2021

APIMAN - Getting Started

Confidential

REDHAT

April/2021

APIMAN - GETTING STARTED

Contents

1. Intended Audience

This Getting Started guide is aimed at the below groups of Apiman users:

- API Providers
- API Consumers
- API Management Administrators
- API Developers

2. What is API Management

API stands for Application Programming Interface. It defines how two pieces of software talk with each other to achieve a common goal. You can define how it is used using an API contract. This enables developers to reuse functionality across applications and domains. A simple example of API usage is using your Facebook login to access a particular game.

However, there are some disadvantages to this approach. There are difficulties in,

- Discovering or sharing existing APIs
- Sharing common functionality across API implementations
- Tracking of API usage/consumption

This is where an effective API management tool comes into the picture. API Management can address these and other issues by providing a tool to manage, configure, track APIs. To apply these configurations during runtime, an API Gateway is used.

In summary, API management provides the below features to developers and consumers. It facilitates easy

- Centralized governance and policy configuration
- API tracking
- API Consumer tracking
- API discovery and sharing
- Leveraging common policy configuration across different APIs

Typical Use Cases

Some common API management use cases include:

Security

You can authenticate the consumers connecting to your API. This can be done by the API itself but an optimized method would be to do it using an API management tool. This also ensures centralizing authentication configuration which can be done at the tool level.

Throttling/Rate Limiting

You can limit your API usage to a particular customer. This can be done by setting a throttling or rate limit at the API Gateway. This ensures a balanced usage of API leading to quicker execution at the consumer level.

Metering/Billing

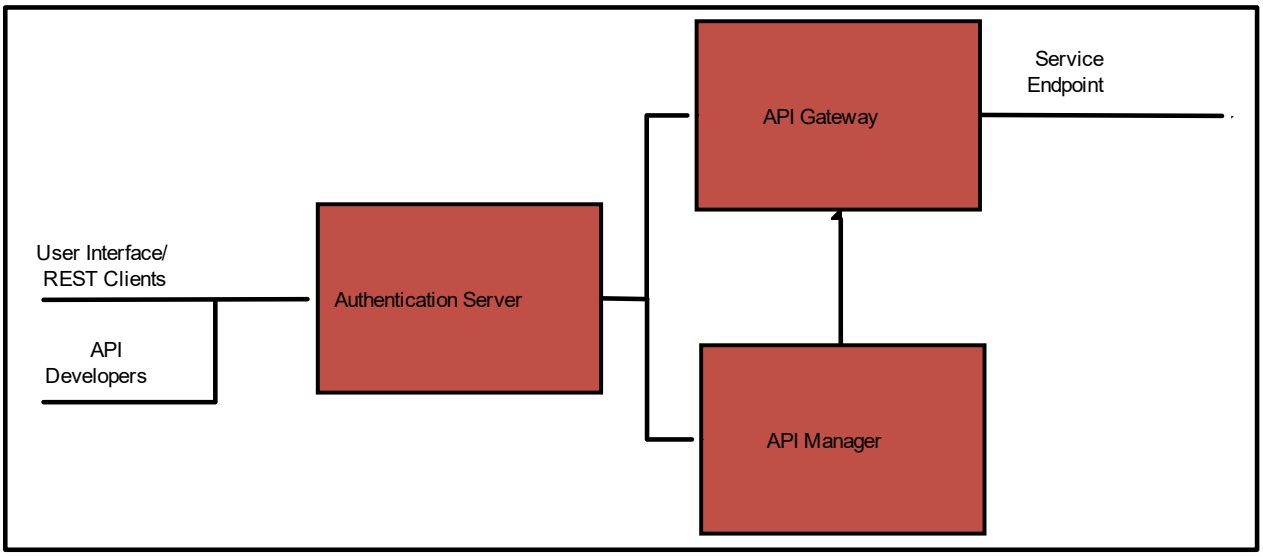
You can implement billing based on your public-facing API usage. API management can help you track this usage.

3. API Man

You can go over some concepts and terminology that are important when using or extending Apiman.

Note: Assumption is that you are familiar with the basics of API Management use cases and typical functionality.

Figure: Architecture of Apiman



Apiman consists of two primary components: the **API Manager**, and **API Gateway**

- The API Manager is a set of REST endpoints and a User Interface. It is used to manage and configure all the APIs being provided and the clients consuming them.
- The API Gateway applies the configurations and policies at runtime to every API request from every registered consumer. This is like a security check at an airport where travelers fulfilling all the security gateway can travel.

API developers can create their custom API management policies and install them into Apiman. Also, custom implementations of core Apiman components can be created and used, without needing to rebuild the core system.

The above can be better understood by reading the following conceptual information.

4. API Manager Concepts

API Manager is where providers go to configure the APIs they would like to manage, and it's also where API consumers go to find APIs to consume.

Organization

Organization entity is a container for everything. A user must be a member of an organization to manage any of the APIs contained within it. You can choose to take up role-based Membership in an Organization. This means different users may be granted different capabilities within it.

Service

When an API developer or manager wants to manage their APIs, they do so by creating Services within the Organization. The Service is given a name, description, and details about the API's

implementation endpoint. The Service can also be given a set of Policies, which are applied by the API Gateway whenever the Service is invoked.

Plan

A plan is a subscription model for consumers. It is a collection of policies that are applied whenever a service is invoked by the consumers utilizing that plan. There can be multiple plans that enable the API providers to have a centralized configuration system for policies. You can configure different access levels, rate limits, etc. using different plans.

Application

APIs can be consumed by creating an application in the API Manager. An Application is typically some kind of mobile app or an integration application. The Application is given a name and a description. Applications can also have configured Policies, which are applied whenever the Application invokes any Service.

Contract

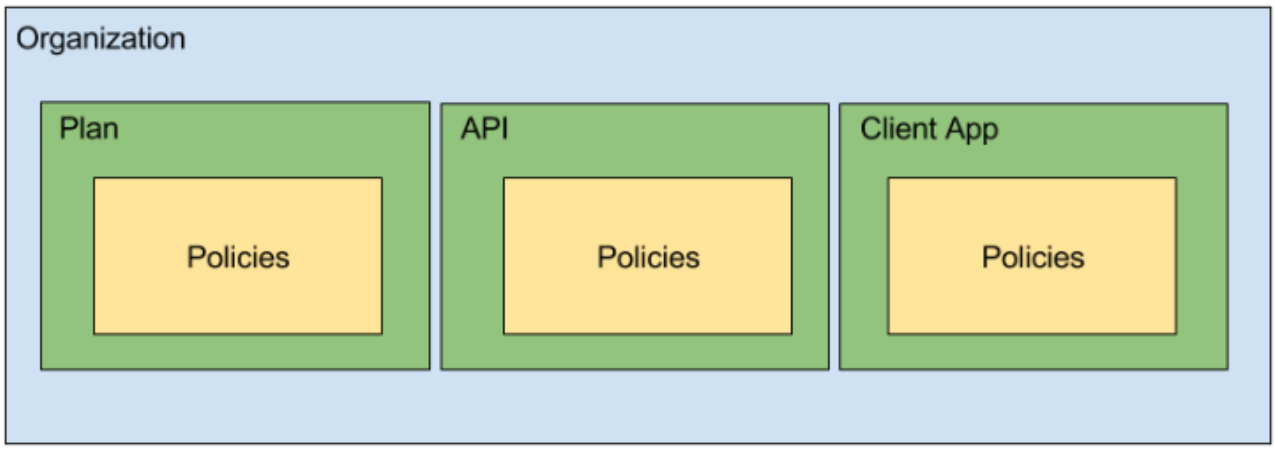
Once an application is created, APIs can be consumed by creating Contracts between the Application and the Services it wishes to invoke. A contract is simply a promise between application and service to perform the services' implementation through one of the Service's available Plans.

Policy

All these concepts are ultimately used to determine which Policies get applied when a particular request is received by the API Gateway. The Policy is the unit of governance executed at runtime and is, therefore, the most important concept. Common examples of Policies include Authentication, Rate Limiting, and IP filtering.

Policies can be configured in three places, Plan, Service, and Application. These Policies are then applied to API requests by the API Gateway at runtime.

Figure 1: API MAN Data Model



5. API Gateway Concepts

Once the API Manager has been used to fully configure a Service or Application, the resulting configuration is published to the API Gateway. This published configuration is used by the API Gateway to apply the Policies to all API Requests.

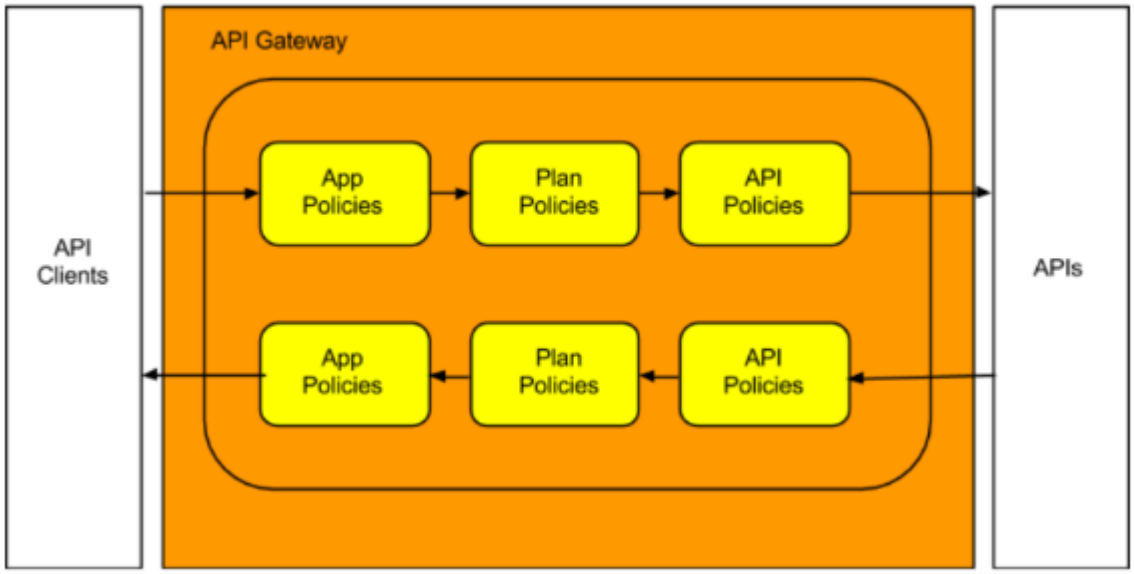
Policy Chain

The Policies configured on the Plan, Service, and Application are aggregated into a Policy Chain which is what gets applied to an API request by the API Gateway.

When an API request is received by the Gateway, the following steps are taken:

1. First, the Policy Chain is applied to the API request. The Application Policies are applied first, followed by the Plan Policies and then the Service Policies.
2. If any of the Policies reject the API request, an error response is sent to the client.
3. If all of the Policies pass, the Gateway forwards the request to the API back-end implementation.
4. Authentication is in form of an API key.
5. Once the back-end responds, the Policy Chain is then applied again, in reverse order, to that response.
6. If all the Policies pass, the response is sent back to the original client.

Figure 2: Apiman 2-way Policy Chain



In summary,

- API Manager - The API Manager provides an easy way for API providers to use a web UI to define **Plans** for their APIs, apply these plans across multiple APIs, and control **Role-Based** user access and API versioning. These plans can govern access to APIs and limits on the rate at which consumers can access APIs. The same UI enables API consumers to easily locate and access APIs. All features available in the web UI are also available via a REST interface, allowing full automation.
- API Gateway - The gateway applies the **Policies** configured in the API Manager, enforcing them as runtime rules for each managed API request made. The way that the API Gateway works is that the consumer of the API accesses the API through a URL that designates the API Gateway as a proxy for the API. If the policies defined to govern access to the API permit that access, the API Gateway then proxies requests to the backend API implementation.

- An Extensible Plugin-Based Architecture - API Developers can create their custom API management policies and install them into Apiman. Also, custom implementations of core Apiman components can be created and used, without needing to rebuild the core system.

6. Installing Apiman

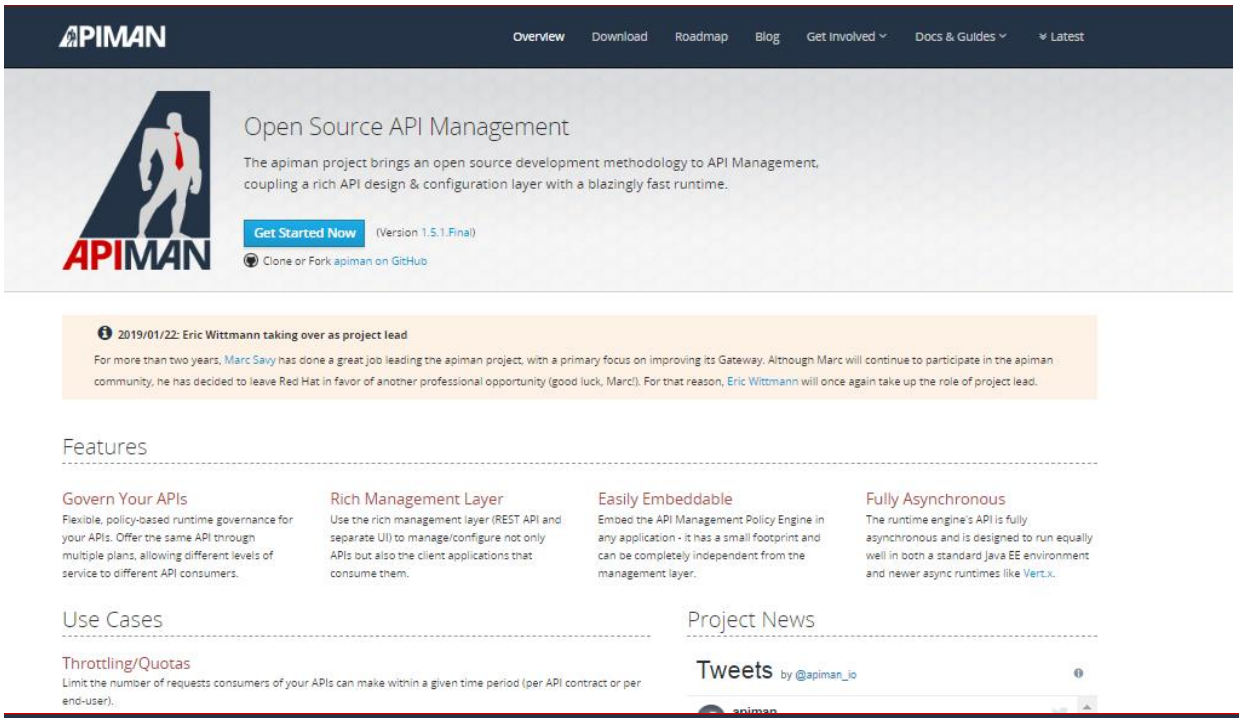
You need to make sure that you have a few things already installed. You can run Apiman on any operating system that supports Java software development. Every component can be downloaded.

