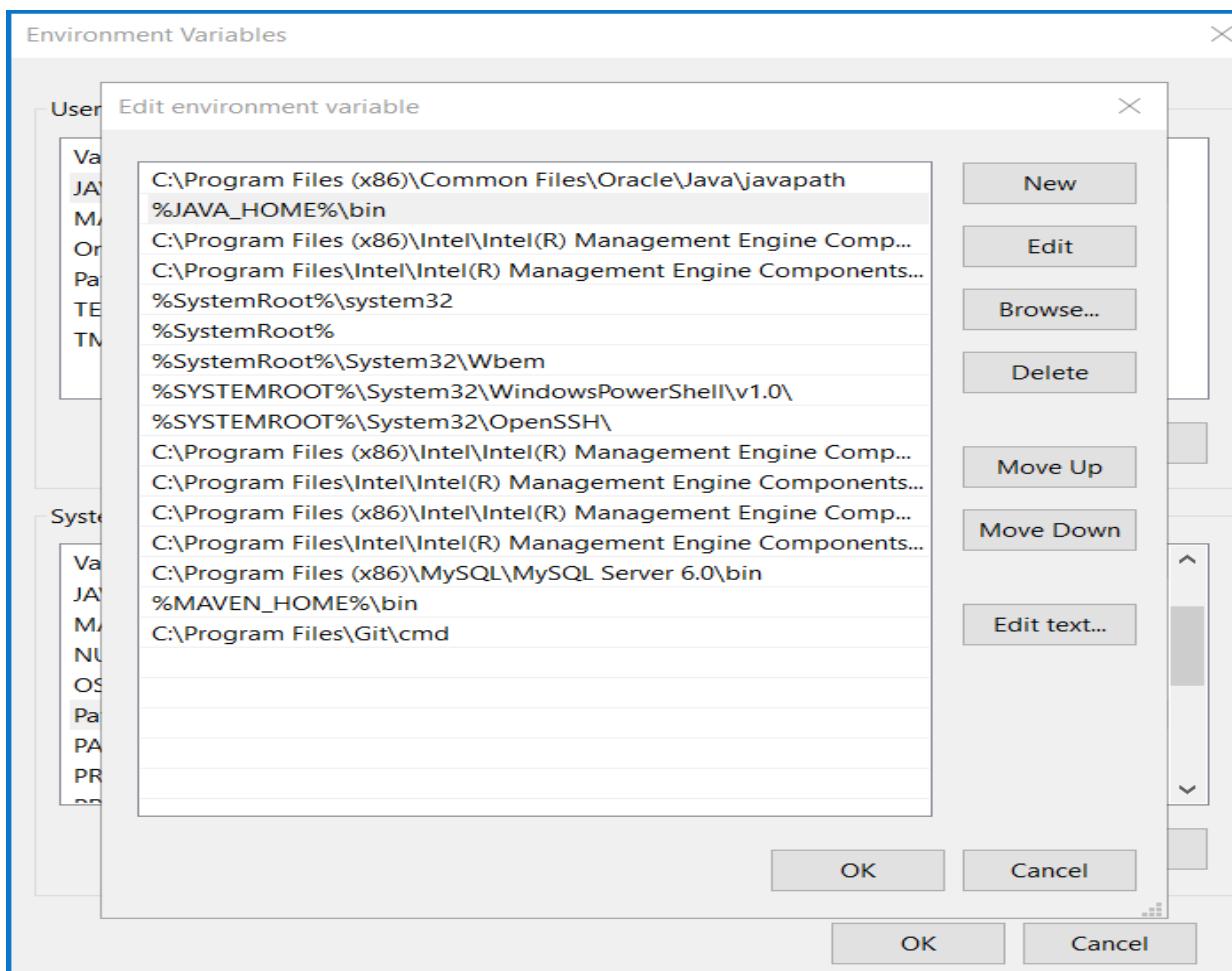


Figure 6: Windows system environment variable setting

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

Name	Date modified	Size	Attributes	Type
.installation	18-08-2016 19:12		D	File folder
appclient	18-08-2016 19:12		D	File folder
bin	18-08-2016 19:12		D	File folder
docs	18-08-2016 19:12		D	File folder
domain	18-08-2016 19:12		D	File folder
modules	18-08-2016 19:12		D	File folder
standalone	18-08-2016 19:12		D	File folder
welcome-content	18-08-2016 19:12		D	File folder
copyright.txt	18-08-2016 19:12	3 KB	N	Text Document
jboss-modules.jar	18-08-2016 19:12	358 KB	N	Executable Jar File
LICENSE.txt	18-08-2016 19:12	26 KB	N	Text Document
README.txt	18-08-2016 19:12	3 KB	N	Text Document

Figure 7: WildFly installation 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 YES⁴.

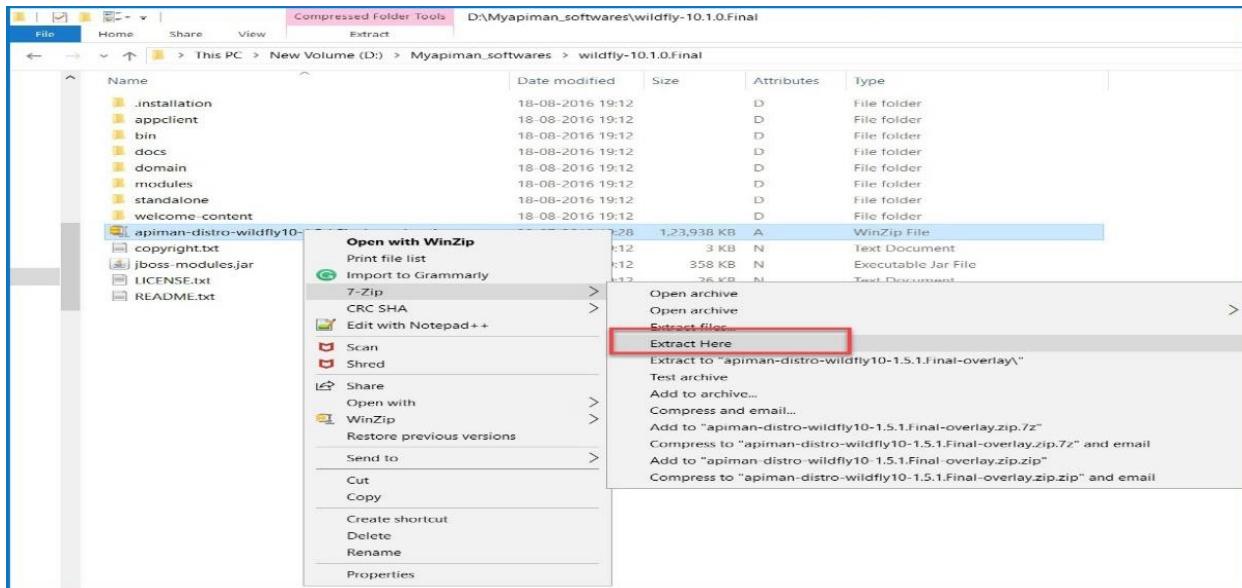


Figure 8: Unzipping apiman overlay zip file

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

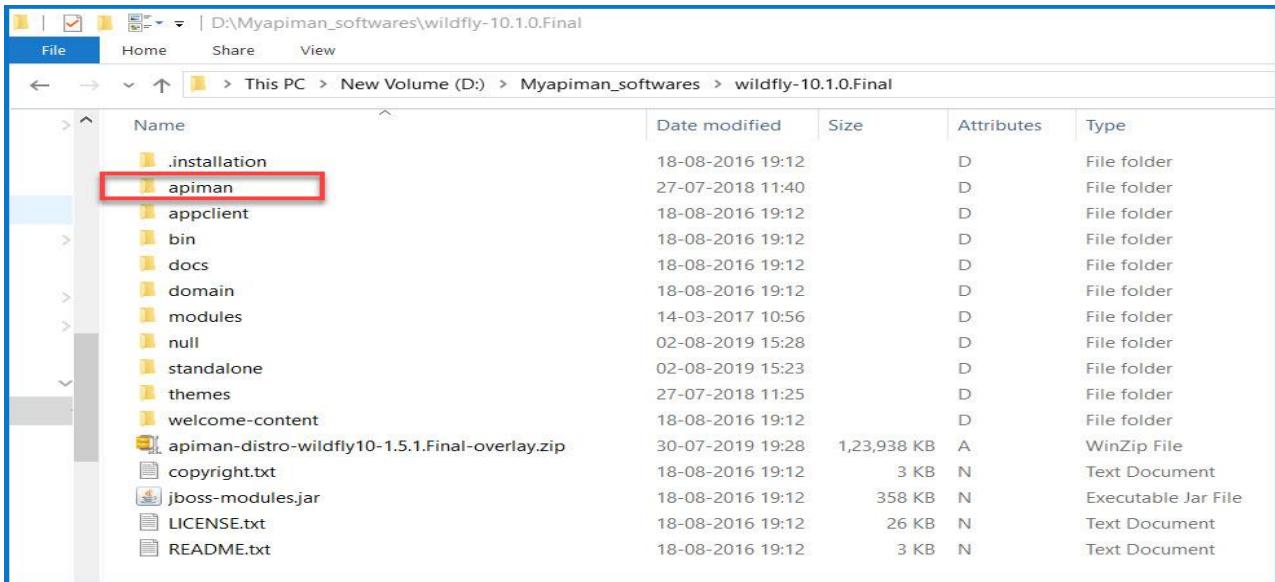


Figure 9: Apiman extracted directory structure in Wildfly folder

Installation on RedHat Linux Operating System

- 1) Got to <http://www.apiman.io/>

The screenshot shows the Apiman website homepage. The sidebar on the right has a section titled "Tutorials & Walkthroughs" with a "Crash Course!" link, which is highlighted with a red box.

