|  |  |
| --- | --- |
|  |  |
| *Open Onderwijs API – Installatiehandleiding* |
|  |
| *SURFnet* |



**Versiebeheer**

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Versie | Commentaar | Auteur |
| 16-12-2015 | 1.0 |  | Ton Swieb |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **Titel** | Installatiehandleiding |
| **Status** | Definitief |
| **Project** | Open Onderwijs API |
| **Datum** | 16-12-2015 |
| **Auteurs** | Jan Blom, Ton Swieb |
| **Document eigenaar** | Ton Swieb |
| **Distributielijst** | Frans Ward (SURFnet) |
| **Goedkeuring** | Frans Ward (SURFnet) |

**Contactpersoon**

|  |  |
| --- | --- |
| **Naam** | Ton Swieb |
| **Functie** | Solution Expert Applicatie Integratie |
| **Kantooradres** | Finalist Rotterdam  Groot Handelsgebouw  Stationsplein 45, A4.205  3013 AK Rotterdam |
| **Telefoonnummer** | 088 217 08 15 |
| **E-mail adres** | [ton.swieb@finalist.nl](mailto:ton.swieb@finalist.nl) |

|  |  |
| --- | --- |
| **Naam** | Jan Blom |
| **Functie** | Software Engineer |
| **Kantooradres** | Finalist Eindhoven  Willemstraat 1L  5611 HA Eindhoven |
| **Telefoonnummer** | 088 217 08 00 |
| **E-mail adres** | jan.blom@finalist.nl |

**Inhoudsopgave**

1. Installatie 5

1.1. Installatie mogelijkheden 5

1.2. OOAPI Docker Image 5

1.3. JBoss Apiman overlay voor JBoss Wildfly 5

1.3.1. Installatie JBoss Wildfly 5

1.3.2. Installatie JBoss Apiman 6

1.3.3. Installatie JBoss Keycloak 6

1.3.4. Installatie js-console app 6

1.3.5. Installatie spssodescriptor service 6

1.3.6. Opstarten JBoss Apiman 7

2. Configuratie 7

2.1. JBoss Wildfly 7

2.1.1. Signed SSL certificaat installeren 7

2.1.2. Uitschakelen HTTP 8

2.2. JBoss Keycloak 8

2.2.1. Inloggen JBoss Keycloak 9

2.2.2. Opvragen Realm certificaat 10

2.2.3. Toevoegen js-console Client 11

2.2.4. Toevoegen Identity Provider 12

2.2.5. Attribute Mapping 12

2.3. Configuratie JBoss Apiman 18

2.3.1. Installatie extra plugins 19

2.3.2. Configureren Gateway Endpoint 22

2.3.3. Aanmaken nieuwe organisatie 24

2.3.4. Publiceren service 34

# Installatie

Dit document is een handleiding voor de installatie van een demo omgeving voor de Open Onderwijs API (OO-API). Deze omgeving demonstreert de werking van API management in combinatie met authenticatie op basis van SURFconext.

De demo omgeving is gebaseerd op de producten:

* JBoss Apiman v1.1.9.Final als API manager
* JBoss Keycloak v1.2.0 als Identity Broker
* JBoss Wildfly v8.2 als Applicatie Server

## Installatie mogelijkheden

Voor het opzetten van de demo omgeving zijn een aantal mogelijkheden:

1. OO-API Docker Image: Een Docker image met een volledig geïnstalleerde en geconfigureerde omgeving welke verder geen handmatige configuratie handelingen vereist.
2. JBoss Apiman overlay voor JBoss Wildfly: Een kale installatie direct op een VM welke nog handmatige installatie en configuratie handelingen vereist. De benodigde handelingen staan beschreven in dit document.

## OO-API Docker Image

Voor een uitgebreide toelichting op wat Docker is en hoe dit werkt zie [Docker](https://www.docker.com/). Voor de doel van deze Proof of Concept stelt [Docker](https://www.docker.com/) ons in staat om een volledig geïnstalleerde en geconfigureerde omgeving op te leveren welke in elke omgeving moet kunnen draaien.

De OO-API Docker image is nog niet beschikbaar als image in [Docker Hub](https://hub.docker.com/) omdat de Docker build nog niet volledig omgeving onafhankelijk is. De volgende zaken moeten handmatig gezet worden tijdens de build:

* URL van de omgeving waarop de demo geïnstalleerd wordt;
* Admin wachtwoord;

Zie de documentatie bij het project voor instructies voor het bouwen van het Docker image en het opstarten van de Docker container.