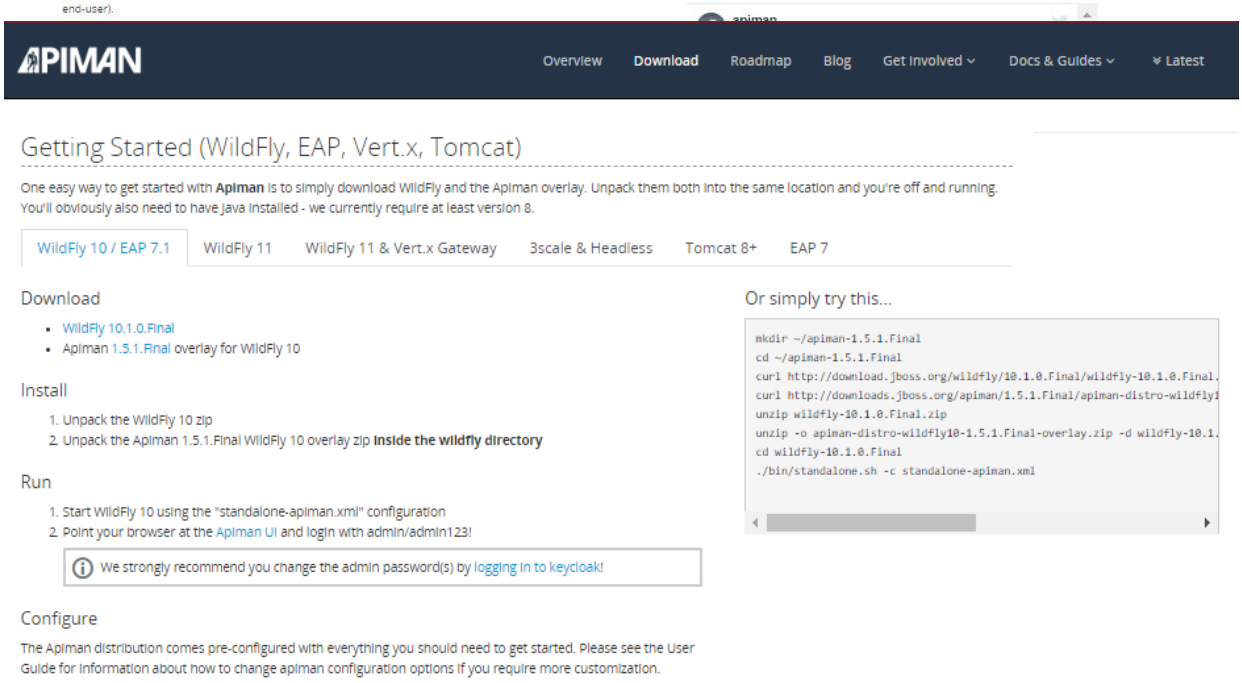
Prerequisites for installation of Apiman:

- Java runtime version 7 or later
- Some sort of unzip utility so you can unpack the downloaded distributions.
- Java Version 1.8 or later
- Java JDK like OpenJDK or Oracle's JDK
- Build tool like Apache Maven Version 3.3 or later
- Git

Downloading Apiman

Now you can start downloading a couple of zip distributions. Both of these distributions can be found on the Apiman project site, on the Download page - http://www.apiman.io/latest/download.html.





1. Download WildFly 10.0.0 Final or later, which is the platform on which you are going to run Apiman. Here you are shown installing Apiman on Wildfly 10.0.0. If the above link doesn't work use - http://download.jboss.org/wildfly/10.1.0.Final/wildfly-10.1.0.Final.zip to download it directly
2. The second is Apiman 1.5.1 Final. If the above link doesn't work use - http://downloads.jboss.org/apiman/1.2.9.Final/apiman-distro-wildfly10-1.2.9.Final-overlay.zip to download directly

RESULT: You have downloaded Apiman and the server to run it.

Contents of Wildfly overlay

The contents of Wildfly 10.0.0 overlay.

The Apiman directory - This directory contains configuration data specific to Apiman such as the DDL (Data Description Language) files that define database schemas used by Apiman, JSON files that define policy and security settings, and a quickstart example program. The Apiman directory is a new directory that is created when you unzip the WildFly Overlay file. The top-level directories in the Apiman directory look like this:

```
├── apiman
│   ├── data
│   │   ├── all-policyDefs.json
│   │   └── apiman-realm.json
│   ├── ddls
│   │   ├── apiman_mysql5.ddl
│   │   └── apiman_postgresql9.ddl
│   ├── quickstarts
│   │   ├── echo-service
│   │   ├── LICENSE
│   │   ├── pom.xml
│   │   └── README.md
│   └── sample-configs
│       ├── apiman-ds_mysql.xml
│       └── apiman-ds_postgresql.xml
```

The modules directory - This directory contains configuration files, including Keycloak (URL) configuration files that are added to the WildFly server for Apiman. These files are added to the WildFly standalone server configuration. The top levels in this directory look like this:

```
├── modules
│   └── system
│       └── layers
└── standalone
    ├── configuration
    │   ├── apiman.jks
    │   ├── apiman.properties
    │   ├── keycloak-server.json
    │   ├── providers
    │   ├── standalone-apiman.xml
    │   ├── standalone-keycloak.xml
    │   └── themes
    ├── data
    │   ├── es
    │   ├── h2
    │   └── keycloak.h2.db
```

The deployments directory - This directory contains the Apiman API Gateway, back end APIs, and Apiman Management UI, packaged as .war files. By unzipping the WildFly Overlay file, these .war files are deployed to the WildFly server. The top levels in this directory look like this:

```
├── deployments
    ├── apiman-ds.xml
    ├── apiman-es.war
    ├── apiman-gateway-api.war
    ├── apiman-gateway.war
    ├── apimanui.war
    └── apiman.war
```

Unpacking the zip files

Download both the Wildfly 10.0.0 and Apiman 1.5.1 zip files

1. Unpack WildFly 10.0.0 Final into the directory where you want to run the server.
2. Unpack Apiman 1,5,1 Final distribution inside the directory that was created when you unzipped WildFly directory. This installs Apiman into the WildFly server.

All that's left to do is startup WildFly.

Installing Apiman on Wildfly server

The commands that you need to execute to install the server are:

1. Invoke the Command Prompt using the following
   a. In MAC OS 10.14, go to Applications **>** Run Utilities folder.
   b. In Windows press Win + R **>** Type 'Cmd'.
   c. In Linux OS, press Alt+Ctrl+T or type 'terminal'.
2. Create a new directory and go to that directory.
   mkdir ~/apiman-1.2.9.Final
   cd ~/apiman-1.2.9.Final
3. Move the contents of url http://download.jboss.org/wildfly/10.1.0.Final/wildfly-10.1.0.Final.zip to the directory wildfly-10.1.0.Final.zip.
   curl http://download.jboss.org/wildfly/10.1.0.Final/wildfly-10.1.0.Final.zip -o wildfly-10.1.0.Final.zip
4. Move the contents of url http://downloads.jboss.org/apiman/1.2.9.Final/apiman-distro-wildfly10-1.2.9.Final-overlay.zip to the directory Apiman-distro-wildfly10-1.2.9.Final-overlay.zip.
   curl http://downloads.jboss.org/apiman/1.2.9.Final/apiman-distro-wildfly10-1.2.9.Final-overlay.zip -o apiman-distro-wildfly10-1.2.9.Final-overlay.zip
5. Unzip the Wildfly server and Apiman directories.
   unzip wildfly-10.1.0.Final.zip
   unzip -o apiman-distro-wildfly10-1.2.9.Final-overlay.zip -d wildfly-10.1.0.Final
6. Create a server user, so that you can log into the server administrative console. This is necessary as WildFly does not come pre-installed with any users.
   cd apiman-1.2.9.Final/wildfly-10.1.0.Final/bin
   ./add-user.sh
7. Select Management user when prompted for the type of user.
   What type of user do you wish to add? a) Management User (mgmt-users.properties) b) Application User (application-users.properties) (a):
8. Define a username and password.
9. Complete creation of user account by taking default values or selecting Yes.

RESULT: You have installed Apiman on Wildfly 10 server.

Running Apiman

The overlay includes not only the apiman binaries but also configuration files and a pre-configured H2 database appropriate for getting started quickly. In particular, an Apiman-specific version of WildFly's standalone.xml config file is provided, which is what you should use when starting up WildFly.

1. Go to the WildFly directory and start it up using the Apiman config:
   cd apiman-1.2.9.Final/wildfly-10.1.0.Final
   ./bin/standalone.sh -c standalone-apiman.xml
2. View the below message to know that the server is up and running. The server also creates a log file with these messages.
   "apiman-gateway.war")23:28:49,091 INFO [org.jboss.as]
   (Controller Boot Thread) WFLYSRV0060: Http management interface listening on http://127.0.0.1:9990/management
   23:28:49,091 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin console listening on http://127.0.0.1:999
   0
   23:28:49,091 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: WildFly Full 10.1.0.Final (WildFly Core 2.0.10.Fi
   nal)

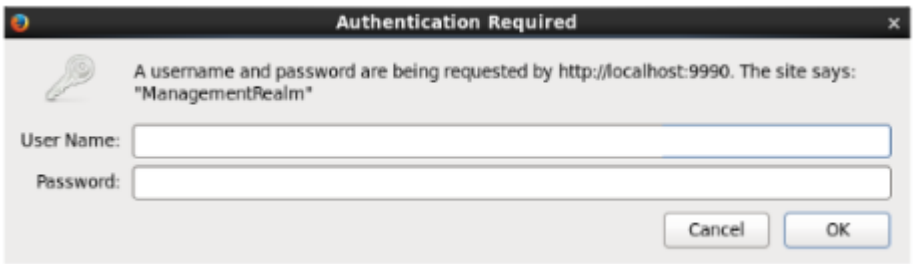
started in 11891ms -
Started 1131 of 1543 services (616 services are lazy, passive or on-demand)

Testing Installation

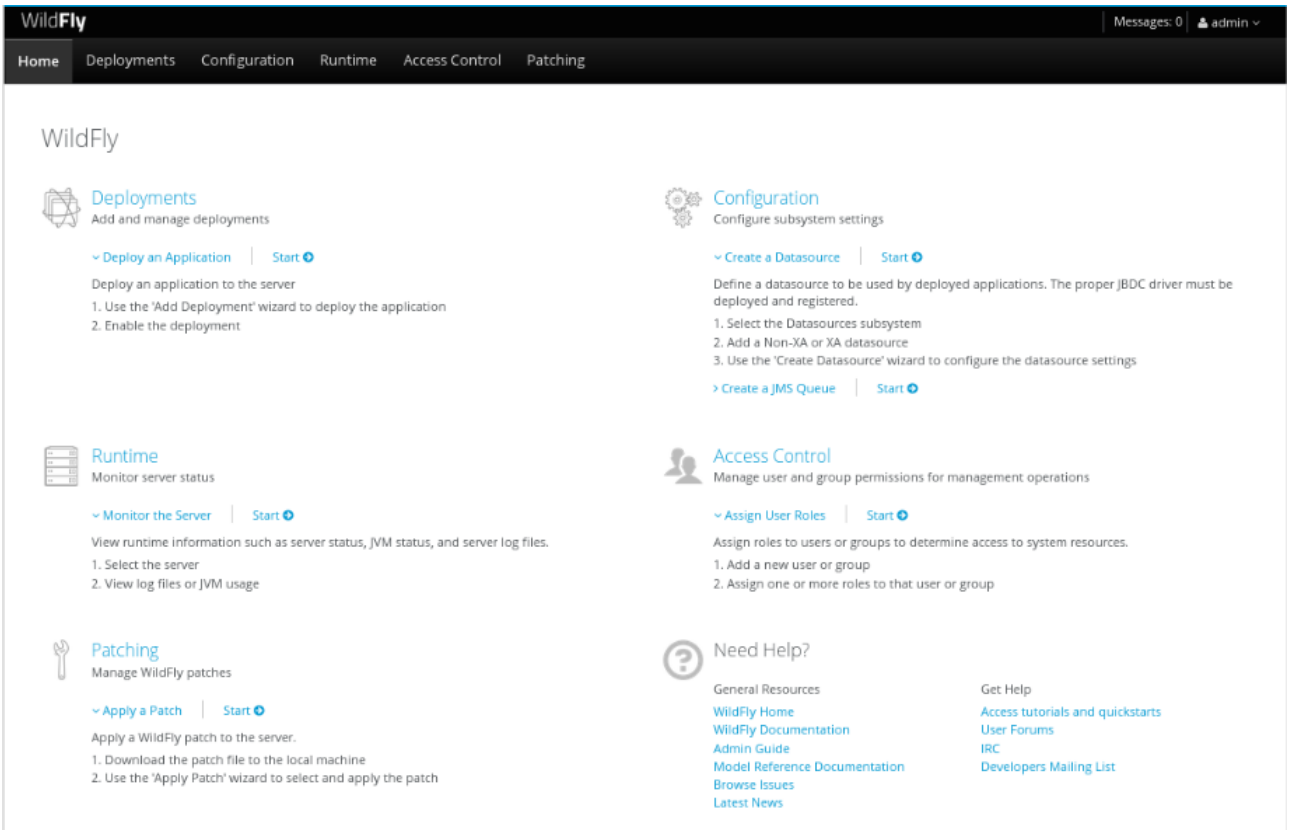
Test if the installation of the Wildfly server is successful.