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.

```
Administrator: Windows PowerShell
$ D:\Myapiman_softwares\wildfly-10.1.0.Final\bin> ./standalone.ps1 -c standalone-apiman.xml
=====
JBoss Bootstrap Environment
JBoss_HOME: D:\Myapiman_softwares\wildfly-10.1.0.Final
JAVA: C:\Program Files\Java\jdk1.8.0_221\bin\java.exe
MODULE_OPTS:
JAVA_OPTS: -Xms64M -Xmx512M -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman
=====
5:23:16,785 INFO [org.jboss.modules] (main) JBoss Modules version 1.5.2.Final
5:23:19,225 INFO [org.jboss.msc] (main) JBoss MSC version 1.2.6.Final
5:23:19,652 INFO [org.jboss.as] (MSC service thread 1-7) WFLYSRV0049: WildFly Full 10.1.0.Final (WildFly Core 2.2.0.Final) starting
5:23:27,315 INFO [org.jboss.as.repository] (ServerService Thread Pool -- 11) WFLYDR0001: Content added at location D:\Myapiman_softwares\wildfly-10.1.0.Final\standalone\content
5:23:27,836 INFO [org.jboss.as.repository] (ServerService Thread Pool -- 11) WFLYDR0001: Content added at location D:\Myapiman_softwares\wildfly-10.1.0.Final\standalone\content
5:23:27,978 INFO [org.jboss.as.repository] (ServerService Thread Pool -- 11) WFLYDR0001: Content added at location D:\Myapiman_softwares\wildfly-10.1.0.Final\standalone\content
5:23:28,072 INFO [org.jboss.as.repository] (ServerService Thread Pool -- 11) WFLYDR0001: Content added at location D:\Myapiman_softwares\wildfly-10.1.0.Final\standalone\content
5:23:28,202 INFO [org.jboss.as.repository] (ServerService Thread Pool -- 11) WFLYDR0001: Content added at location D:\Myapiman_softwares\wildfly-10.1.0.Final\standalone\content
5:23:28,247 INFO [org.jboss.as.repository] (ServerService Thread Pool -- 11) WFLYDR0001: Content added at location D:\Myapiman_softwares\wildfly-10.1.0.Final\standalone\content
5:23:28,524 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0039: Creating http management service using socket-binding (management-http)
5:23:28,775 INFO [org.xnio] (MSC service thread 1-4) XNIO version 3.4.0.Final
5:23:28,798 INFO [org.xnio.nio] (MSC service thread 1-4) XNIO NIO Implementation Version 3.4.0.Final
5:23:28,839 WARN [org.jboss.as.txn] (ServerService Thread Pool -- 56) WFLYTX0013: Node identifier property is set to the default value. Please make sure it is unique.
5:23:28,861 INFO [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 40) WFLYCLINF0001: Activating Infinispan subsystem.
5:23:28,863 INFO [org.jboss.as.webservices] (ServerService Thread Pool -- 58) WFLWWS0002: Activating WebServices Extension
5:23:28,872 INFO [org.jboss.as.security] (ServerService Thread Pool -- 55) WFLYSEC0002: Activating Security Subsystem
```

Figure 11: WildFLy server start

Getting started with APIMan

- 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_6tBc0zjro08w}{[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>