<https://github.com/OpenOnderwijsAPI/Docker-Image>

## JBoss Apiman overlay voor JBoss Wildfly

JBoss Apiman is beschikbaar als overlay voor JBoss Wildfly. Een overlay is een ZIP bestand dat uitgepakt wordt over een JBoss Wildfly directory structuur. *Het is belangrijk dat dit wordt geïnitieerd vanuit de juiste directory met de juiste ZIP configuratie opties!*

De installatie instructies zijn gebaseerd op de [Apiman Getting Started](http://www.apiman.io/latest/download.html). De installatie instructies gaan uit van een Linux/Unix omgeving, maar installatie op Windows is ook mogelijk. Het betreffen immers ZIP bestanden die uitgepakt moeten worden waarna het opstarten van de applicatie uitgevoerd wordt met respectievelijk \*.sh of \*.bat bestanden.

### Installatie JBoss Wildfly

Installatie van JBoss Wildfly in de directory apiman-1.1.9.Final in de home directory.

* mkdir ~/apiman-1.1.9.Final
* cd ~/apiman-1.1.9.Final
* curl [http://downloads.jboss.org/wildfly/8.2.0.Final/wildfly-8.2.0.Final.zip -o wildfly-8.2.0.Final.zip](http://downloads.jboss.org/wildfly/8.2.0.Final/wildfly-8.2.0.Final.zip)
* unzip wildfly-8.2.0.Final.zip

### Installatie JBoss Apiman

* curl [http://downloads.jboss.org/apiman/1.1.9.Final/apiman-distro-wildfly8-1.1.9.Final-overlay.zip -o apiman-distro-wildfly8-1.1.9.Final-overlay.zip](http://downloads.jboss.org/apiman/1.1.9.Final/apiman-distro-wildfly8-1.1.9.Final-overlay.zip)
* unzip -o apiman-distro-wildfly8-1.1.9.Final-overlay.zip -d wildfly-8.2.0.Final

### Installatie JBoss Keycloak

De JBoss Apiman overlay bevat naast de JBoss Apiman installatie ook de JBoss Keycloak installatie. JBoss Keycloak hoeft dus niet apart geïnstalleerd te worden. De versie van JBoss Keycloak is een oudere versie, v1.2.0, dan die momenteel beschikbaar is, v1.7.0. Echter voor deze Proof of Concept maakt dit geen verschil.

### Installatie js-console app

* curl <https://github.com/OpenOnderwijsAPI/Docker-Image/raw/master/content/deployments/js-console.war>
* cp js-console.war wildfly-8.2.0.Final/standalone/deployments/

### Installatie spssodescriptor service

* curl <https://github.com/OpenOnderwijsAPI/Docker-Image/raw/master/content/deployments/keycloak-saml-spssodescriptor-1.0.0-SNAPSHOT.war>
* cp keycloak-saml-spssodescriptor-1.0.0-SNAPSHOT.war wildfly-8.2.0.Final/standalone/deployments/

#### Configuratie system properties

Voor het goed functioneren van de SP SSO Descriptor service moeten Java system properties geconfigureerd worden, te weten:

* *spssodescriptor.username* : De gebruikersnaam van een gebruiker met tenminste de rol *view-identity-providers* binnen de te bevragen realm.
* *spssodescriptor.password* : Het wachtwoord van de gebruiker
* *spssodescriptor.baseUrl* : De URL, <protocol:host:port>, waarop JBoss Wildfly beschikbaar is.

Het toevoegen van system properties kan door:

1. Het handmatig opnemen van de properties in standalone-apiman.xml
2. Via de JBoss Command Line Interface.  
   De JBoss CLI vereist dat JBoss Wildfly is opgestart.