1. Log in to the Apiman user interface by pointing your browser to localhost:8080/apiman-manager/.
   You'll need to give WildFly a few seconds to startup. The below welcome screen appears:

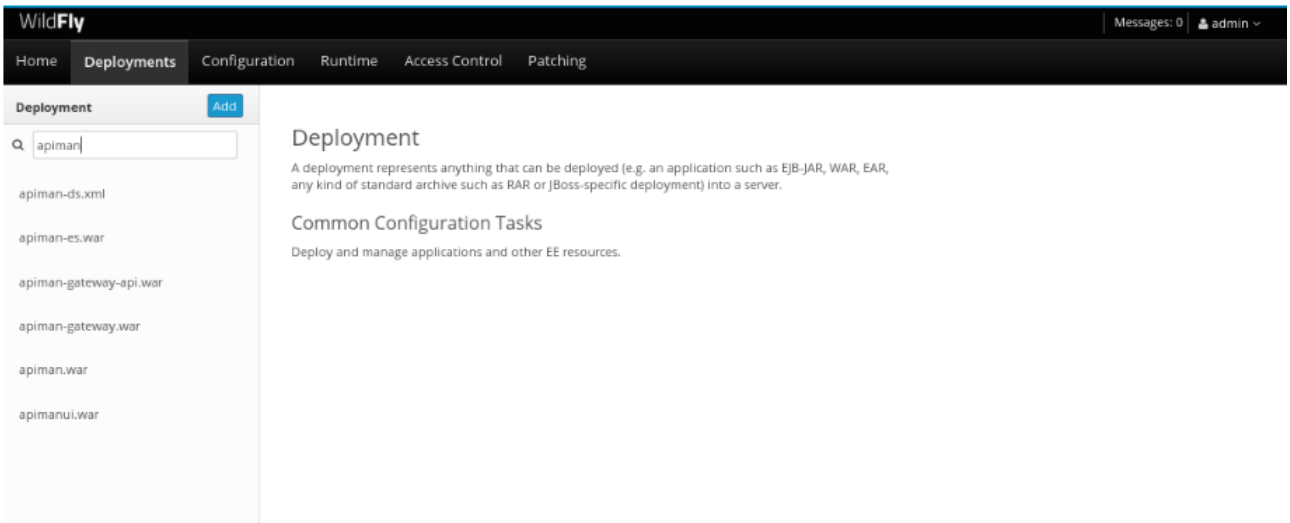
2. Select the Administration Console selection, you will be prompted for the username and password.
   The authentication dialog appears:



3. Enter the user credentials that you had set during Apiman installation.
   The Wildfly Server Administration Console screen appears if you have entered the right credentials:



4. Click the Deployments tab at the top of the page, you'll see the applications deployed to the server. This is where you should see the Apiman deployments for the APIs, Gateway, and Management UI.
   The Deployment screen appears:



5. Change the password to a production appropriate one by logging in to the KeyCloak admin console -localhost:8080/auth/admin/

RESULT: The Wildfly server and Apiman are successfully installed.

7. Troubleshoot Installation

You can view the basic troubleshooting processes involved in the Wildfly server and Apiman installation.

Fixing Apiman and Wildfly server Installation

If you don't see the Apiman deployments in Wildfly Administration Console **>** Deployments, there might be some problem with the installation. The most common reason for the Apiman deployments to be missing is that you unzipped the Apiman overlay.zip file into a directory different from the WildFly server. Below are some of the steps to troubleshoot the installation.

1. Confirm the unzipped Apiman contents by looking in the WildFly server's deployment directory here: **wildfly-10.1.0.Final/standalone/deployments**
   You should see these files with the **.deployed** suffix indicating that the corresponding file was deployed successfully

   ```
   apiman-ds.xml
   apiman-ds.xml.deployed
   apiman-es.war
   apiman-es.war.deployed
   apiman-gateway-api.war
   apiman-gateway-api.war.deployed
   apiman-gateway.war
   apiman-gateway.war.deployed
   apimanui.war
   apimanui.war.deployed
   apiman.war
   apiman.war.deployed
   ```

2. Check WildFly server's server.log file at - **wildfly-10.1.0.Final/standalone/log/server.log**. Look for lines like the below to ensure that the server started cleanly:

   ```
   23:28:48,978 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -
   - 71) WFLYUT0021: Registered web context: /apiman-es
   23:28:49,000 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-gateway-
   api.war" (runtime-name : "apiman-gateway-api.war")
   23:28:48,999 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 60) WFLYSRV0010: Deployed "keycloak-
   server.war" (runtime-name : "keycloak-server.war")
   23:28:49,000 INFO  [org.jboss.as.server] (ServerService Thread Pool -
   - 36) WFLYSRV0010: Deployed "apiman.war" (runtime-name : "apiman.war")
   23:28:49,000 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-
   es.war" (runtime-name : "apiman-es.war")
   23:28:49,001 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-
   ds.xml" (runtime-name : "apiman-ds.xml")
   23:28:49,001 INFO  [org.jboss.as.server] (ServerService Thread Pool -
   - 36) WFLYSRV0010: Deployed "apimanui.war" (runtime-name : "apimanui.war")
   23:28:49,001 INFO  [org.jboss.as.server] (ServerService Thread Pool -
   - 36) WFLYSRV0010: Deployed "services.war" (runtime-name : "services.war")
   23:28:49,001 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "authtest-
   ds.xml" (runtime-name : "authtest-ds.xml")23:28:49,001 INFO
   [org.jboss.as.server] (ServerService Thread Pool -- 36) WFLYSRV0010: Deployed "apiman-gateway.war" (runtime-name :
   ```

If the above-mentioned files are not present in the Wildfly server deployment directory or if the above lines are not present in the log file, stop the server and start the installation over. Unzip the Apiman overlay file directly into the directory created when you unzipped the WildFly server .zip file.

8. Echo API Quickstart

Here you can view how to build and install an Echo API which simply sends back a JSON payload containing all of the meta-data sent to it in the request. Then create provider and user, configure the API and execute a basic limiting policy.

You can get the source code of Echo API Quickstart service in a git repo (http://git-scm.com) hosted at GitHub (https://github.com/Apiman). To download a copy, navigate to the directory in which you want to build the service and execute the below git command:

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

Once the code is downloaded, you can see the below message:

```
git clone git@github.com:apiman/apiman-quickstarts.git
Initialized empty Git repository in apiman-quickstarts/.git/
remote: Counting objects: 104, done.
remote: Total 104 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (104/104), 18.16 KiB, done.
Resolving deltas: 100% (40/40), done.
```

The source code Echo API Quickstart is provided in the **wildfly-10.1.0.Final/apiman/quickstarts** directory.

NOTE: In JBoss software, the term quickstart refers to an example program.

The echo-API quickstart includes these files:

```
└── apiman-quickstarts
        ├── echo-service
        │   ├── pom.xml
        │   ├── README.md
        │   └── src
        │       └── main
        │               ├── java
        │               │   └── io
        │               │       └── apiman
        │               │               └── quickstarts
        │               │               └── echo
        │               │                       ├── EchoResponse.java
        │               │                       └── EchoServlet.java
        │               └── webapp
        │               └── WEB-INF
        │                       ├── jboss-web.xml
        │                       └── web.xml
        ├── LICENSE
        ├── pom.xml
        ├── README.md
        ├── release.sh
        └── src
        └── main
        └── assembly
                └── dist.xml
```

The only action that the Echo API Quickstart performs is to respond to the metadata in the REST requests that it receives as a response.

Building and Deploying API Provider

Maven is used to build the API into a deployable .war file.

1. Type the below command to navigate to the directory into which you downloaded the API example.
   cd apiman-quickstarts/echo-service
2. Type the below maven command to build the .war file.
   mvn package
3. View the below as the build is taking place.
   [INFO] Scanning for projects...
   [INFO]
   [INFO] ----------

```
[INFO] Building apiman-quickstarts-echo-service 1.2.4-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.7:resources (default-resources) @ apiman-quickstarts-echo-service ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory  local/redhat_git/apiman-quickstarts/echo-service/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.2:compile (default-compile) @ apiman-quickstarts-echo-service ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to  local/redhat_git/apiman-quickstarts/echo-service/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.7:testResources (default-testResources) @ apiman-quickstarts-echo-service ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory  local/redhat_git/apiman-quickstarts/echo-service/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.2:testCompile (default-testCompile) @ apiman-quickstarts-echo-service ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ apiman-quickstarts-echo-service ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-war-plugin:2.5:war (default-war) @ apiman-quickstarts-echo-service ---
[INFO] Packaging webapp
[INFO] Assembling webapp [apiman-quickstarts-echo-service] in [ local/redhat_git/apiman-quickstarts/echo-
service/target/apiman-quickstarts-echo-service-1.2.4-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [ local/redhat_git/apiman-quickstarts/echo-service/src/main/webapp]
[INFO] Webapp assembled in [37 msecs]
[INFO] Building war:  local/redhat_git/apiman-quickstarts/echo-service/target/apiman-quickstarts-echo-service-1.2.4-
SNAPSHOT.war
[INFO]
[INFO] --- maven-source-plugin:2.4:jar-no-fork (attach-sources) @ apiman-quickstarts-echo-service ---
[INFO] Building jar:  local/redhat_git/apiman-quickstarts/echo-service/target/apiman-quickstarts-echo-service-1.2.4-
SNAPSHOT-sources.jar
[INFO]
[INFO] --- maven-javadoc-plugin:2.10.1:jar (attach-javadocs) @ apiman-quickstarts-echo-service ---
[INFO]
Loading source files for package io.apiman.quickstarts.echo...
[INFO] Building jar:  local/redhat_git/apiman-quickstarts/echo-service/target/apiman-quickstarts-echo-service-1.2.4-
SNAPSHOT-javadoc.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.061 s
[INFO] Finished at: 2016-04-16T22:13:10-04:00
[INFO] Final Memory: 26M/307M
[INFO] ------------------------------------------------------------------------
```

NOTE: You can see the location of the .war file at the end of the output.

local/redhat_git/apiman-quickstarts/echo-service/target/apiman-quickstarts-echo-service-1.2.4-SNAPSHOT.war

4. Copy the .war file to WildFly server's deployments directory.
   You can see the below message after successfully copying the .war file

```
22:33:59,794 INFO  [org.jboss.as.repository] (DeploymentScanner-threads - 1)
WFLYDR0001: Content added at location local/redhat_git/apiman/tools/server-all/target/wildfly-
10.1.0.Final/standalone/data/content/31/f9a163bd92c51daf54f70d09bff518c2aeef7e/content
22:33:59,797 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-6)
WFLYSRV0027: Starting deployment of "apiman-quickstarts-echo-service-1.2.4-SNAPSHOT.war" (runtime-name: "apiman-
quickstarts-echo-service-1.2.4-SNAPSHOT.war")
22:33:59,907 INFO  [org.wildfly.extension.undertow] (ServerService Thread Pool -
- 76) WFLYUT0021: Registered web context: /apiman-echo
```

```
22:33:59,960 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 1)
WFLYSRV0010: Deployed "apiman-quickstarts-echo-service-1.2.4-SNAPSHOT.war" (runtime-name : "apiman-quickstarts-echo-service-1.2.4-SNAPSHOT.war")
```

5. The URL for the Echo API is generated as - http://localhost:8080/apiman-echo

RESULT: You have installed the Echo API.

Installing and Configuring API Consumer

You can use a browser as a client to access the Echo API.

1. Enter the API's URL - http://localhost:8080/apiman-echo into a browser.
   An HTTP GET command is executed. The response is as shown below:

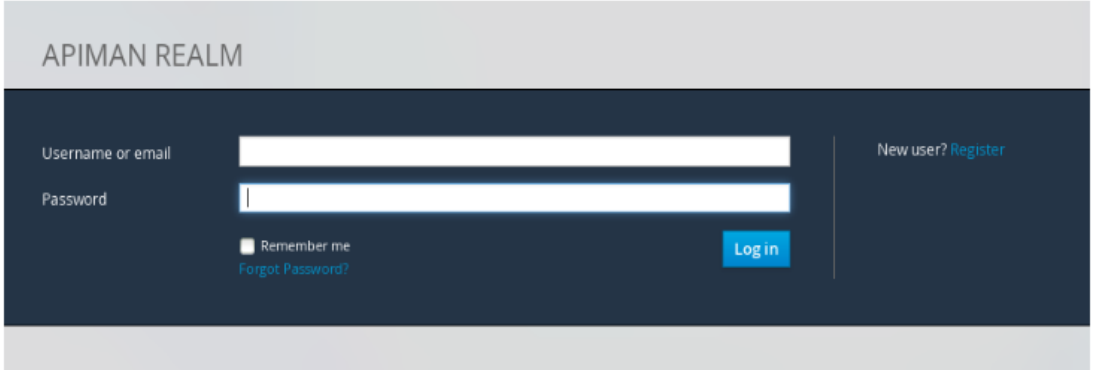

```
{
 "method" : "GET",
 "resource" : "/apiman-echo",
 "uri" : "/apiman-echo",
 "headers" : {
  "Cookie" : "s_fid=722D028B20E49214-13EAE1456E752098; __utma=111872281.807845787.1452188093.1460777731.1460777731.4; __utmz=111872281.1452188093.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(none); _ga=GA1.1.807845787.1452188093; __qca=P0-404983419-1452188093717; __utmc=111872281",
  "Accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
  "Connection" : "keep-alive",
  "User-Agent" : "Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0",
  "Host" : "localhost:8080",
  "Accept-Language" : "en-US,en;q=0.5",
  "Accept-Encoding" : "gzip, deflate",
  "DNT" : "1"
 },
 "bodyLength" : null,
 "bodySha1" : null
}
```


RESULT: You have configured a browser as a simple client for the Echo API.