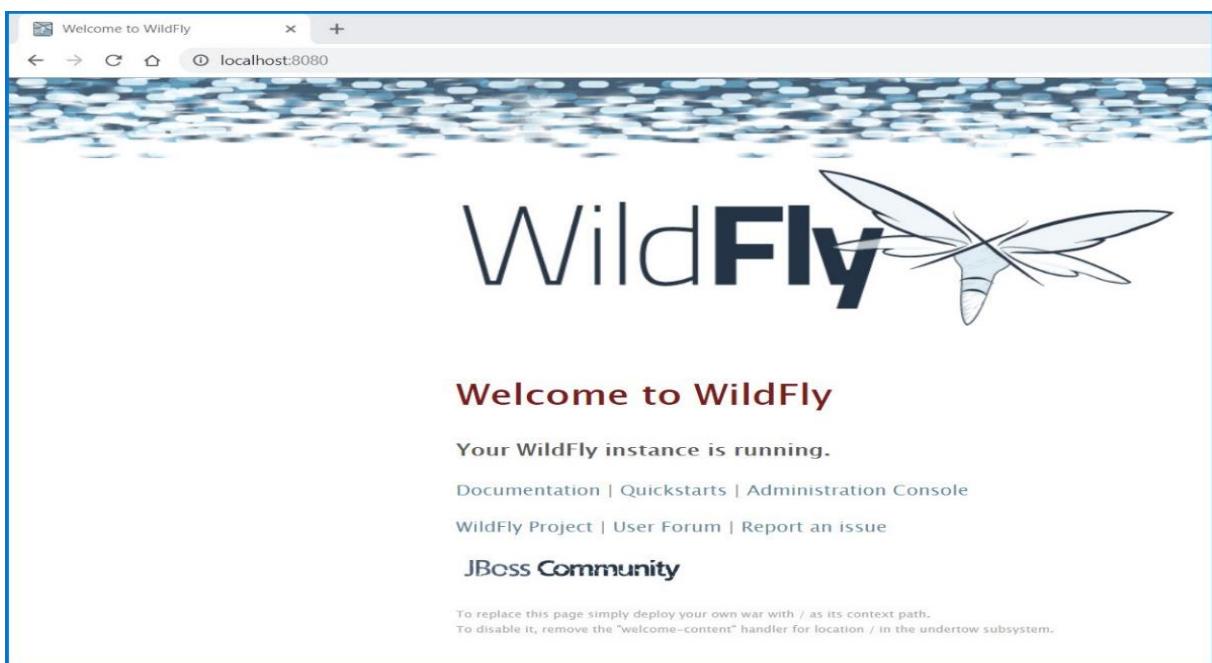


Figure 13: Wildfly Server home page

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

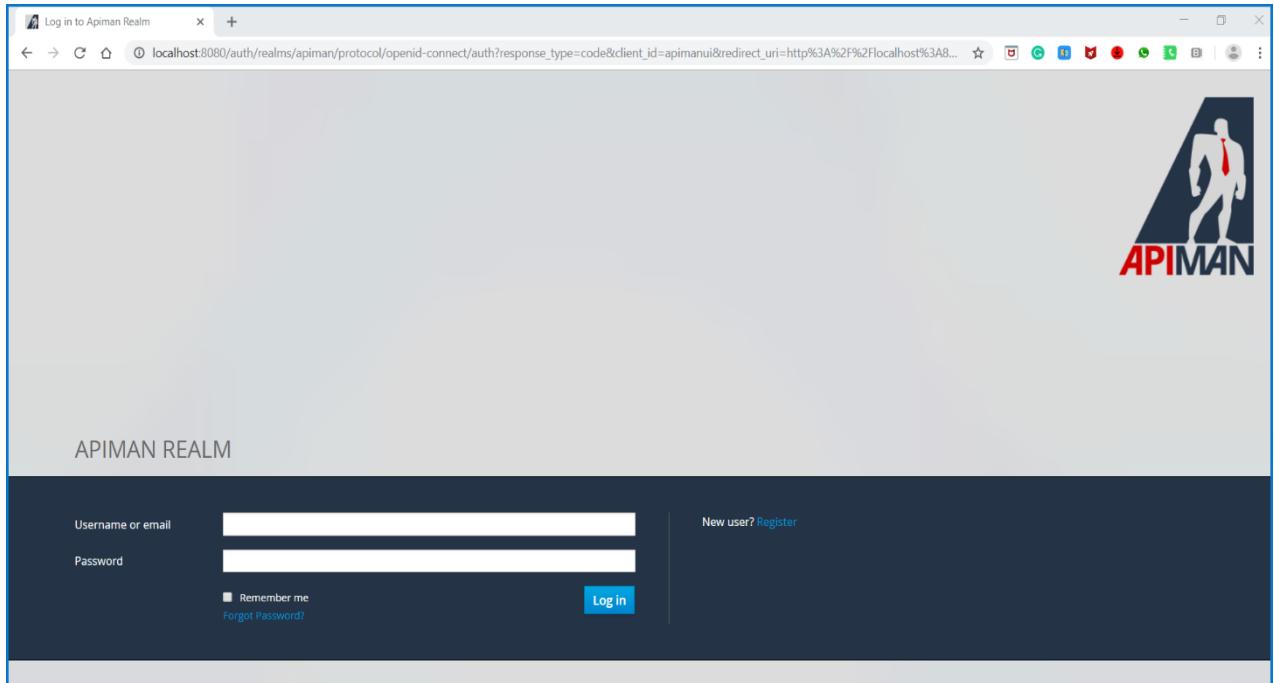


Figure 14: APiman Login(Realm) page

- 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

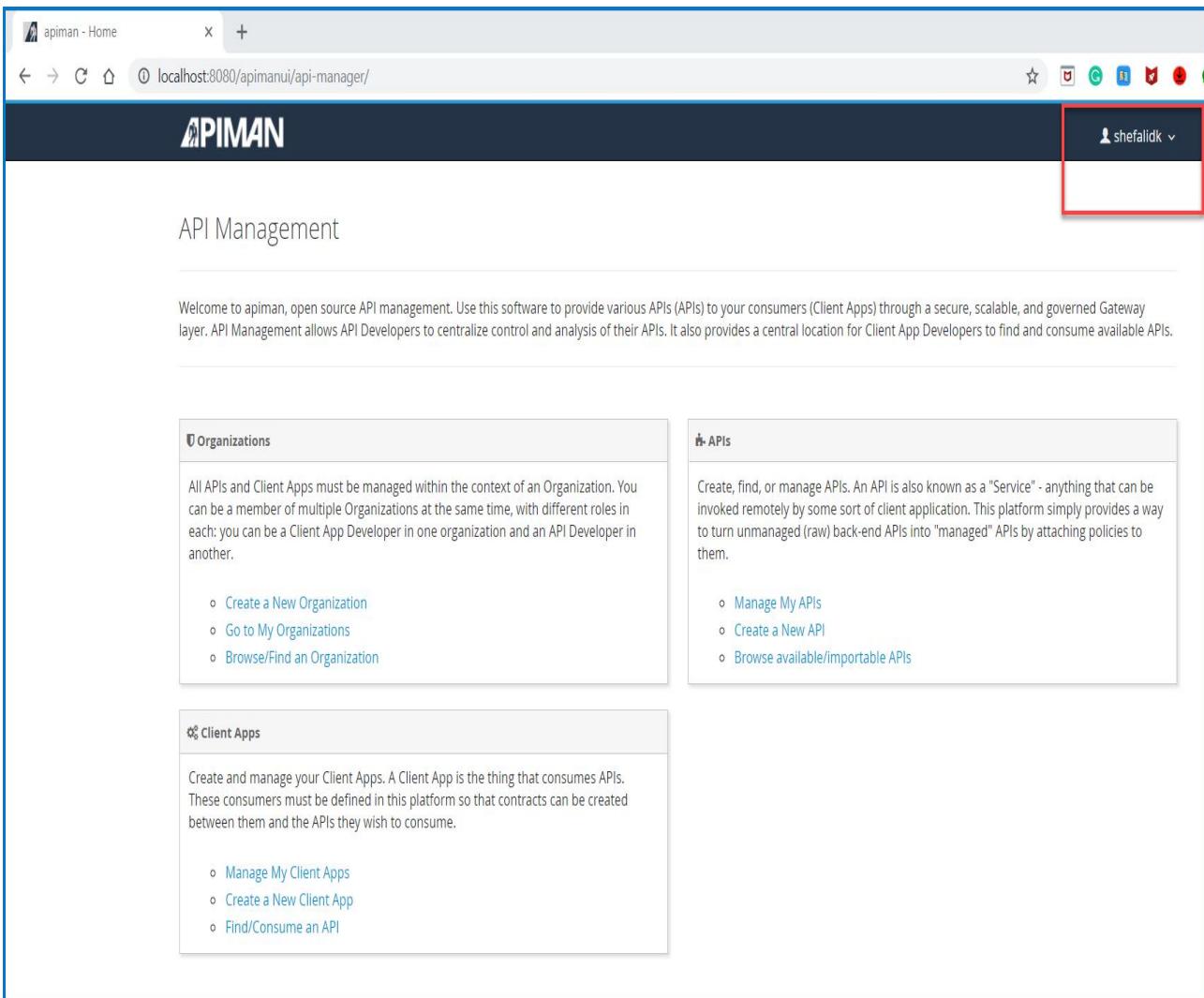
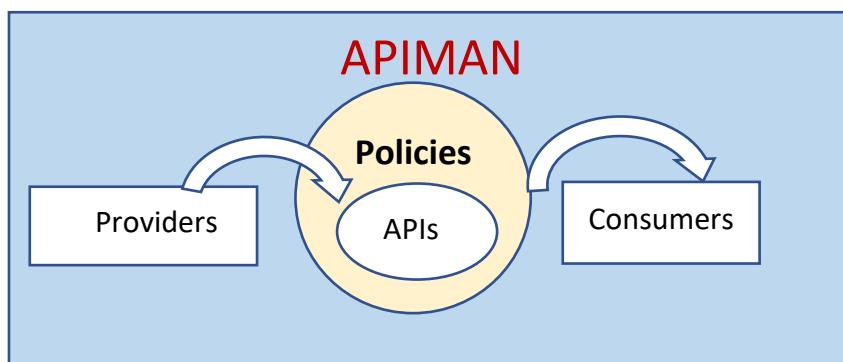


Figure 17: APIMan (API Manger UI) Home page

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
- Develop/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 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**