##### Handmatig configureren van system properties

Neem het volgende stukje XML op in wildfly-8.2.0.Final/standalone/configuration/standalone-apiman.xml en pas de properties waardes aan zodat ze overeenkomen met de eigen omgeving

<system-properties>

<property name="spssodescriptor.username" value="admin"/>

<property name="spssodescriptor.password" value="admin123!"/>

<property name="spssodescriptor.baseUrl" value="http://apiman.openonderwijsapi.nl:7443"/>

</system-properties>

Het stukje XML moet opgenomen worden tussen </extensions> en <management>

##### Configureren van system properties via JBoss CLI

Start JBoss Wildfly op:

* cd wildfly-8.2.0.Final
* ./bin/standalone(.sh|.bat) -c standalone-apiman.xml -b 0.0.0.0

Start JBoss CLI op:

* cd wildfly-8.2.0.Final
* ./bin/jboss-cli(.sh|.bat)

Vervolgens In JBoss CLI de volgende commando’s uitvoeren:

connect

1. /system-property=spssodescriptor.username:add(value=admin)
2. /system-property=spssodescriptor.password:add(value=admin123!)
3. /system-property=spssodescriptor.baseUrl:add(value=http://apiman.openonderwijsapi.nl:7443)
4. exit

### Opstarten JBoss Apiman

* cd wildfly-8.2.0.Final
* ./bin/standalone.sh -c standalone-apiman.xml -b 0.0.0.0

De -b optie is noodzakelijk om de services van JBoss Wildfly op alle network interfaces beschikbaar te stellen en niet alleen op localhost.

De -c optie is noodzakelijk om de JBoss Wildfly op te starten met de JBoss Apiman / JBoss Keycloak configuratie en niet met de standaard configuratie van een “lege” applicatie server.

# Configuratie

De configuratie van de demo omgeving bestaat uit:

* Configuratie van de applicatie server JBoss Wildfly
* Configuratie van JBoss Keycloak
* Configuratie van JBoss Apiman

## JBoss Wildfly

De JBoss Apiman overlay voor JBoss Wildfly verzorgt de noodzakelijk te configureren onderdelen van de applicatie server zodat de JBoss Apiman en JBoss Keycloak kunnen functioneren.

Voor een demo omgeving is het wenselijk dat tenminste:

* Een signed SSL certificaat geïnstalleerd wordt welke overeenkomt met de URL waarop de demo omgeving beschikbaar wordt gesteld.
* Het HTTP endpoint uitgeschakeld wordt zodat alleen verbindingen via HTTPS geaccepteerd worden.

Voor een lokale omgeving kunnen deze stappen achterwegen gelaten worden. Bijvoorbeeld wanneer:

* De omgeving gekoppeld wordt aan een SURFconext test omgeving;
* De omgeving gekoppeld wordt aan een ander IdP voor testdoeleinde zoals SimpleSAMLphp;

### Signed SSL certificaat installeren

Op basis van een signed SSL certificaat moet een Java keystore gemaakt worden.

Het maken van een Java keystore staat … beschreven. Aandachtspunten hierbij zijn:

* De private key moet bestaan uit de keten van het SSL certificaat + intermediate certificaten + root certificaat.  
  Hiermee wordt niet bedoelt het los opnemen van de intermediate certificaten en het root certificaat in de keystore!
* Indien de keystore de naam apiman.jks krijgt en als wachtwoord voor keystore en key *secret* gebruikt wordt dan zijn er verder geen configuratieaanpassingen noodzakelijk in JBoss Wildfly.

De JBoss Apiman installatie is standaard voorzien van een keystore met een unsigned certificaat. Overschrijf deze keystore, *wildfly-8.2.0.Final/standalone/configuration/apiman.jks*, met de nieuwe keystore.

#### Aanpassen keystore configuratie

Het aanpassen van de keystore configuratie is noodzakelijk indien een andere naam, locatie of wachtwoorden gebruikt wordt voor de keystore.

Wijzig de volgende sectie in *wildfly-8.2.0.Final/standalone/confiugration/standalone-apiman.xml*:

<keystore path="apiman.jks" relative-to="jboss.server.config.dir" keystore-password="secret" alias="apimancert" key-password="secret" />

Een herstart van JBoss Wildfly is noodzakelijk voordat de wijziging actief wordt.