Creating Users for API Provider and Consumer Organizations

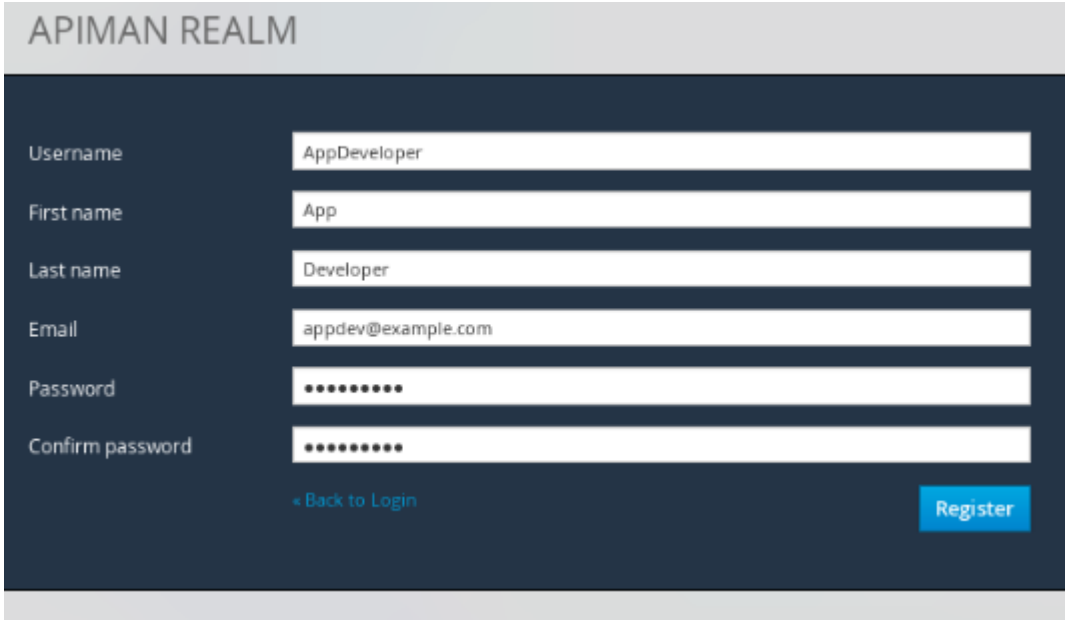
This is the first step in configuring the API manager.

1. Logout from the admin account in the API Manager UI.
   The login dialog appears:



2. Select New user? Register from the dialog and register **API provider** user.

3. Log out and register new **application developer** users too.

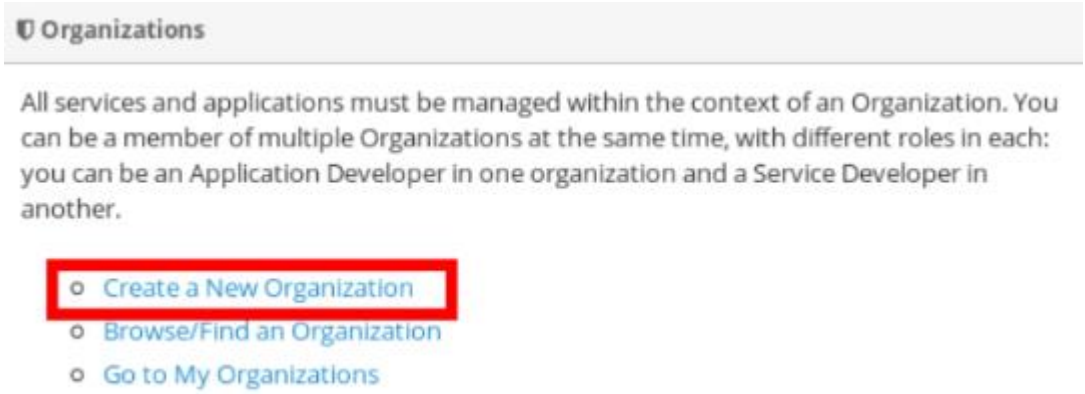


RESULT: You have created a new API provider and user.

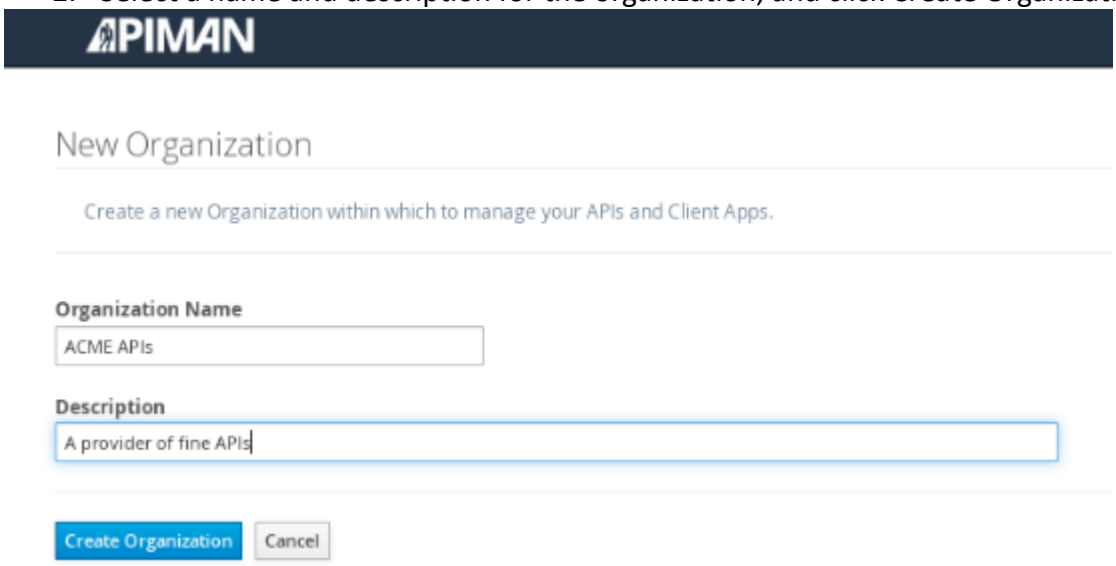
Creating the API Provider Organization

After creating the user and provider, you can proceed to create a new organization as a Service Provider user. Then create all of the entities necessary to publish a service to the API Gateway for consumption. This includes a Plan and the Service itself.

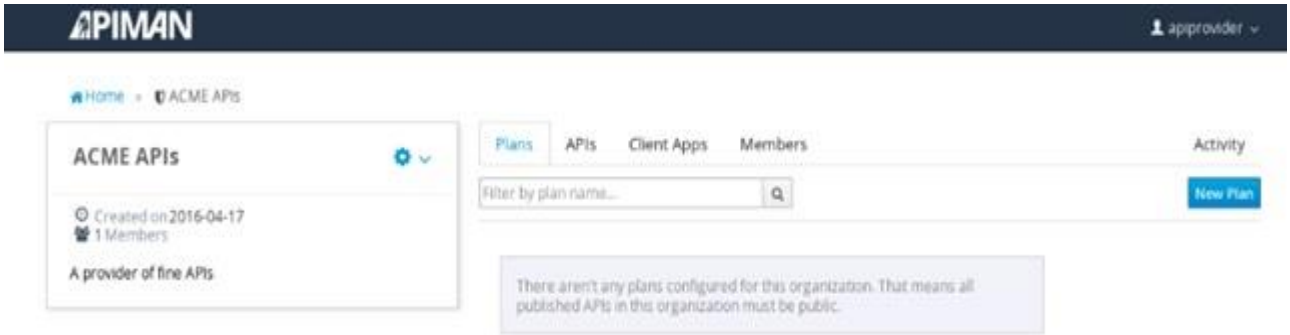
1. Log in to API Manager UI as the **apiprov** user and select Create a new Organization.



2. Select a name and description for the organization, and click Create Organization.



RESULT: You have created a new organization.

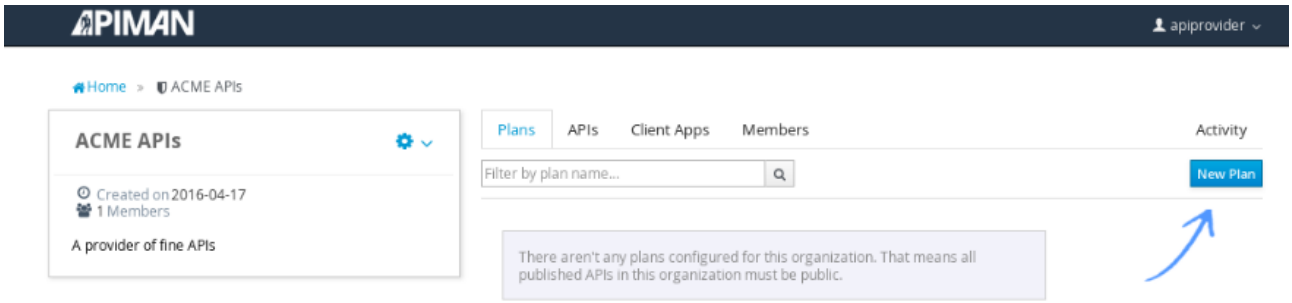


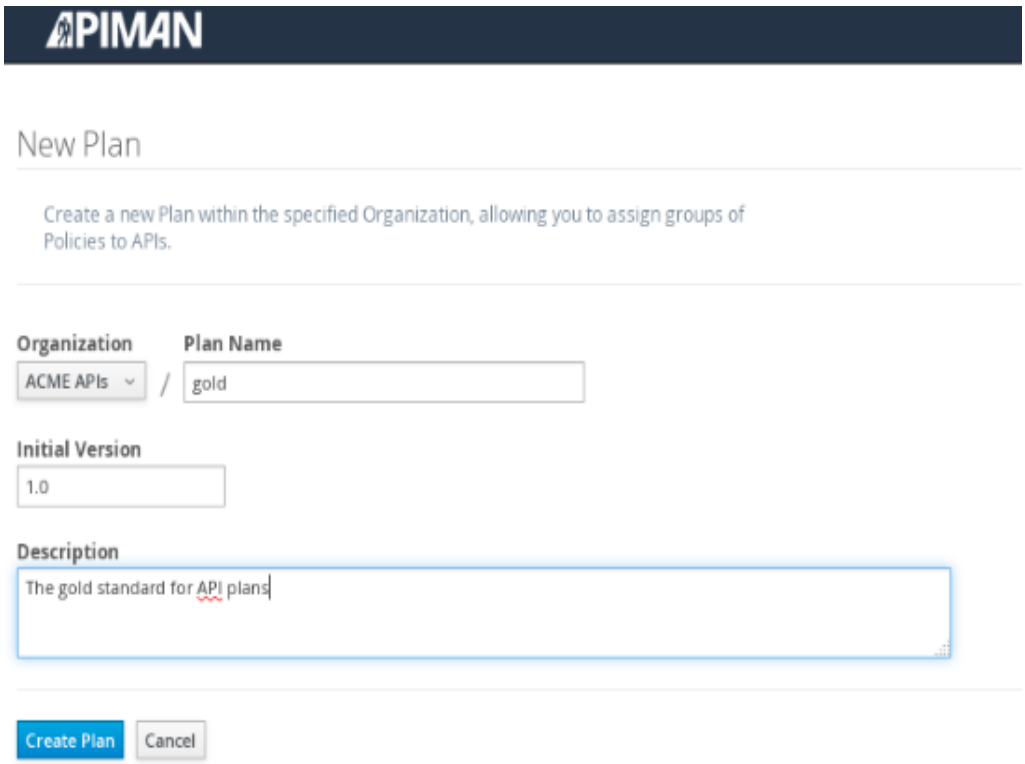
Configuring the API, its Policies, and Plans

After creating the organization, you can start creating a plan, policies which are applied by Gateway at runtime when requests to the API are made by consumers

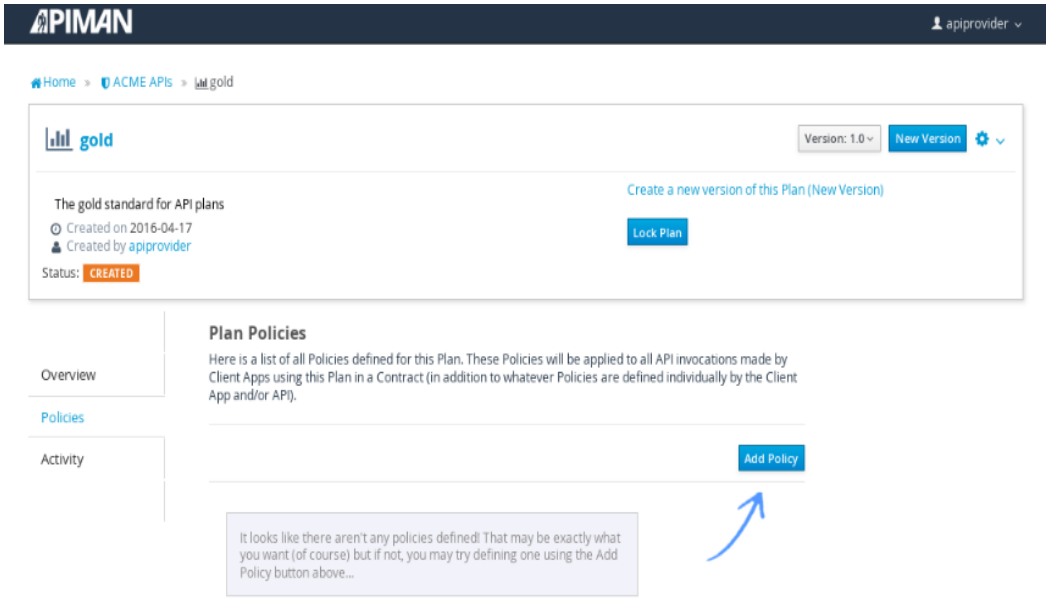
To create a new plan, select the “Plans” tab. We’ll create a “gold” plan:

1. Select Plans in Organization homepage. Here you are creating a Gold plan.



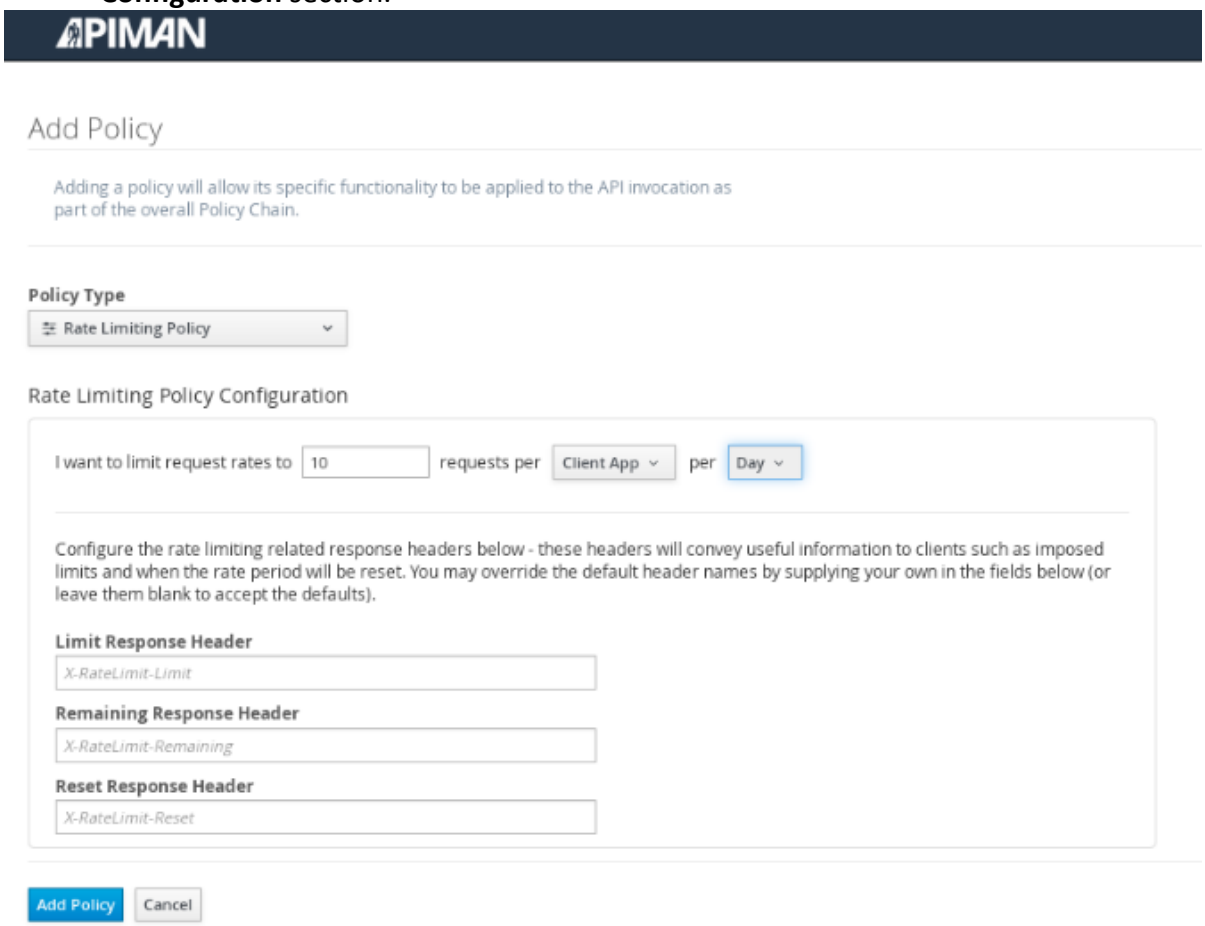


NOTE: You can create multiple plans.



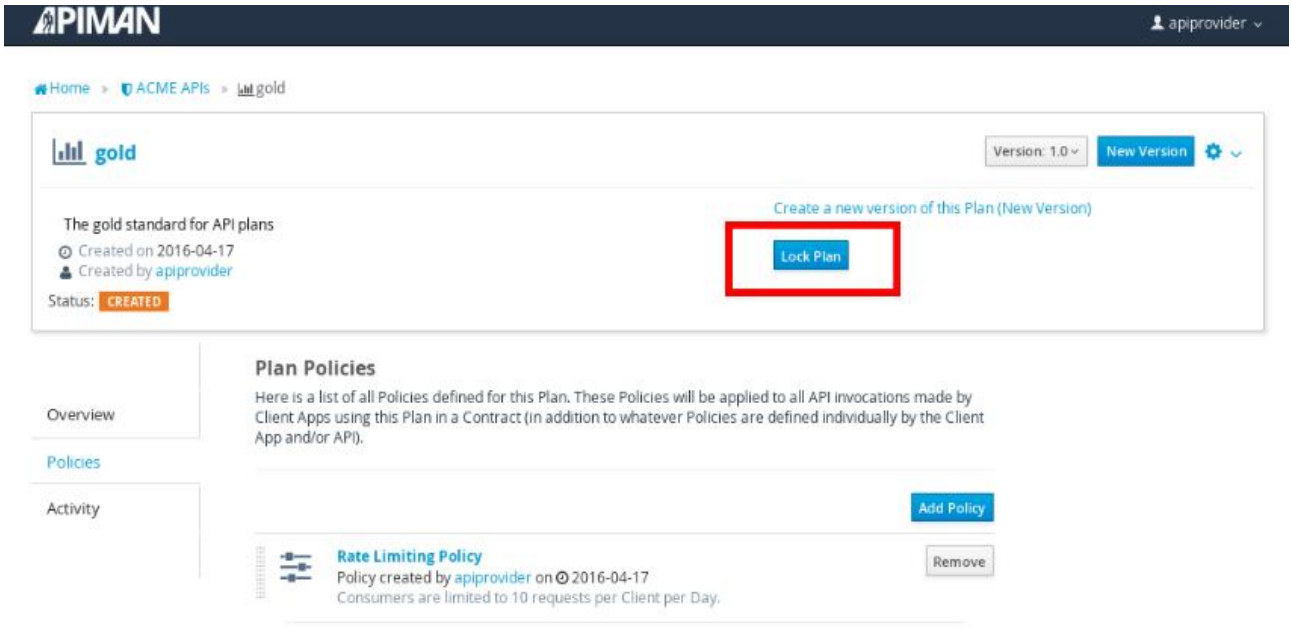
2. View Policies in API Manager UI Home > Organization > Plan and click Add Policy.

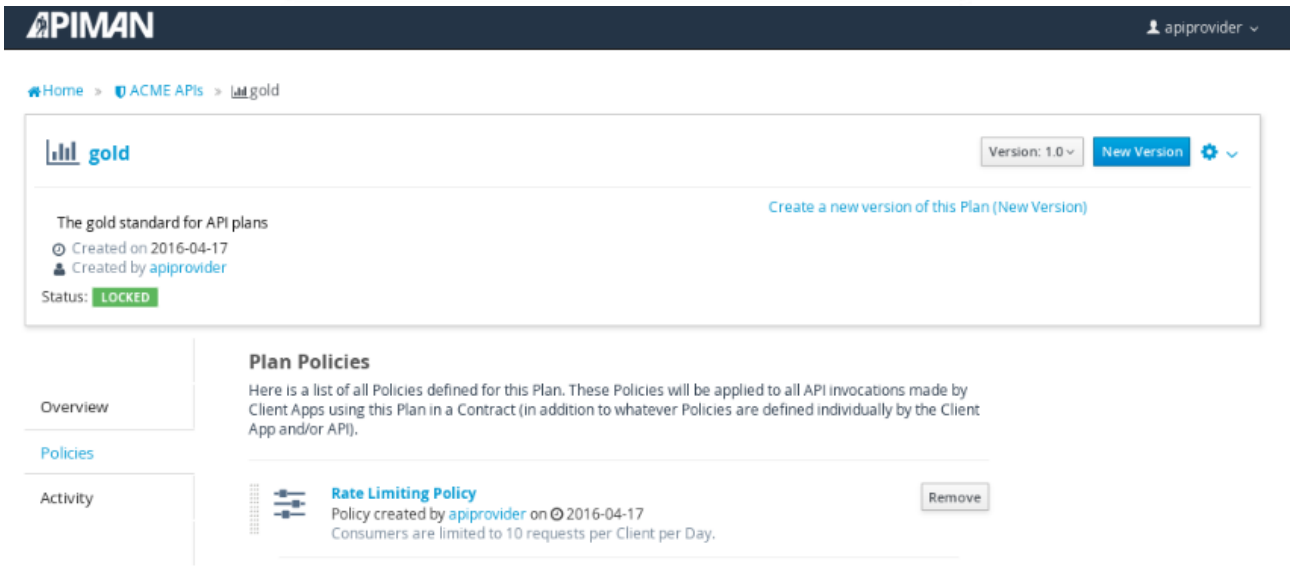
3. For this demonstration, we can set Rate Limiting Policy as 10. This restricts the number of requests for this API to 10 per day/month as per your selection in the **Rate Limiting Policy Configuration** section.



NOTE: You can set various other policies too. However here we are focusing on Rate Limiting Policy.

4. Lock the plan once the policies for the plan are set.



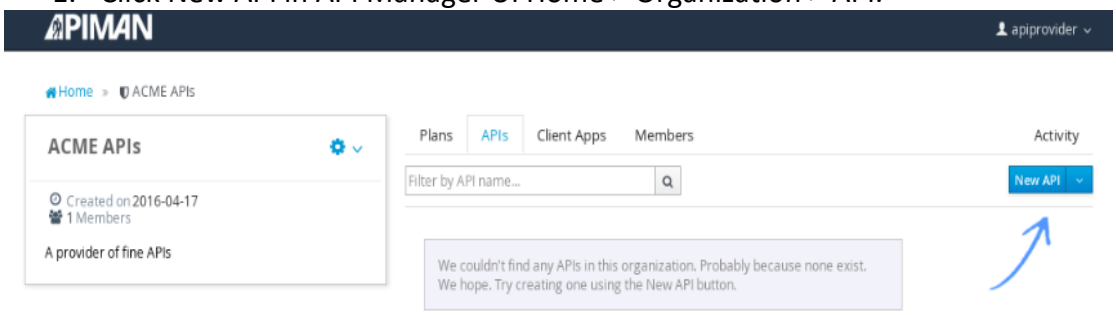


NOTE: Make sure to Lock each plan when done, otherwise you won't be able to use it in your services.

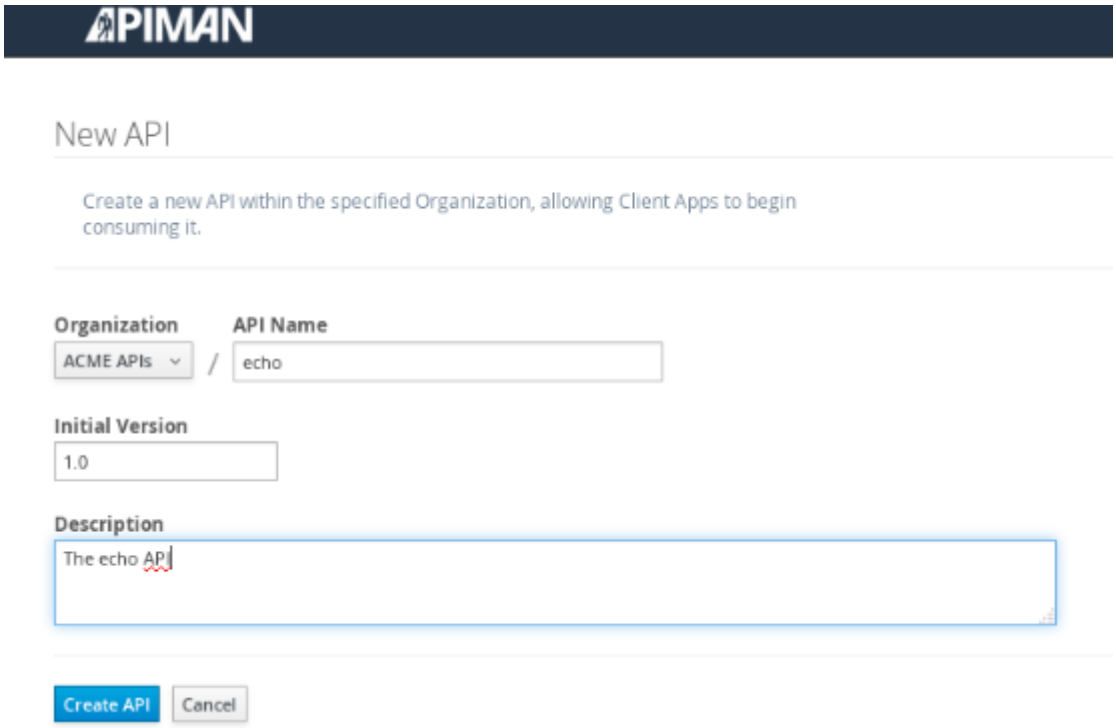
RESULT: You have configured the plans and policies for the Echo API.