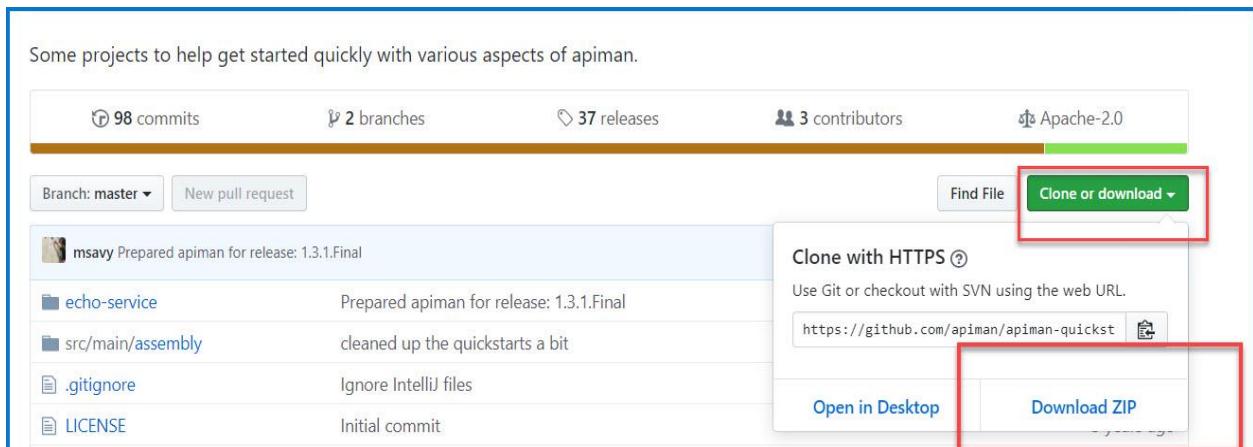


Figure 18: Git page for download echo-service

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

Working With Apiman

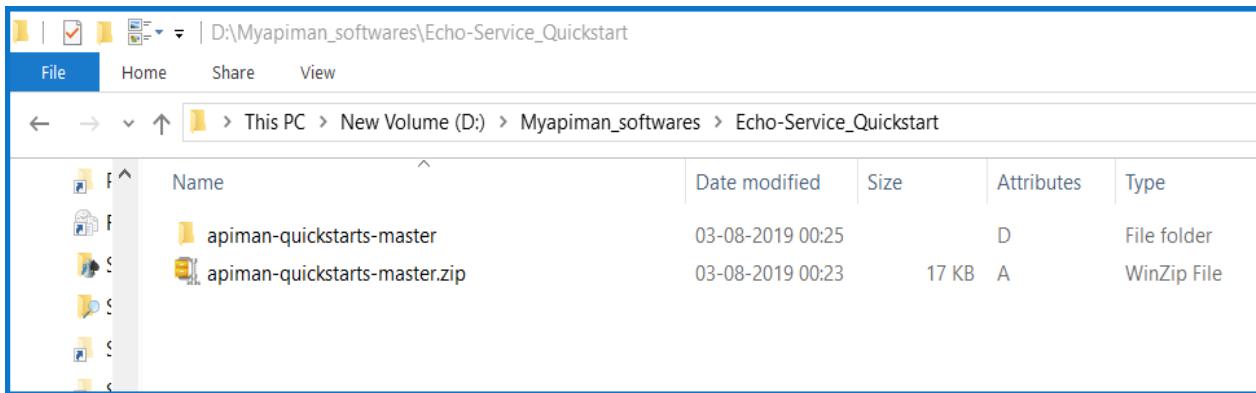


Figure 19: Unzipped apiman-quickstart folder

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

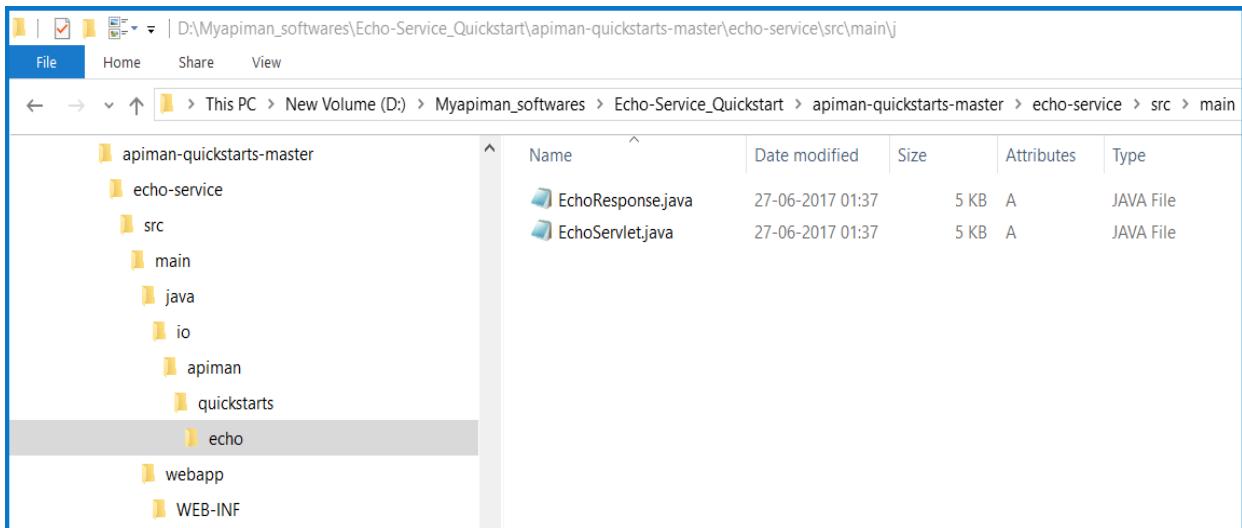


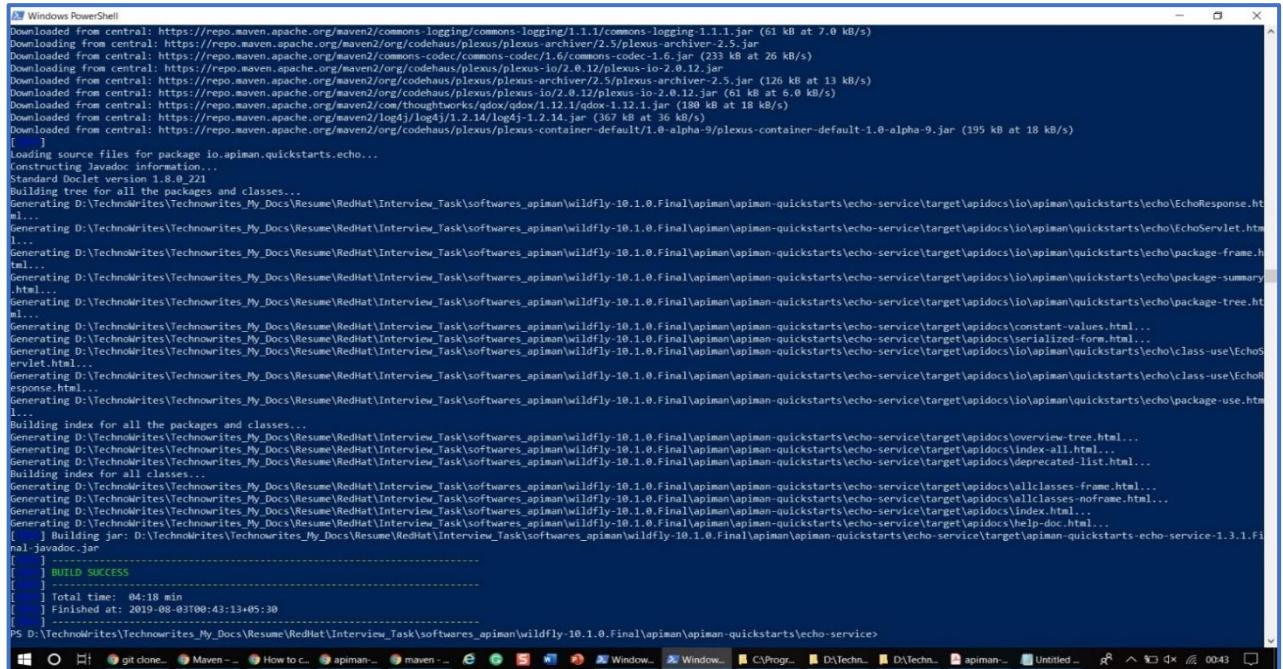
Figure 20: Directory structure of apiman-quickstart (echo-service) API

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.



```

Windows PowerShell
Downloaded from central: https://repo.maven.apache.org/maven2/commons-logging/commons-logging/1.1.1/commons-logging-1.1.1.jar (61 kB at 7.0 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/2.5/plexus-archiver-2.5.jar
Downloaded from central: https://repo.maven.apache.org/maven2/commons-code/commons-code/1.6/commons-code-1.6.jar (233 kB at 26 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.0.12/plexus-io-2.0.12.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.0.12/plexus-io-2.0.12.jar (126 kB at 13 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/gdox/gdox/1.12.1/gdox-1.12.1.jar (180 kB at 6.0 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/log4j/log4j/1.2.14/log4j-1.2.14.jar (367 kB at 36 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-container-default/1.0-alpha-9/plexus-container-default-1.0-alpha-9.jar (195 kB at 18 kB/s)
[ ]
Loading source files for package io.apiman.quickstarts.echo...
Constructing Javadoc information...
Standard Doclet version 1.8.0_221
Building tree for all the packages and classes...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\EchoResponse.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\EchoServlet.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\package-frame.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\package-summary.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\package-tree.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\constant-values.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\serialized-form.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\class-use\EchoServlet.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\class-use\EchoService.html...
Generating D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo\package-use.html...
[ ] Building jar: D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service\target\apidocs\io\apiman\quickstarts\echo-service-1.3.1.Final-javadoc.jar
[ ] -----
[ ] BUILD SUCCESS
[ ] -----
[ ] Total time: 04:18 min
[ ] Finished at: 2019-08-03T08:43:13+05:30
[ ]
PS D:\TechnoWrits\TechnoWrits_My_Docs\Resume\RedHat\Interview_Task\softwares_apiman\wildfly-10.1.0.Final\apiman\apiman-quickstarts\echo-service>
```

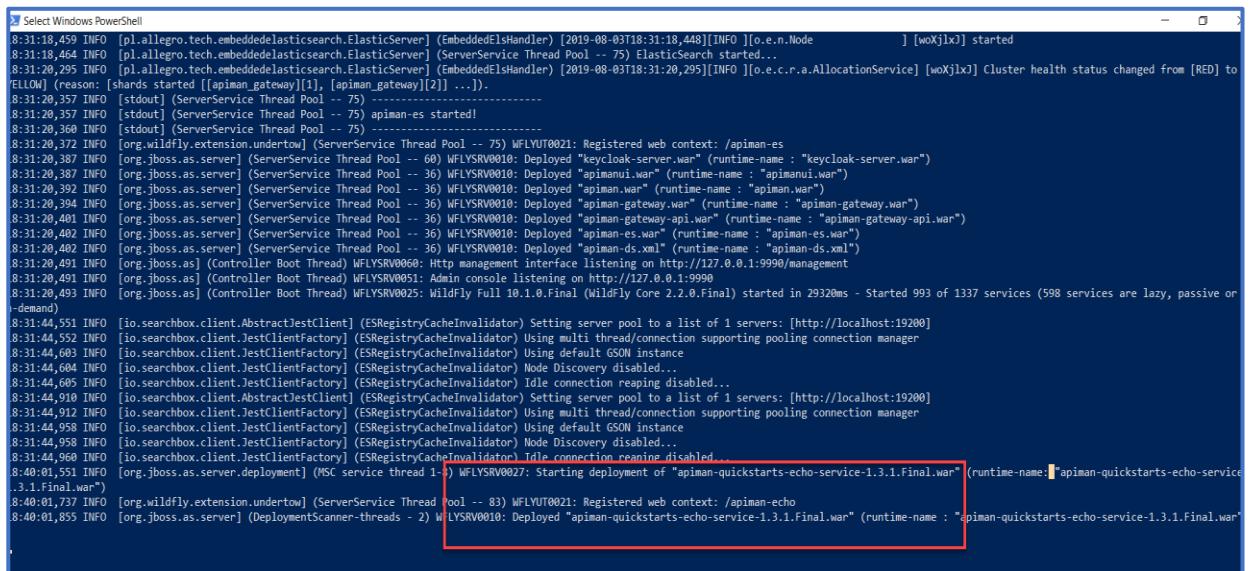
Figure 21: Echo-service build on maven to create war file