### Uitschakelen HTTP

Voor het uitschakelen van HTTP moeten de volgende XML elementen uit het bestand *wildfly-8.2.0.Final/standalone/confiugration/standalone-apiman.xml* verwijderd worden:

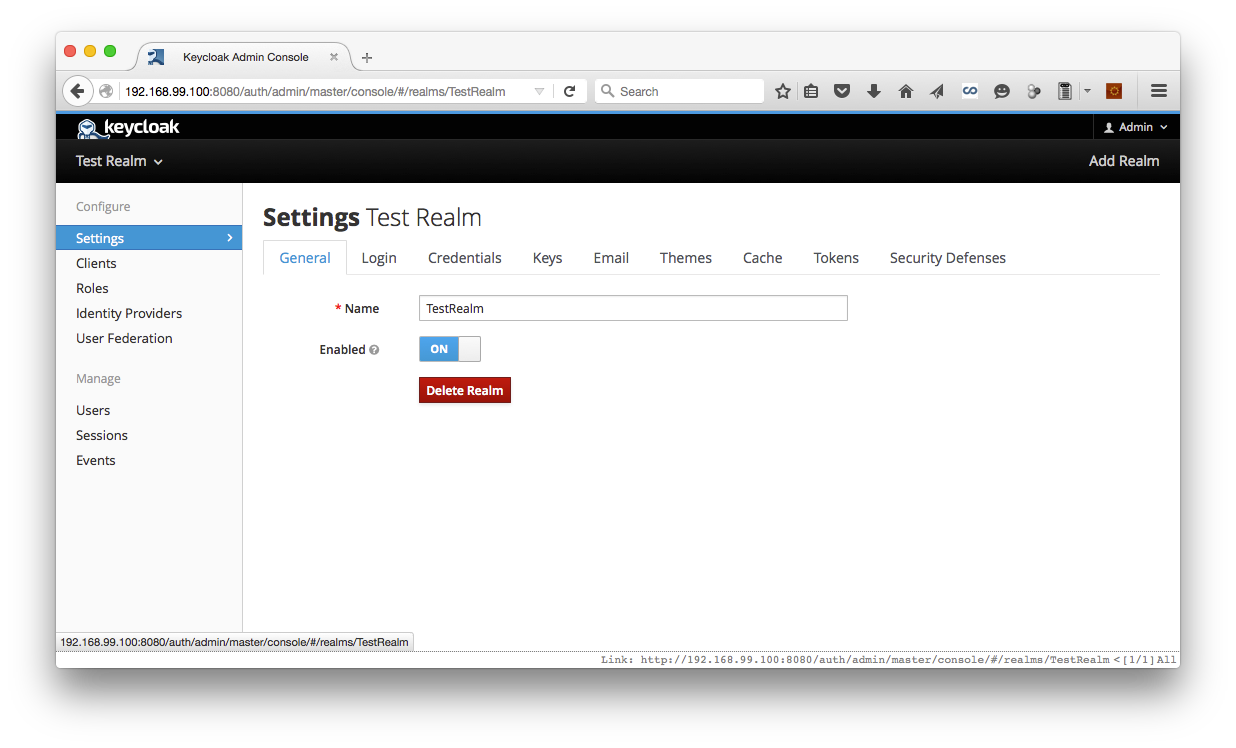
<http-listener name="default" socket-binding="http" />

<socket-binding name="http" port="${jboss.http.port:8080}"/>

## JBoss Keycloak

De meeste relevante instellingen in Keycloak zijn per Real instelbaar, zoals de geldigheidsduur van een Access Token. Het is daarom het beste om voor een specifiek doel/domein ook een eigen Realm aan te maken. Dit is vooral belangrijk als een Keycloak installatie voor verschillende doelen gebruikt wordt: voor een API die alleen gebruikt gaat worden voor read-only toegang op gegevens die minder privacygevoelig zijn, kan de geldigheidsduur van een Access Token veel langer zijn dan voor een API die het wijzigen van dezelfde gegevens toestaat.

Voor het aanmaken van een realm is initieel niet meer nodig dan een naam. Na het aanmaken kan het realm geselecteerd worden, en geconfigureerd:



Binnen het kader van dit project zijn met name de volgende instellingen van belang:

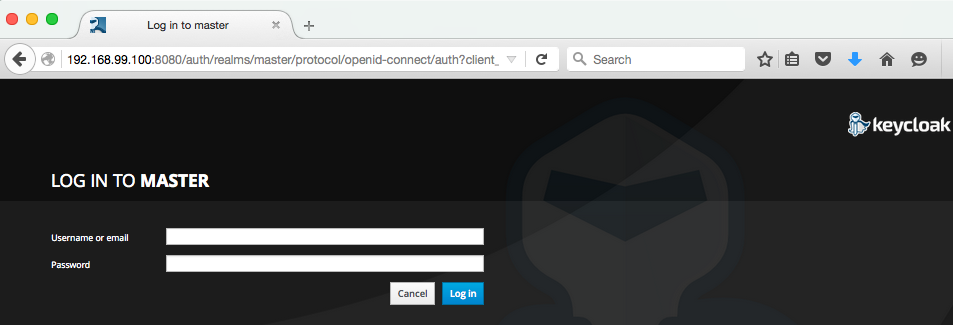
* **Tokens** - hier wordt de geldigheid van sessies en tokens ingesteld. Met name de **Access Token Lifespan** dient te worden ingesteld op een waarde die ruim genoeg is om te voorkomen dat (regelmatige) gebruikers van een API regelmatig genoodzaakt worden opnieuw in te loggen. Vanaf Keycloak versie 1.6.0 is er ook een afzonderlijk offline token beschikbaar. Daarmee is het mogelijk om binnen een realm een onderscheid te maken tussen de geldigheidsduur van online en offline tokens. Offline token maken deel uit van de OpenId specificatie.
* **Cache** - de default instellingen van Keycloak zorgen er voor dat diverse sessie- en usergegevens worden gecached. Met name tijdens testen van configuraties kan dit hinderlijk zijn, en is het handig om de caching hier tijdelijk uit te schakelen

Keycloak biedt ook de mogelijkheid om een realm te configureren vanuit een JSON import file. In de sources van Keycloak zijn daarvoor voorbeelden te vinden.

### Inloggen JBoss Keycloak

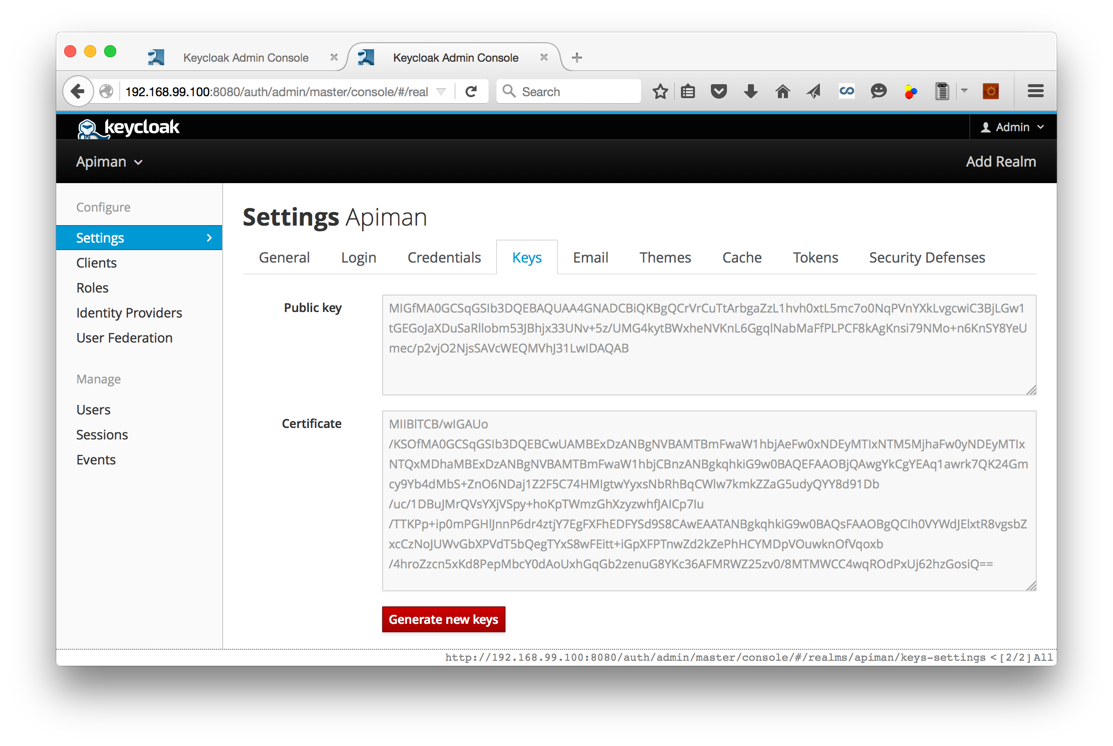
Ga naar de URL <http://localhost:8080/auth/admin>

Het volgende scherm verschijnt waar ingelogd kan worden met de default user **admin** met wachtwoord **admin123!**