Defining API

1. Click New API in API Manager UI Home **>** Organization **>** API.



2. Enter an appropriate name that can be uniquely identified by consumers and providers.

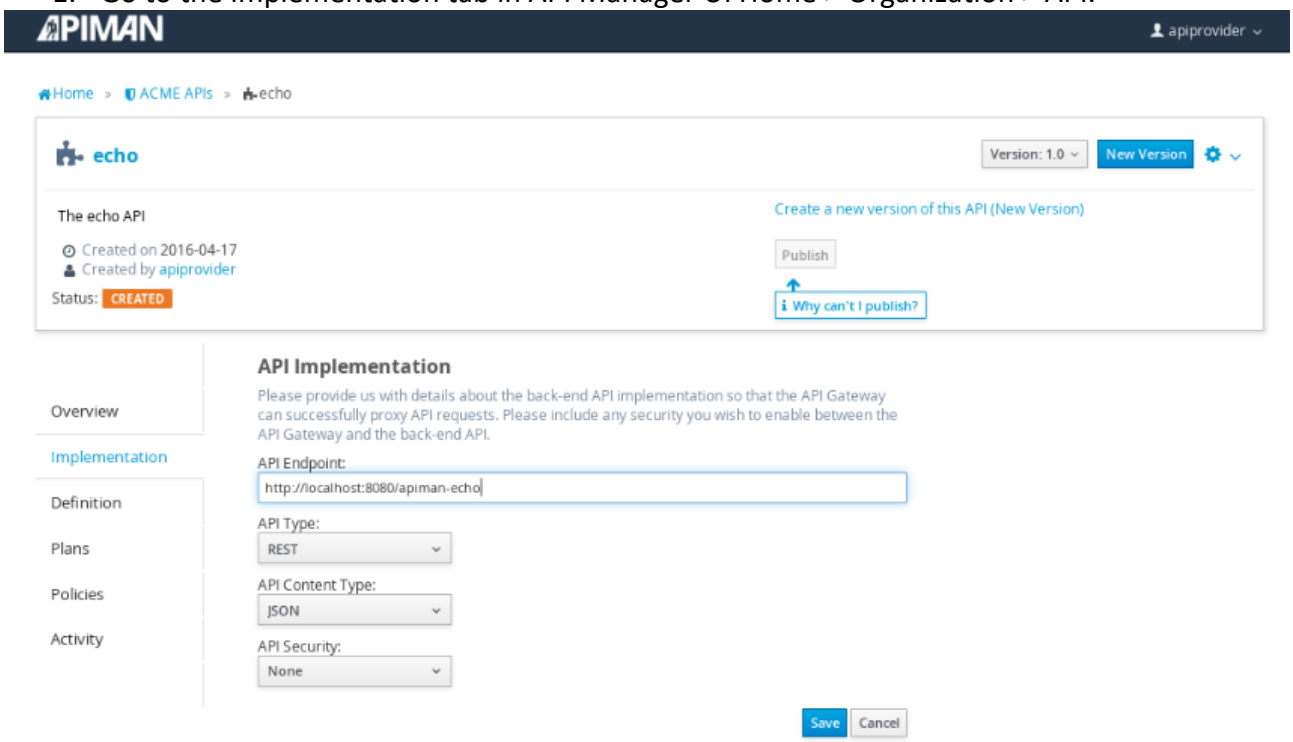


RESULT: You have defined the API.

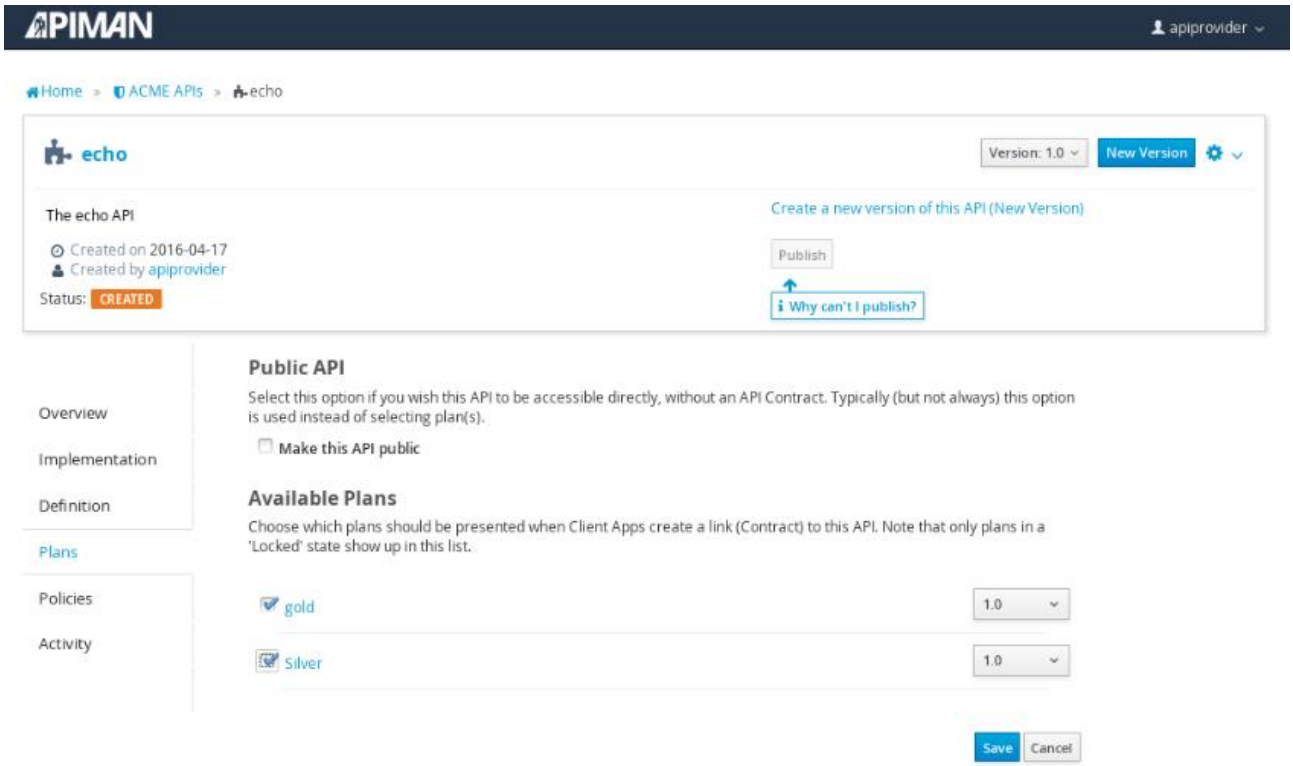
Implementing API

Once you have defined the API, you can define its implementation.

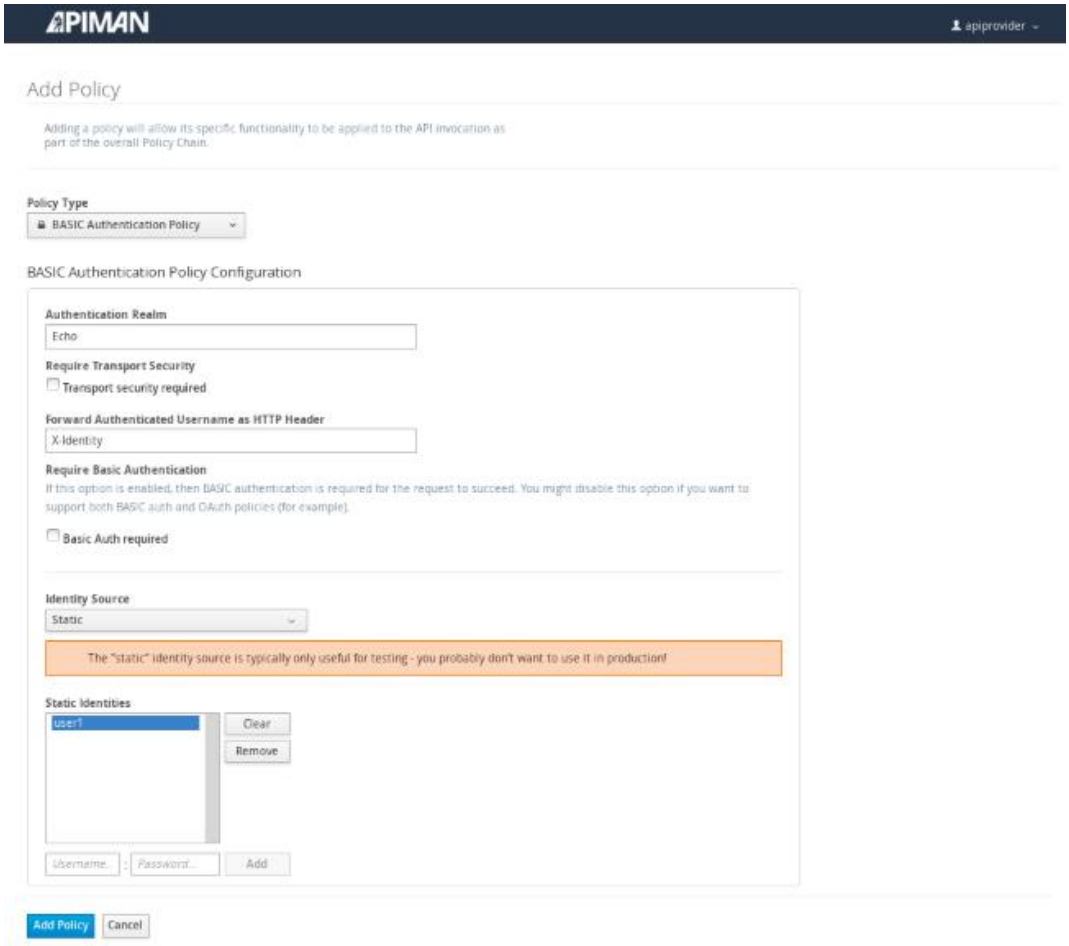
1. Go to the Implementation tab in API Manager UI Home **>** Organization **>** API.



2. Enter the API Endpoint using which the API Gateway identifies the API. The API Endpoint for Echo API is - http://localhost:8080/apiman-echo.
3. Select the Plans which you want to apply for the API.

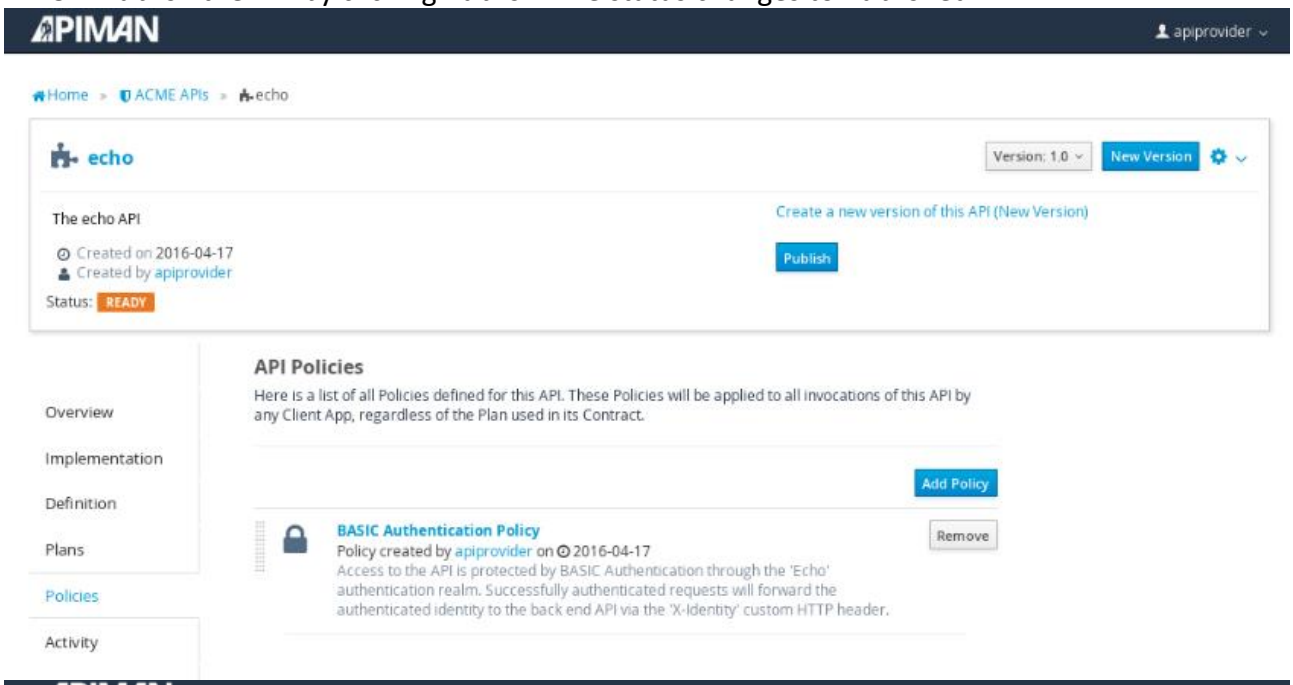


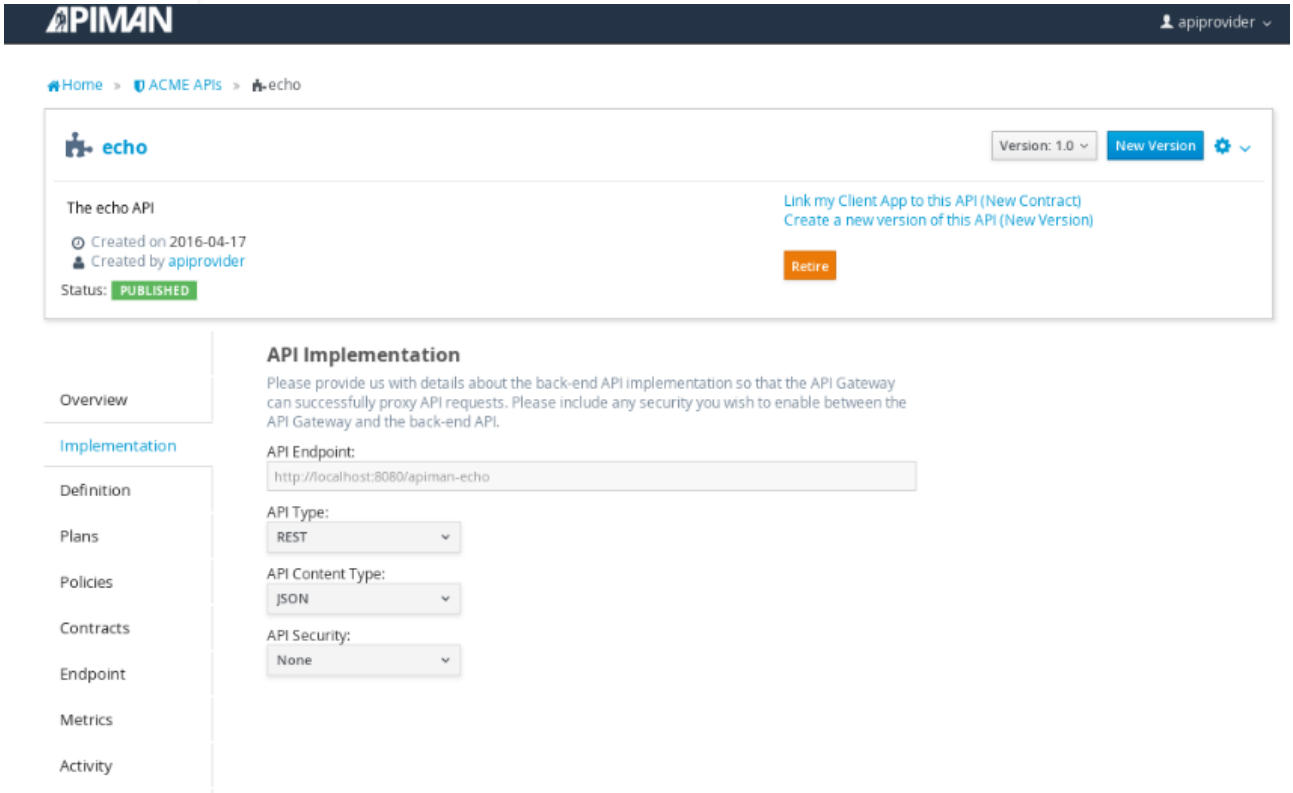
4. Enter the authentication details in the Policies tab. This ensures that consumers have to log in to access the API.