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



```

Select Windows PowerShell
8:31:18,459 INFO [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-03T18:31:18,448][INFO ] [o.e.n.Node] [woXjlx] started
8:31:18,464 INFO [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (ServerService Thread Pool -- 75) ElasticSearch started...
8:31:20,295 INFO [pl.allegro.tech.embeddedelasticsearch.ElasticServer] (EmbeddedElsHandler) [2019-08-03T18:31:20,295][INFO ] [o.e.c.r.a.AllocationService] [woXjlx] Cluster health status changed from [RED] to [YELLOW] (reason: [shards started [[apiman_gateway[[1], [apiman_gateway[[2]] ...]]]] ...
8:31:20,357 INFO [stdout] [ServerService Thread Pool -- 75] -----
8:31:20,357 INFO [stdout] [ServerService Thread Pool -- 75] apiman-es started!
8:31:20,360 INFO [stdout] [ServerService Thread Pool -- 75] -----
8:31:20,372 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 75) WFLYUT0021: Registered web context: /apiman-es
8:31:20,387 INFO [org.jboss.as.server] (ServerService Thread Pool -- 60) WFLYSRV0010: Deployed "keycloak-server.war" (runtime-name : "keycloak-server.war")
8:31:20,394 INFO [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apimanui.war" (runtime-name : "apimanui.war")
8:31:20,392 INFO [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman.war" (runtime-name : "apiman.war")
8:31:20,394 INFO [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-gateway.war" (runtime-name : "apiman-gateway.war")
8:31:20,401 INFO [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-gateway-api.war" (runtime-name : "apiman-gateway-api.war")
8:31:20,402 INFO [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-es.war" (runtime-name : "apiman-es.war")
8:31:20,402 INFO [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-ds.xml" (runtime-name : "apiman-ds.xml")
8:31:20,491 INFO [org.jboss.as] (Controller Boot Thread) WFLYRV0005: Admin console listening on http://127.0.0.1:9990
8:31:20,491 INFO [org.jboss.as] (Controller Boot Thread) WFLYRV0005: WildFly Full 10.1.0.Final (WildFly Core 2.2.0.Final) started in 29320ms - Started 993 of 1337 services (598 services are lazy, passive or demand)
8:31:44,551 INFO [io.searchbox.client.AbstractTestClient] (ESRegistryCacheInvalidator) Setting server pool to a list of 1 servers: [http://localhost:19200]
8:31:44,552 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Using multi thread/connection supporting pooling connection manager
8:31:44,603 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Using default GSON instance
8:31:44,604 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Node Discovery disabled...
8:31:44,605 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Idle connection reaping disabled...
8:31:44,910 INFO [io.searchbox.client.AbstractTestClient] (ESRegistryCacheInvalidator) Setting server pool to a list of 1 servers: [http://localhost:19200]
8:31:44,912 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Using multi thread/connection supporting pooling connection manager
8:31:44,958 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Using default GSON instance
8:31:44,958 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Node Discovery disabled...
8:31:44,960 INFO [io.searchbox.client.JestClientFactory] (ESRegistryCacheInvalidator) Idle connection reaping disabled...
8:40:01,551 INFO [org.jboss.as.server.deployment] (MSC service thread 1-1) WFLYSRV0027: Starting deployment of "apiman-quickstarts-echo-service-1.3.1.Final.war" (runtime-name: "apiman-quickstarts-echo-service-1.3.1.Final.war")
8:40:01,737 INFO [org.wildfly.extension.undertow] (ServerService Thread Pool -- 83) WFLYUT0021: Registered web context: /apiman-echo
8:40:01,855 INFO [org.jboss.as.server] (DeploymentScanner-threads - 2) WFLYRV0010: Deployed "apiman-quickstarts-echo-service-1.3.1.Final.war" (runtime-name : "apiman-quickstarts-echo-service-1.3.1.Final.war")
```

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**.

Working With Apiman

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.

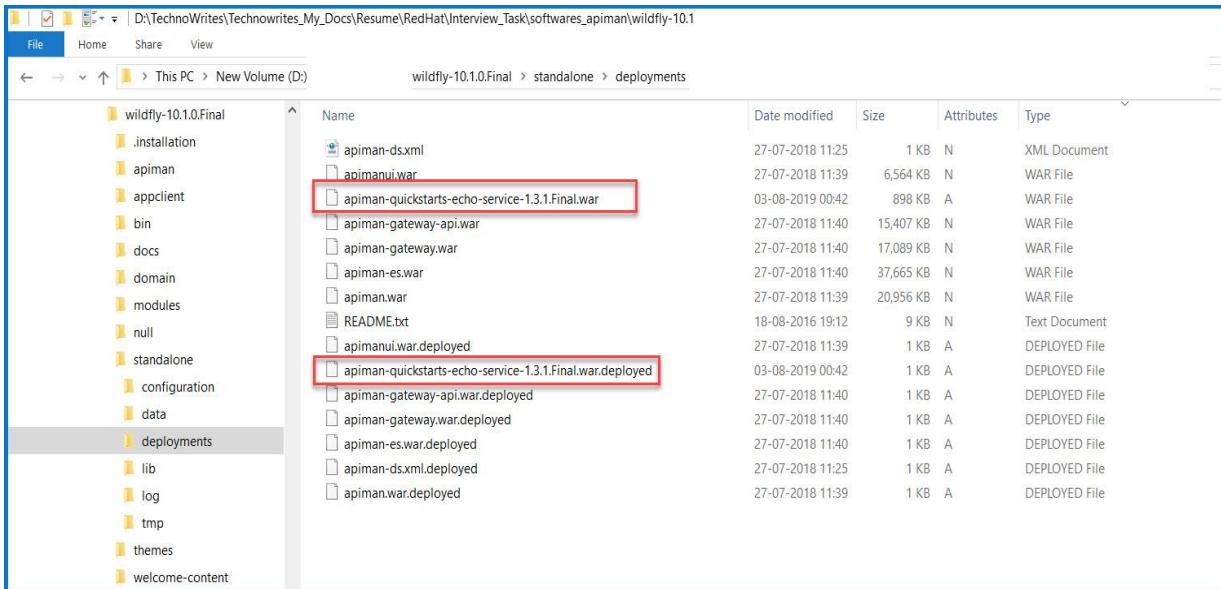


Figure 23: Echo-service deployed on Wildfly

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

```
{
  "method" : "GET",
  "resource" : "/apiman-echo",
  "uri" : "/apiman-echo",
  "headers" : {
    "Cookie" : "__utmz=111872281.1564513919.1.1.utmcsl=(direct)|utmccn=(direct)|utmcmd=(none); __utma=111872281.1759220456.1564513919.156utmb=111872281.2.10.1564837037",
    "Accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3",
    "Upgrade-Insecure-Requests" : "1",
    "User-Agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36",
    "Connection" : "keep-alive",
    "Host" : "localhost:8080",
    "Accept-Language" : "en-GB,en-US;q=0.9,en;q=0.8",
    "Accept-Encoding" : "gzip, deflate, br",
    "DNT" : "1"
  },
  "bodyLength" : null,
  "bodySha1" : null
}
```

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.

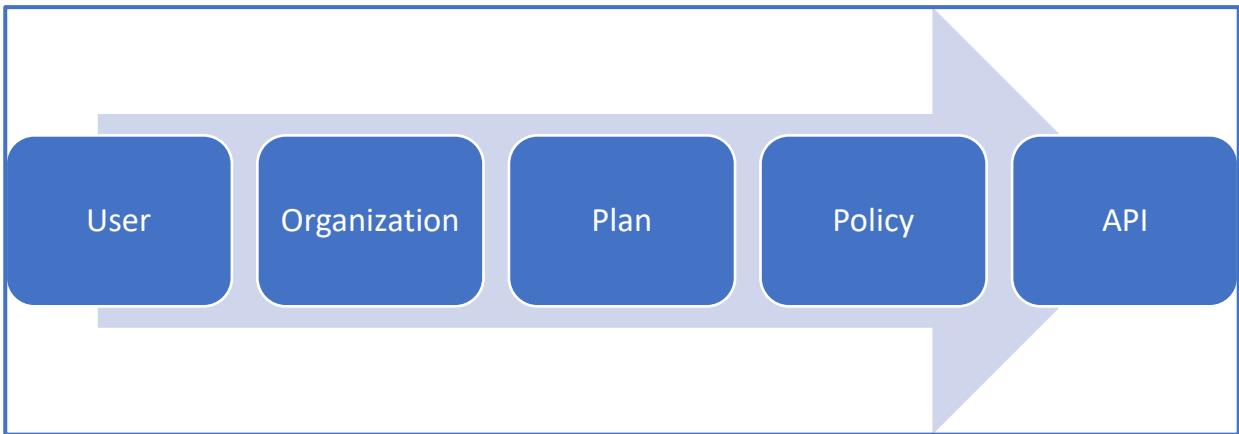


Figure 25: API Provider process flow

- 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**

The screenshot shows the APIMAN API Management UI. The top navigation bar has tabs for 'apiman - Home', 'localhost:8080/apimanui/api-manager/' (selected), and 'dance - Google Search'. The main header says 'APIMAN'. Below it is a 'API Management' section with a welcome message. On the left, there's a sidebar with 'Organizations', 'APIs', and 'Client Apps'. The 'Organizations' section is expanded, showing a list with 'Create a New Organization' (which is highlighted with a red box), 'Go to My Organizations', and 'Browse/Find an Organization'. The 'APIs' section shows links for 'Manage My APIs', 'Create a New API', and 'Browse available/importable APIs'. The 'Client Apps' section shows links for 'Manage My Client Apps', 'Create a New Client App', and 'Find/Consume an API'.

Figure 26: Create Organization link on APiman UI

Enter details of the organization

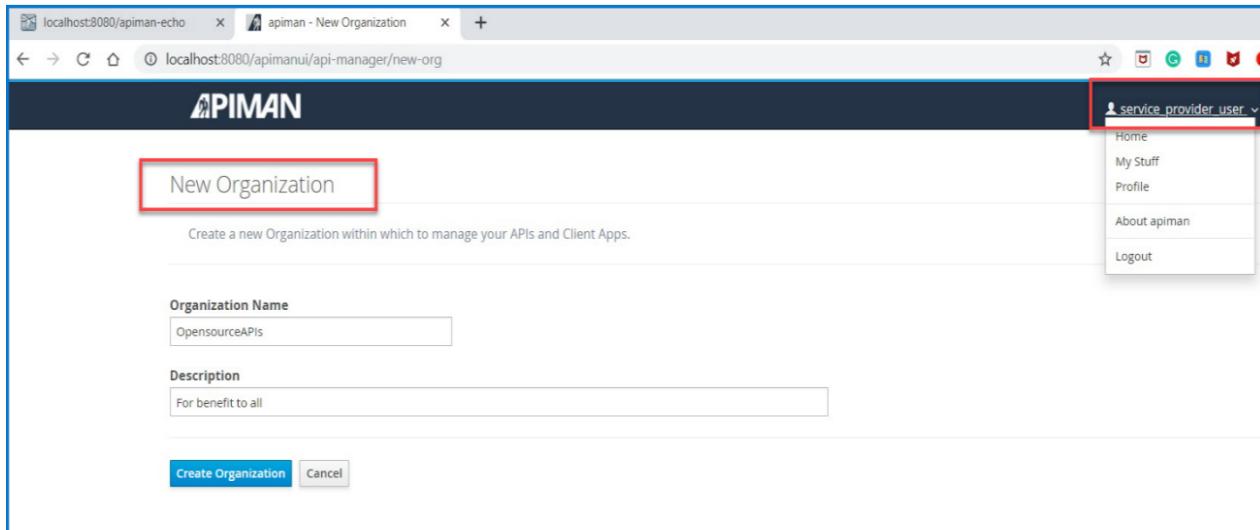


Figure 27: Create Organization for user

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

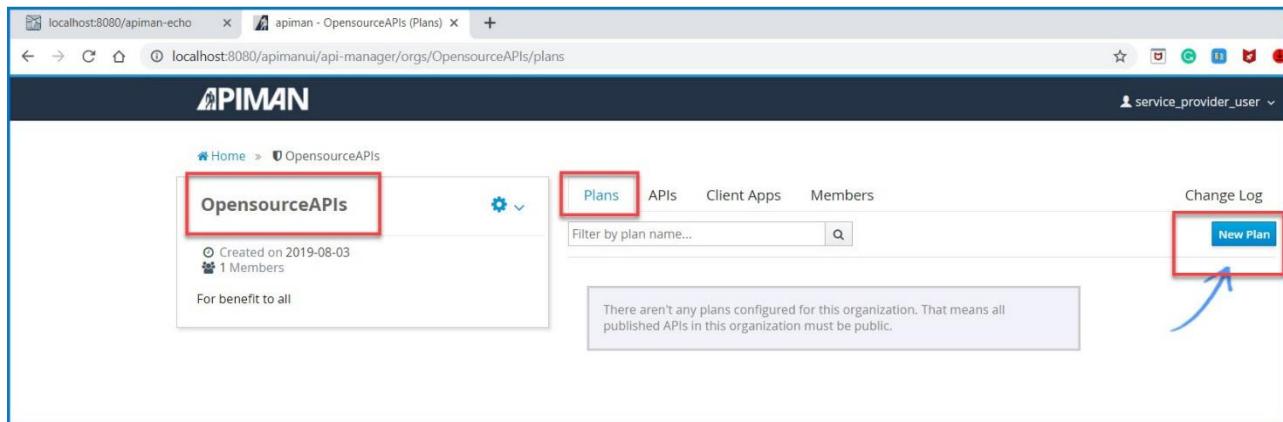


Figure 28: Create Plan

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

The screenshot shows the APIMAN UI for managing API plans. A specific plan, "WelcomeToOpenSourceAPI", is selected. The "Policies" tab is highlighted with a red box. Other tabs visible include "Overview" and "Change Log". On the right, there's a "Plan Details" panel with information about the plan's creation date and status, and a "Lock Plan" button.

Figure 29: Plan with Policies tab

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

The screenshot shows the "Add Policy" page. It includes a brief description of what adding a policy does, a "Policy Type" dropdown menu, and "Add Policy" and "Cancel" buttons.

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:

The image displays two separate views of the "Policy Type" dropdown menu. The left view highlights "BASIC Authentication Policy". The right view highlights "IP Whitelist Policy". Both lists include other policy types such as Authorization, Caching, Ignored Resources, IP Blacklist, Quota, Rate Limiting, Time Restricted Access, Transfer Quota, and URL Rewriting.

Figure 31: Policy Types

Working With Apiman

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

The screenshot shows the 'Add Policy' page in the Apiman UI. At the top, there is a header bar with tabs for 'ian-echo' and 'apiman - Add Policy'. The URL is 'localhost:8080/apimanui/api-manager/orgs/OpensourceAPIs/plans/WelcomeToOpenSourceAPI/1.0/new-policy'. On the right, there is a user dropdown for 'service_provider_user'. The main content area has a title 'Add Policy' and a sub-section 'Time Restricted Access Policy Configuration'. A note says 'Adding a policy will allow its specific functionality to be applied to the API invocation as part of the overall Policy Chain.' A red box highlights the 'Policy Type' dropdown, which is set to 'Time Restricted Access Policy'. Below this, there is a section for 'Time Restricted Access Policy Configuration' with fields for 'Allow all requests matching URI' (containing '/path/to/*'), and time restrictions for 'From time in the day' (08:00), 'To time in the day' (16:00), 'From day of the week' (Mon), and 'To day of the week' (Fri). An 'Add' button is at the bottom right.

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.

The screenshot shows the 'WelcomeToOpenSource' plan page in the Apiman UI. The URL is 'localhost:8080/apimanui/api-manager/orgs/OpensourceAPIs/plans/WelcomeToOpenSourceAPI/1.0/policies'. The main content area has a title 'WelcomeToOpenSourceAPI'. On the left, there is a sidebar with 'Overview', 'Policies' (which is selected and highlighted in blue), and 'Change Log'. In the main area, there is a 'Plan Policies' section with a red box around it. It contains a list of policies: 'Rate Limiting Policy' (created by 'service_provider_user' on 2019-08-03) and 'Time Restricted Access Policy' (created by 'service_provider_user' on 2019-08-03). Both policies have a 'Remove' button next to them. A red box also highlights the 'Lock Plan' button, which is located above the policy list. Other buttons in the top right include 'Version: 1.0', 'New Version', and a gear icon.

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

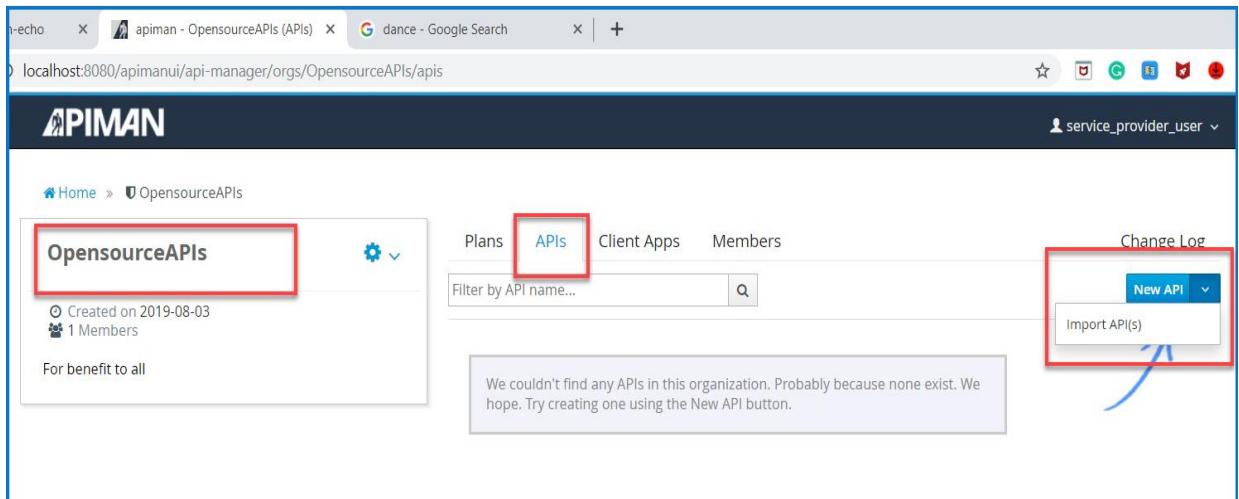


Figure 34: Define API inside Organization

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

Create a new API within the specified Organization, allowing Client Apps to begin consuming it.