NOTE: Remember the user name and password that you define here as you will need them later when you send requests to the API.

5. Publish the API by clicking Publish. The status changes to Published



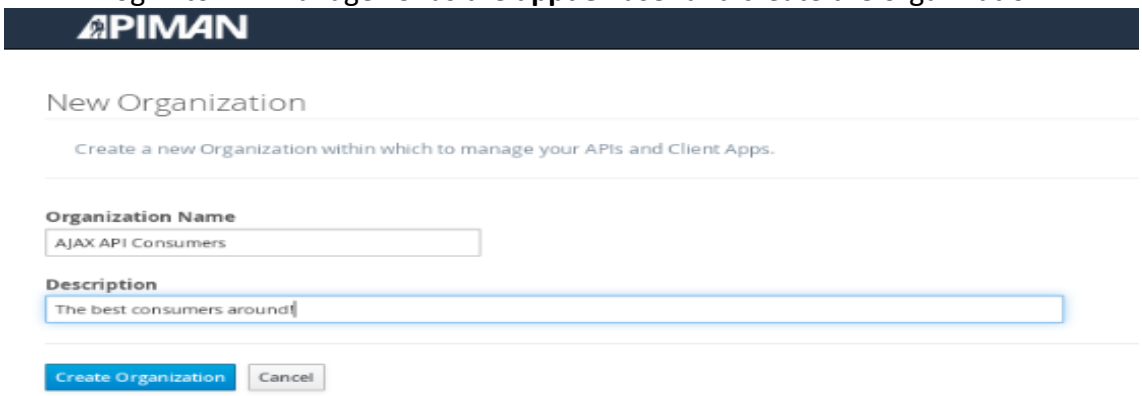


RESULT: You have implemented the API, applied plans and policies, and published it.

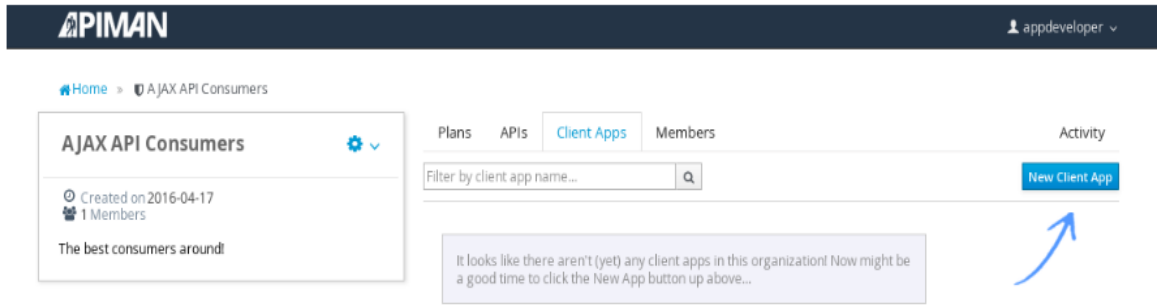
Creating API consumer organization

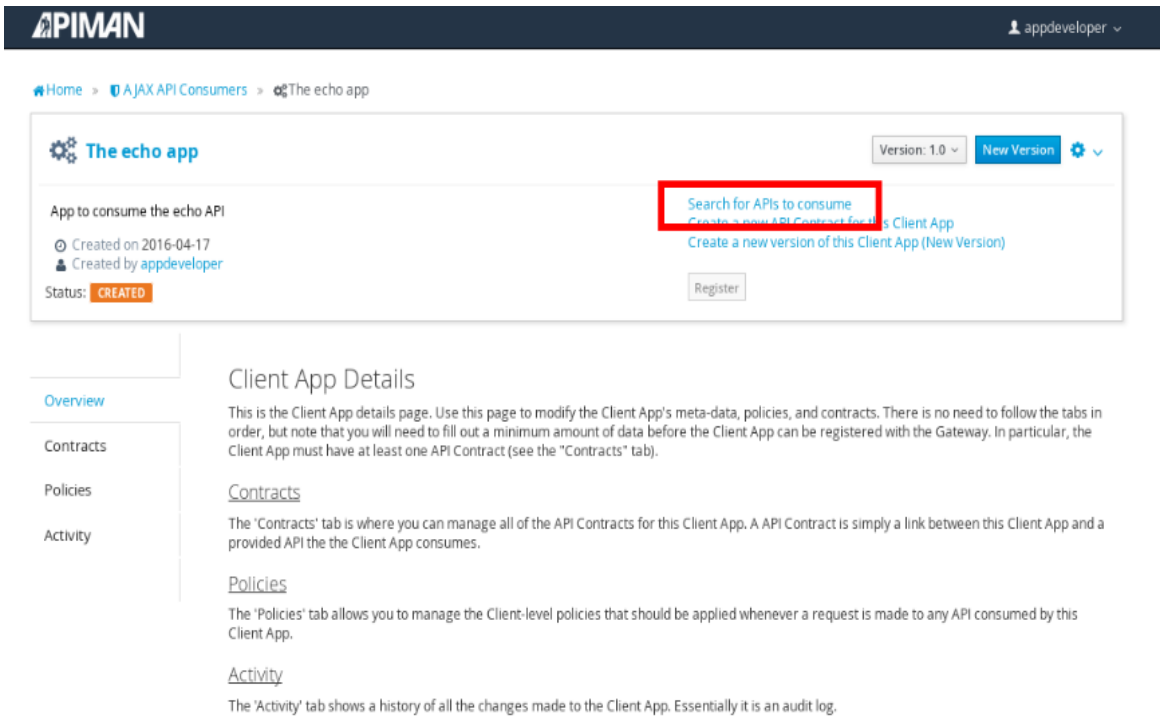
Log in to the API consumer side as an application developer and create API consumer organization and register an application that connects with the Echo API via API Gateway by creating a Service Contract with it.

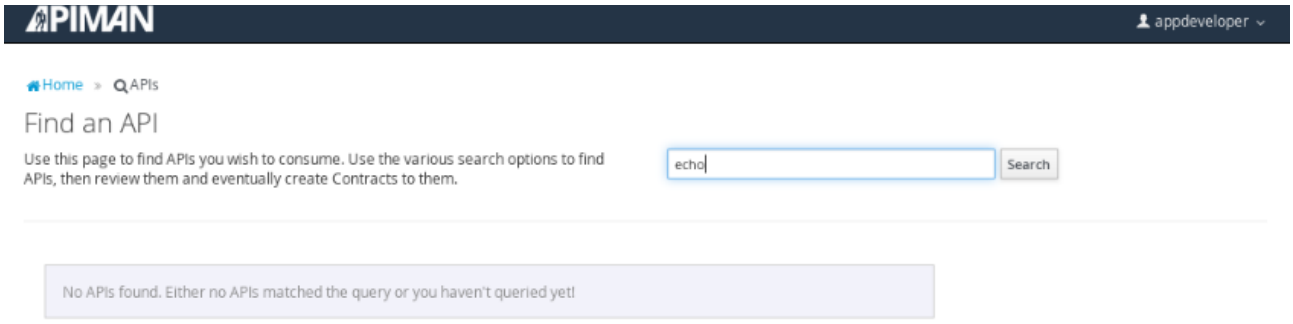
1. Log in to API Manager UI as the **appdev** user and create the organization.

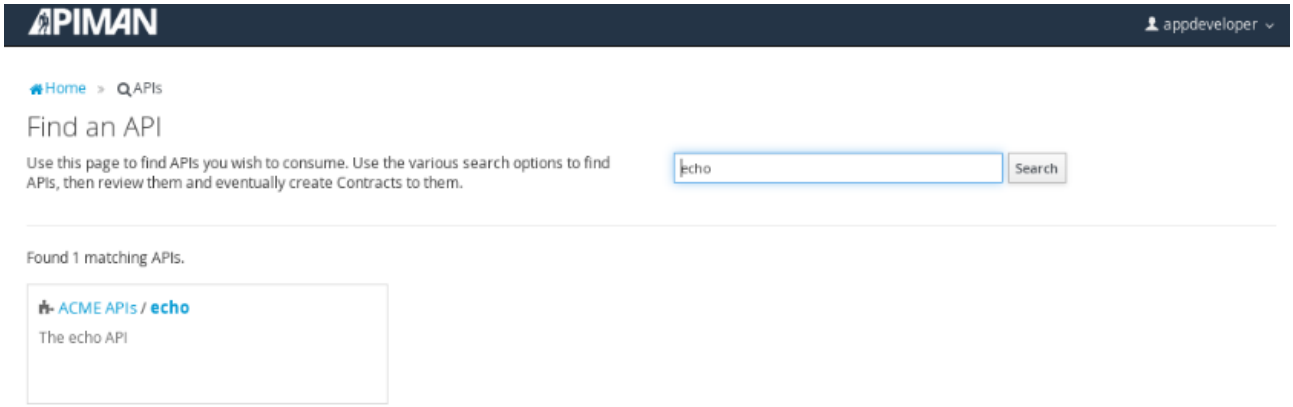


2. Create a new application in API Manager UI **>** Organization **>** Client Apps and then search for the API to be used by the application. Since you are acting as the Application Developer now, you don't have to create any Plans or Services. Simply create an Application so that you can consume Services.