Organization	API Name
OpensourceAPIs	Echo-HTTP-Request-Service
Initial Version	1.0
Description	This API will display in response the parameters passed in HTTP requests

Create API **Cancel**

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>

Working With Apiman

The screenshot shows the Apiman API Manager interface for the Echo-HTTP-Request-Service API. The main content area is titled 'Echo-HTTP-Request-Service' and contains the following information:

- Description: This API will display in response the parameters passed in HTTP request.
- Created on: 2019-08-03
- Created by: service_provider_user
- Status: CREATED
- API Implementation section:
 - API Endpoint: http://localhost:8080/apiman-echo (highlighted with a red box)
 - API Type: REST
 - API Content Type: JSON
 - Enable stateful request payload inspection (with a question mark icon)
- Buttons: Publish, Why can't I publish?
- Version: 1.0
- New Version

A sidebar on the left lists navigation options: Overview, Implementation, Definition, Plans, Policies, and Change Log. The 'Implementation' tab is selected.

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.

The screenshot shows the Apiman API Manager interface for the Echo-HTTP-Request-Service API, now published. The main content area is titled 'Echo-HTTP-Request-Service' and contains the following information:

- Description: This API will display in response the parameters passed in HTTP request.
- Created on: 2019-08-03
- Created by: service_provider_user
- Status: PUBLISHED
- Public API section:
 - Make this API public
- Available Plans section:
 - Authentication_Plan (version 1.0)
 - OpensourceAPIPlan2 (version 1.0)
 - WelcomeToOpenSourceAPI (version 1.0)
- Buttons: Link my Client App to this API (New Contract), Create a new version of this API (New Version), Retire
- Version: 1.1
- New Version

A sidebar on the left lists navigation options: Overview, Implementation, Definition, Plans, Policies, Contracts, Endpoint, Metrics, and Change Log. The 'Plans' tab is selected.

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

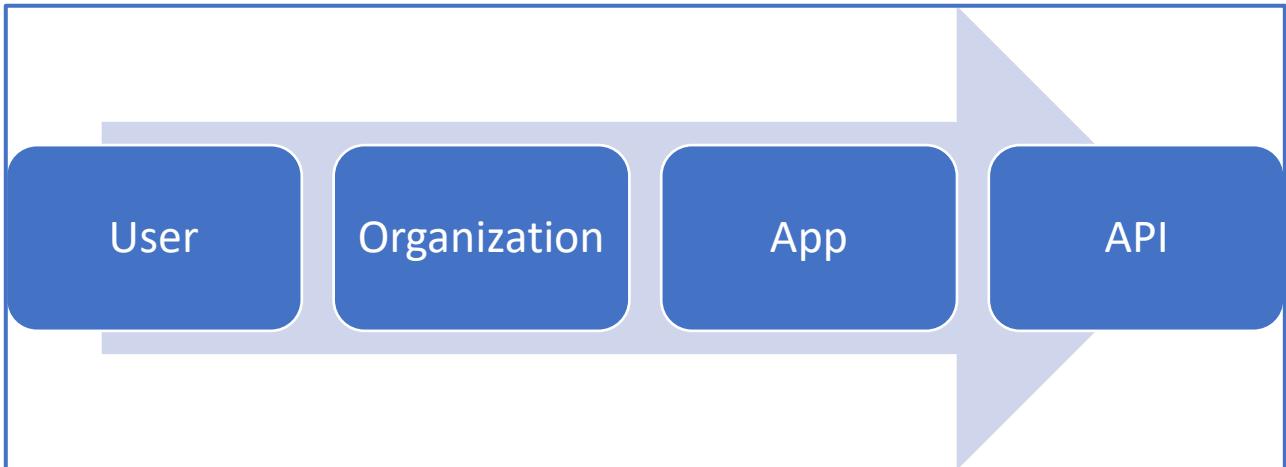


Figure 38: API consumer process flow

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)

The screenshot shows the APIMAN API Management home page. The browser tabs include 'nan-echo', 'apiman - Home', and 'dance - Google Search'. The address bar shows 'localhost:8080/apimanui/api-manager/'. The main header is 'APIMAN' with a dropdown menu showing 'service_consumer_app_developer'. Below the header, there's a 'API Management' section with a welcome message. The page is divided into several sections: 'Organizations' (with links to create a new organization, go to my organizations, or browse/find an organization), 'APIs' (with links to manage my APIs, create a new API, or browse available/importable APIs), and 'Client Apps' (with a red box around it). Under 'Client Apps', there are links to manage my client apps, create a new client app, or find/consume an API. The 'Create a New Client App' link is specifically highlighted with a red box.

Figure 39: Create app

Working With Apiman

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

The screenshot shows the Apiman UI for managing client applications. The main title bar has tabs for 'jan-echo', 'apiman - MyApp (Overview)', and 'dance - Google Search'. The active tab is 'localhost:8080/apimanui/api-manager/orgs/Opensource_my_org/clients/MyApp/1.0'. The top navigation bar includes 'APIMAN' logo, user info ('service_consumer_app_developer'), and dropdowns for 'Version: 1.0', 'New Version', and settings. A red box highlights the search bar labeled 'Search for APIs to consume'. Below it are buttons for 'Create a new API Contract for this Client App' and 'Create a new version of this Client App (New Version)'. On the left, a sidebar menu lists 'Overview', 'Contracts', 'Policies', and 'Change Log', with 'Contracts' being the selected tab. The main content area is titled 'Client App Details' and contains descriptive text about the Client App's purpose and requirements. It also lists 'Created on 2019-08-03' and 'Created by service_consumer_app_developer'. A 'Status: CREATED' button is shown. At the bottom, there are sections for 'Contracts', 'Policies', and 'Change Log'.

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

New Contract

Creating a Contract allows you to connect a Client App to an API via a particular Plan offered by the API. You would want to do this so that your Client App can invoke the API successfully. Note that this is not necessary if the API is public.

From Client App

The Client App that will be used as the source of the new API Contract. Choose one of your available Client Apps below, and then choose a Client App version.

Opensource_my_org / MyApp ▾ 1.0 ▾

Using Plan

Use the drop-down below to choose one of the Plans made available by the selected API.

Authentication_Plain ▾

To API

Use this section to choose what API the Client App will be consuming (aka the "target" of this API Contract).

OpensourceAPIs / Echo-HTTP-Request-Service > 1.1
(click to change)

Create Contract Cancel

Figure 41: New contract

apiman-echo X apiman - MyApp (Contracts) X dance - Google Search X +

localhost:8080/apimanui/api-manager/orgs/Opensource_my_org/clients/MyApp/1.0/contracts

APIMAN service_consumer_app_developer

Home > Opensource_my_org > MyApp

Want to echo on screen

Created on 2019-08-03
Created by service_consumer_app_developer

Status: REGISTERED

Version: 1.0 ▾ New Version ▾

API Contracts

Here is a list of all APIs that this Client App is currently contracted to utilize. This provides a list of all APIs that Client App can potentially invoke.

Filter by org or API name... Q Break All New Contract

OpensourceAPIs / Echo-HTTP-Request-Service
API version 1.0 via plan WelcomeToOpenSourceAPI entered into on 2019-08-03
This API will display in response the parameters passed in HTTP request

Figure 42: Contract created

Working With Apiman

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**

The screenshot shows the Apiman API Manager interface. The top navigation bar has a user dropdown labeled 'service_consumer_app_developer'. The main content area is titled 'API Management'. On the left, there are three sections: 'Organizations', 'Client Apps', and 'APIs'. The 'Client Apps' section is highlighted with a red box and contains the following text: 'Create and manage your Client Apps. A Client App is the thing that consumes APIs. These consumers must be defined in this platform so that contracts can be created between them and the APIs they wish to consume.' Below this text is a list of actions: 'Manage My Client Apps' (with a red box around it and a circled '1'), 'Create a New Client App', and 'Find/Consume an API'. To the right of the 'Client Apps' section, there is another section with a red box around the 'MyApp' link, which is circled with '2'. This section contains text about managing APIs and lists actions: 'Manage My APIs', 'Create a New API', and 'Browse available/importable APIs'.

Figure 43: Manage Client App

- 2) Select the application

The screenshot shows the Apiman API Manager interface. The top navigation bar has a user dropdown labeled 'service_consumer_app_developer'. The main content area is titled 'API Management'. On the left, there is a sidebar for a user named 'ask sk' with the email 'aamolk@gmail.com' and the joining date 'Joined on 2019-08-03'. On the right, there are tabs for 'Organizations', 'Client Apps' (which is selected and highlighted in blue), and 'APIs'. Below the tabs is a search bar with the placeholder 'Filter by org or client name...'. Under the 'Client Apps' tab, there is a list of clients: 'Opensource_my_org' and 'MyApp'. The 'MyApp' link is highlighted with a red box and circled with '2'. A tooltip 'Want to echo on screen' is visible next to the 'MyApp' link.

Figure 44: Select App

3) Select APIs tab in Myapp application page

The screenshot shows the APIMAN interface for the MyApp client application. At the top, there's a navigation bar with Home, Opensource_my_org, and MyApp. Below that is a header for 'MyApp' with a gear icon. The main content area has a sidebar with tabs: Overview, Contracts, Policies, APIs (which is highlighted with a red box and has a red number 3), and Metrics. The main content area is titled 'Client App Details' and contains information about the app's creation date and developer, its status as 'REGISTERED', and buttons for Re-Register and Unregister. A note says 'Want to echo on screen'. To the right, there are links to search for APIs, create contracts, and create new versions. The central part of the page is titled 'Contracts' and describes what it is used for.

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.

This screenshot shows the 'Client App APIs' section of the APIMAN interface for the MyApp client. The APIs tab is selected (highlighted with a red box and marked with a red number 4). It displays a table of APIs consumed by the client. The table has columns for API, Version, and Plan. One row is visible: 'OpensourceAPIs / Echo-HTTP-Request-Service' with Version 1.0 and Plan 'WelcomeToOpenSourceAPI'. To the right of the API name, there's a small blue 'i' icon with a red box around it, indicating where to click to get the URL. The top of the page shows the URL as localhost:8080/apimanui/api-manager/orgs/Opensource_my_org/clients/MyApp/1.0/apis.

Figure 46: Get URL icon

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

Working With Apiman

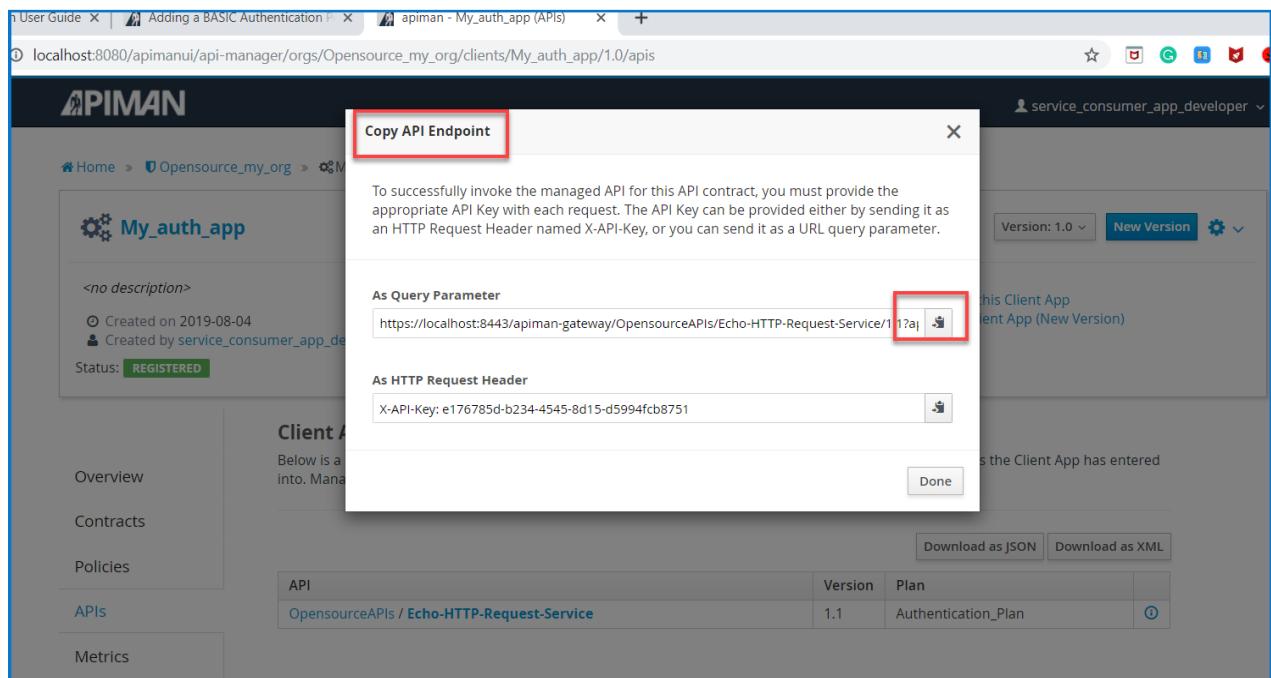


Figure 47: Copy API Endpoint

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

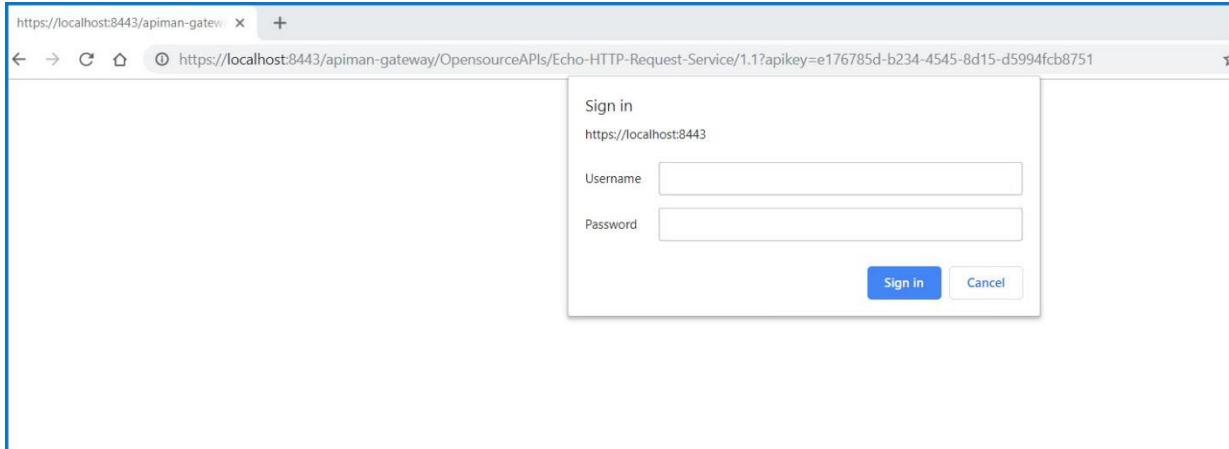
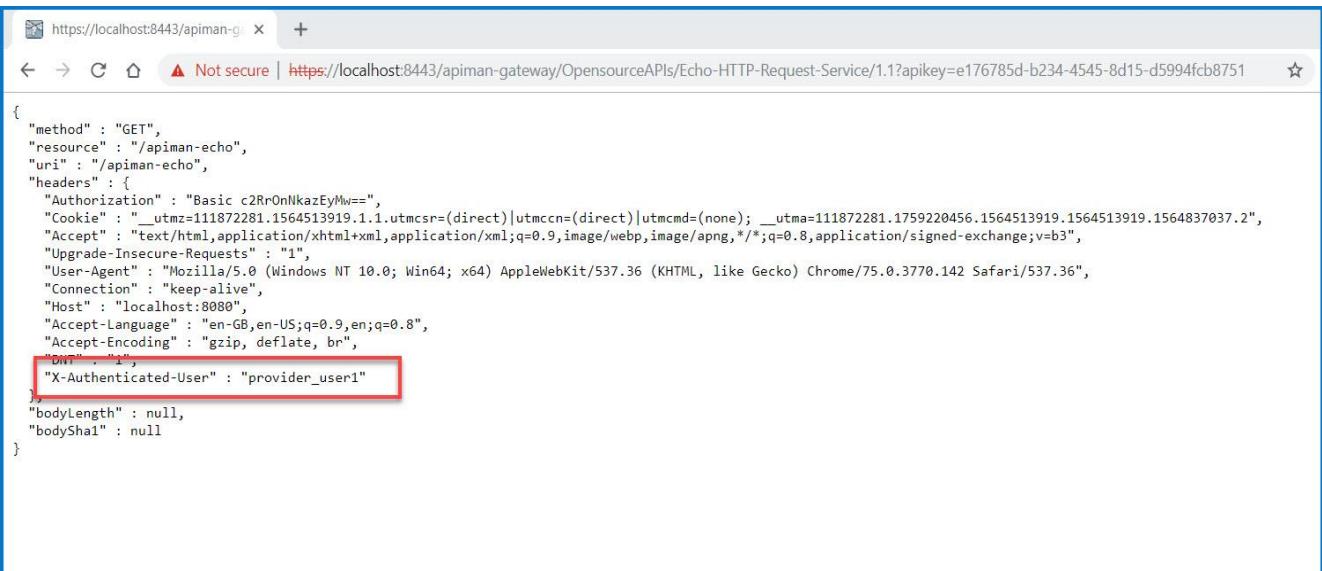


Figure 46: API Authentication

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



```
{  
    "method" : "GET",  
    "resource" : "/apiman-echo",  
    "uri" : "/apiman-echo",  
    "headers" : {  
        "Authorization" : "Basic c2RrOnNkazEyMw==",  
        "Cookie" : "__utmz=111872281.1564513919.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); __utma=111872281.1759220456.1564513919.1564513919.1564837037.2",  
        "Accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3",  
        "Upgrade-Insecure-Requests" : "1",  
        "User-Agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3770.142 Safari/537.36",  
        "Connection" : "keep-alive",  
        "Host" : "localhost:8080",  
        "Accept-Language" : "en-GB,en-US;q=0.9,en;q=0.8",  
        "Accept-Encoding" : "gzip, deflate, br",  
        "DNT" : "1",  
        "X-Authenticated-User" : "provider_user1"  
    },  
    "bodyLength" : null,  
    "bodySha1" : null  
}
```

Figure 49: Managed echo-service API response

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

Index

API, 1
apiman UI, 11
endpoint, 27

Plan, 18
Policy, 18
Start Wildfly web server., 9