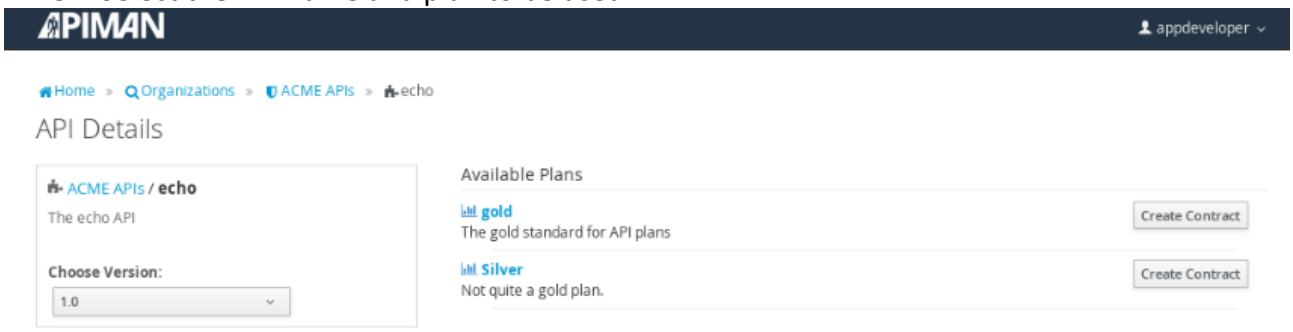






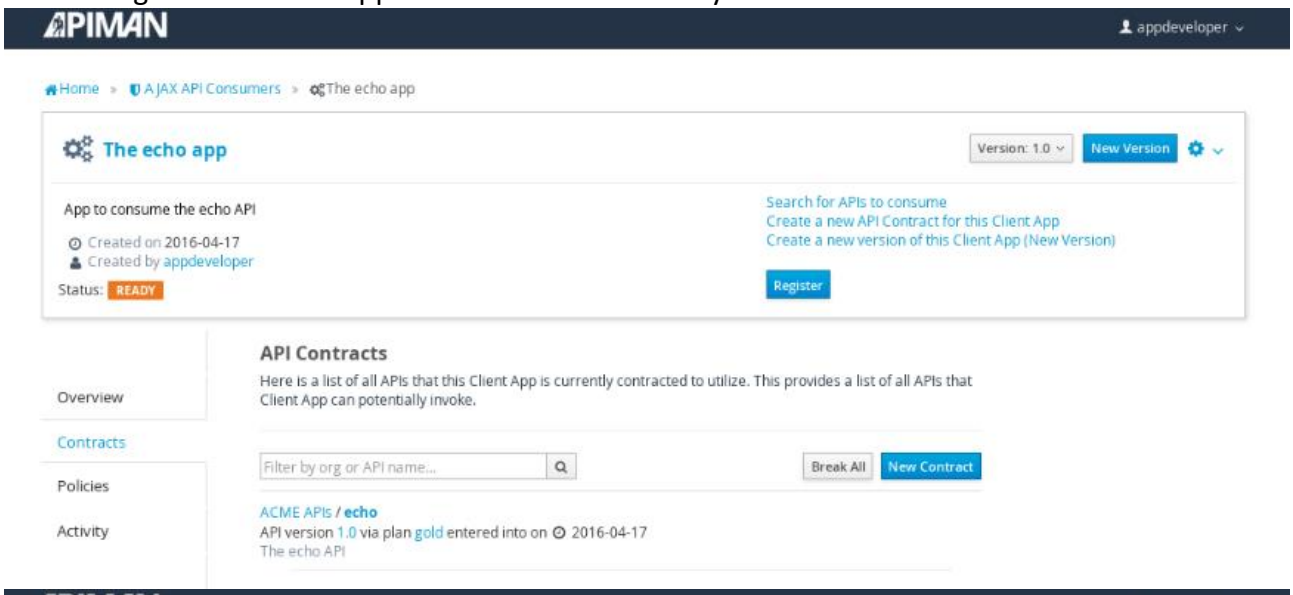


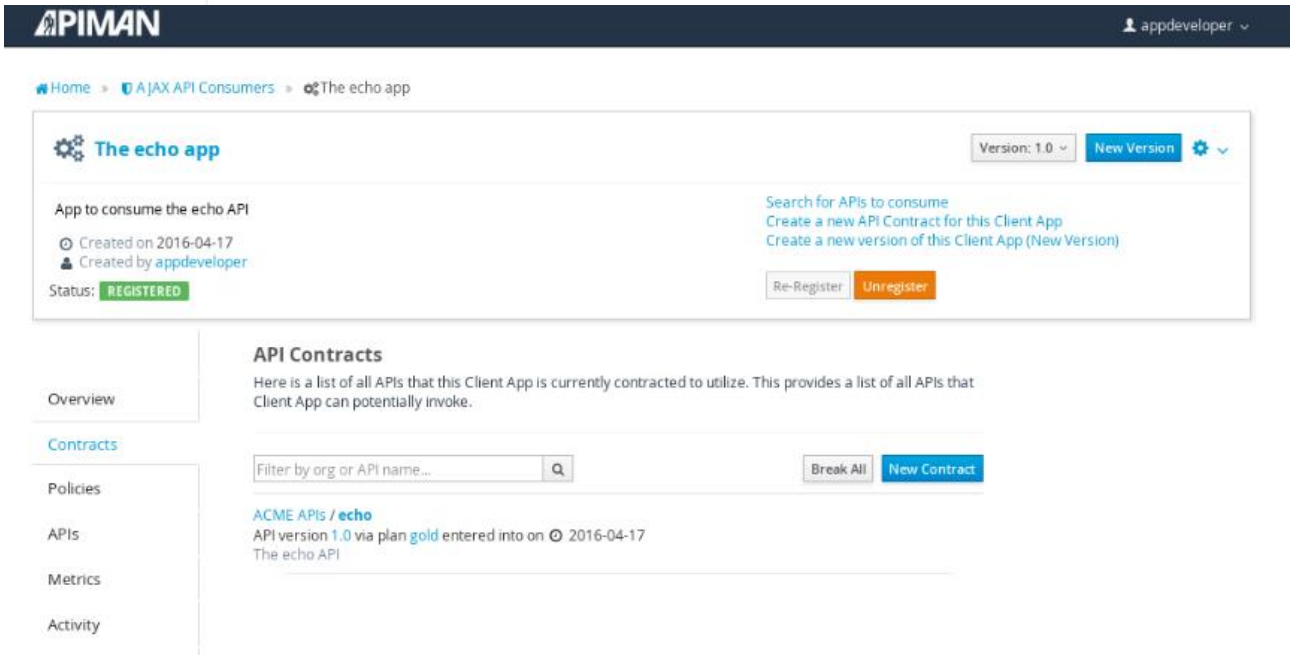
3. Select the API name and plan to be used.



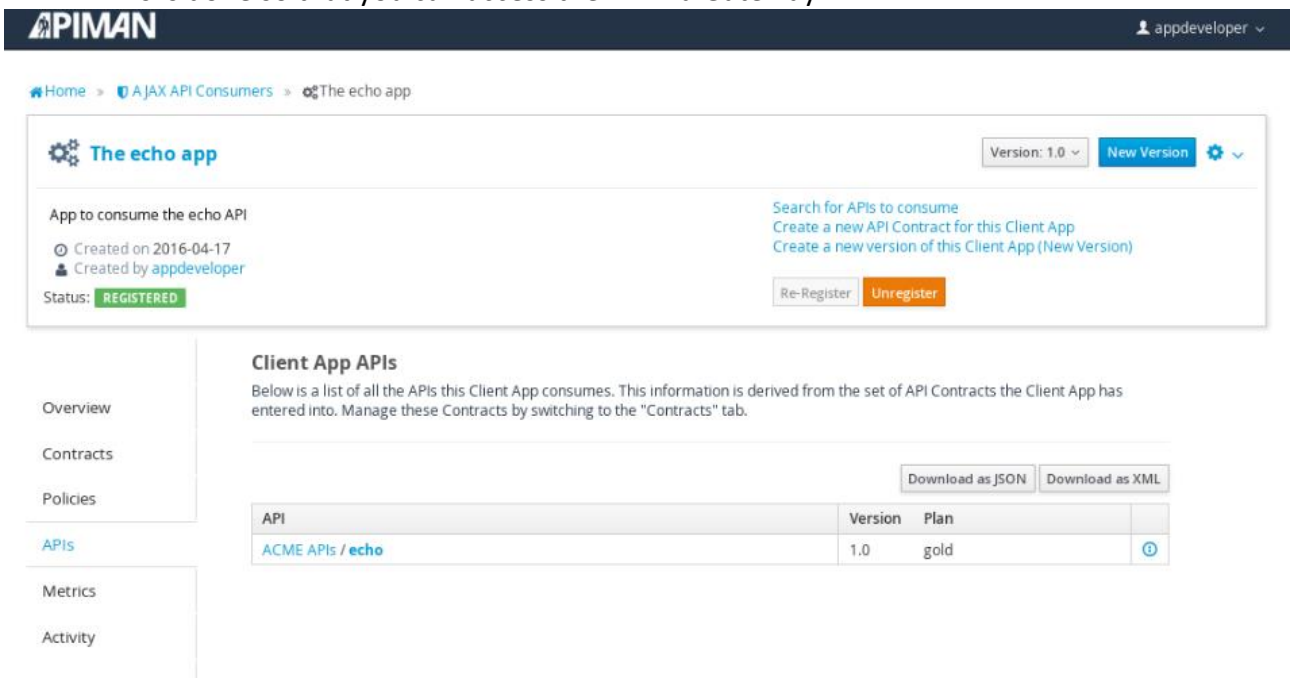
4. Select Create Contract for the plan and configure the settings to default values.

5. Register the client application with API Gateway.

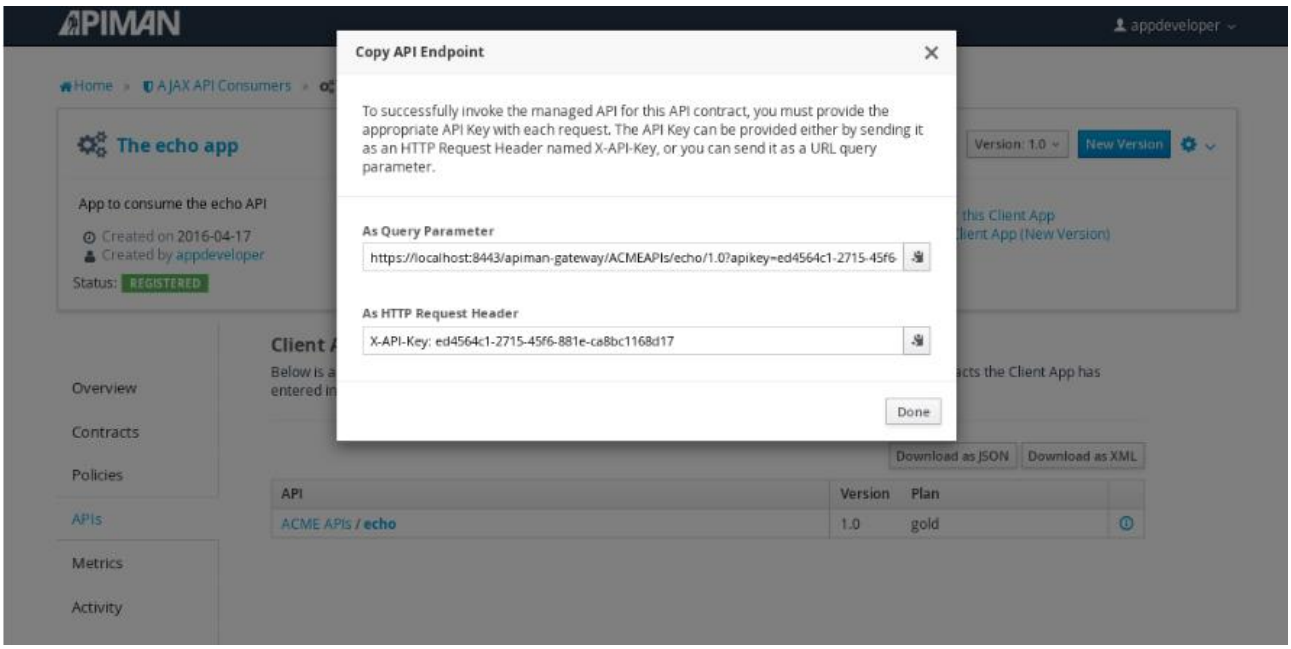




6. Go to the API Manager UI > Client Application > API to obtain the URL of the managed API. This is done so that you can access the API via Gateway.



7. Click the information icon on the above screen.



You can access the Echo API through the Gateway by providing API Key with each request. You can send it through HTTP Header or URL query parameter.

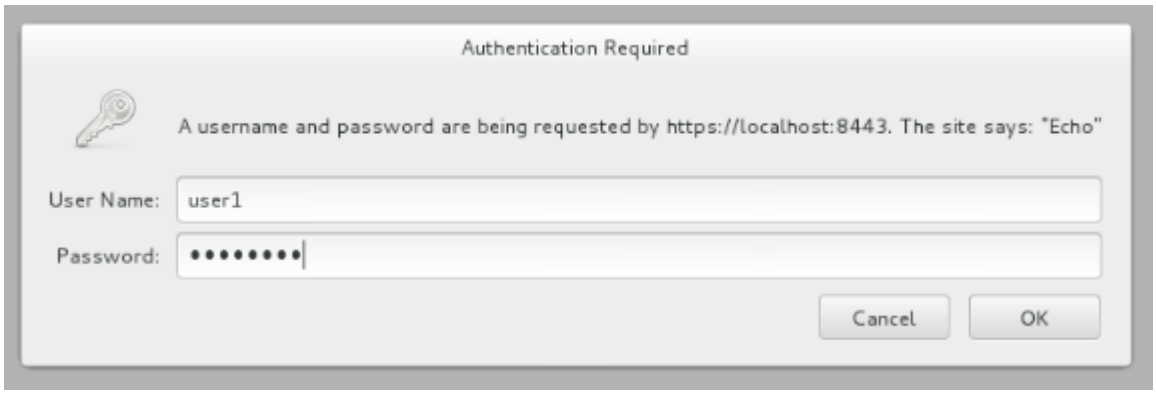
For example, the API request looks like this:

https://localhost:8443/apiman-gateway/ACMEAPIs/echo/1.0?apikey=ed4564c1-2715-45f6-881e-ca8bc1168d17
Copy the URL into the clipboard.

Testing Echo API QuickStart

You can start by firing up the client.

1. Open a new browser window or tab, and enter the URL for the managed API - https://localhost:8443/apiman-gateway/ACMEAPIs/echo/1.0?apikey=ed4564c1-2715-45f6-881e-ca8bc1168d17.
   The below authentication dialog appears:



2. Enter user credential created during authentication policy to access the API. This indicates you are accessing managed API.
3. Send a GET request to the API, you should see a successful response


```
{
"method" : "GET",
"resource" : "/apiman-echo",
"uri" : "/apiman-echo",
"headers" : {
"Cookie" : "s_fid=722D028B20E49214-13EAE1456E752098; __utma=111872281.807845787.
    1452188093.1460777731.1460777731.4; __utmz=111872281.1452188093.1.1.utmcsr=(direct)|utmccn
    =(direct)|utmcmd=(none); _ga=GA1.1.807845787.1452188093; __qca=P0-404983419-1452188093717;
    __utmc=111872281",
"Accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
"User-Agent" : "Mozilla/5.0 (X11; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0",
"Connection" : "keep-alive",
"X-Identity" : "user1",
"Host" : "localhost:8080",
"Accept-Language" : "en-US,en;q=0.5",
"Accept-Encoding" : "gzip, deflate",
"DNT" : "1"
},
"bodyLength" : null,
"bodySha1" : null
}
```


4. Send 10 more requests. You can see a message indicating that you have exceeded the Gold plan.


```
{
  "type" : "Other",
  "headers" : {
    "empty" : false,
    "entries" : [
      {
        "X-RateLimit-Remaining" : "-1"
      },
      {
        "X-RateLimit-Reset" : "50904"
      },
```

```
      {
        "X-RateLimit-Limit" : "10"
      }
    ]
  },
  "failureCode" : 10005,
  "message" : "Rate limit exceeded.",
  "responseCode" : 429
}
```

RESULT: You have successfully tested the Echo API Quickstart.

References:

For more information, see

- Apiman site - http://www.apiman.io/latest/
- Apiman blog - http://www.apiman.io/blog/
- Apiman downloads - http://www.apiman.io/latest/download.html
- Apiman user guide - http://www.apiman.io/latest/user-guide.html
- Apiman developer guide - http://www.apiman.io/latest/developer-guide.html
- Apiman videos - https://vimeo.com/user34396826
- Apiman on Github - https://github.com/apiman
- Apiman on JIRA - https://issues.jboss.org/projects/APIMAN