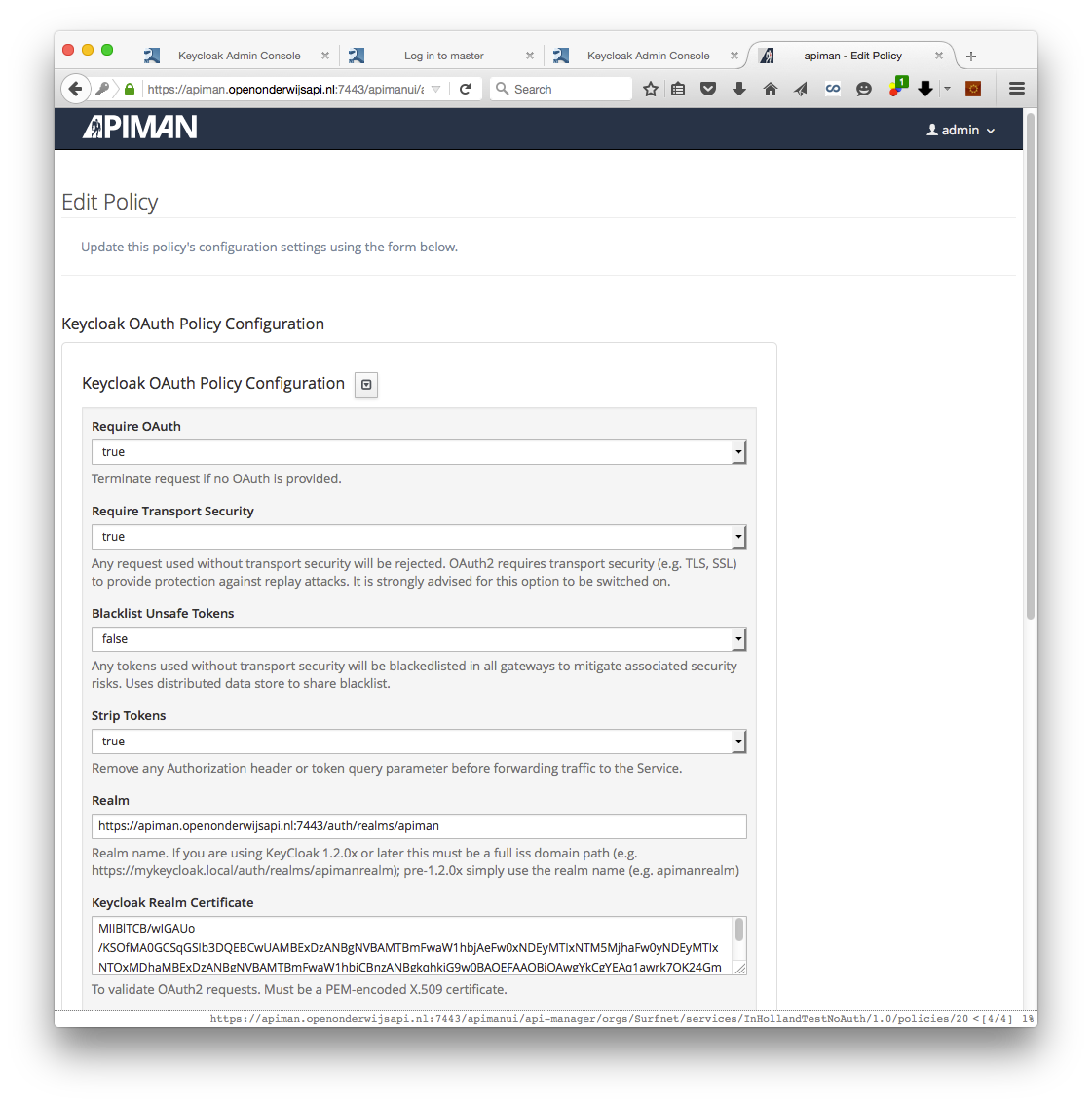


### Opvragen Realm certificaat

Bij het aanmaken van een Realm worden een public key en een certificaat gegenereerd (of ze kunnen worden geïmporteerd via JSON bij het aanmaken van een Realm). Apiman heeft het certificaat nodig om te tokens die Keycloak doorgeeft te valideren:

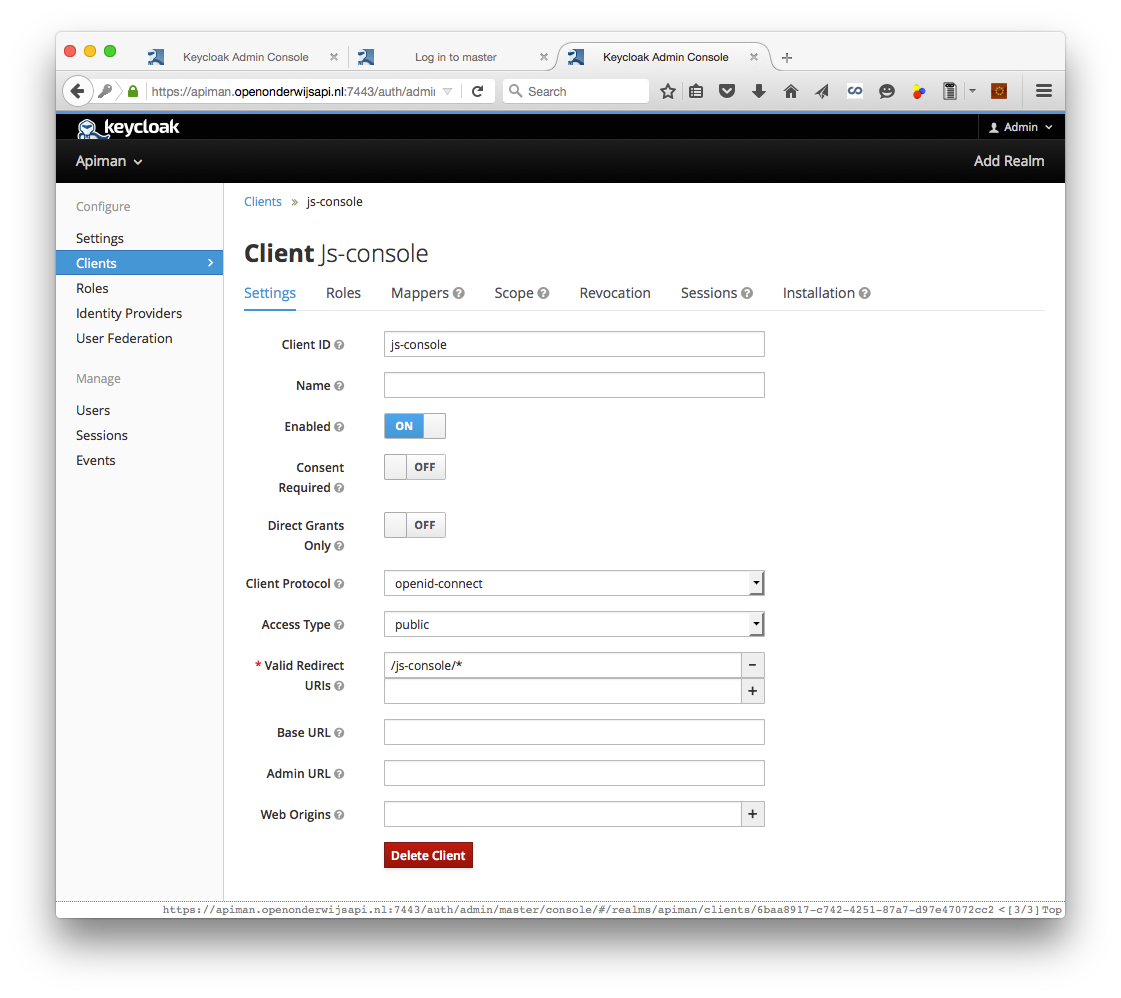


Het certificaat dient in Apiman te worden geconfigureerd bij de Keycloak OAuth Policy Configuration:



### Toevoegen js-console Client

In Keycloak dient een client te worden toegevoegd voor de js-console app. Daarbij moet het Acces Type op Public gezet worden, en de Valid Redirect URIs op /js-console/\*:



### Toevoegen Identity Provider

Per realm kunnen Identity Providers (IdP’s) worden ingesteld. In de context van dit project is SURFconext gebruikt, en dus een Saml-2 IdP. De Saml-2 metadata kan handmatig worden geconfigureerd, maar Keycloak kan de metadata ook vanaf een URL importeren. Voor elke geconfigureerde IdP wordt een knop toegevoegd aan het inlogscherm van de realm waarvoor de IdP geconfigureerd is. Dit is met name bij het testen van de configuratie handig.

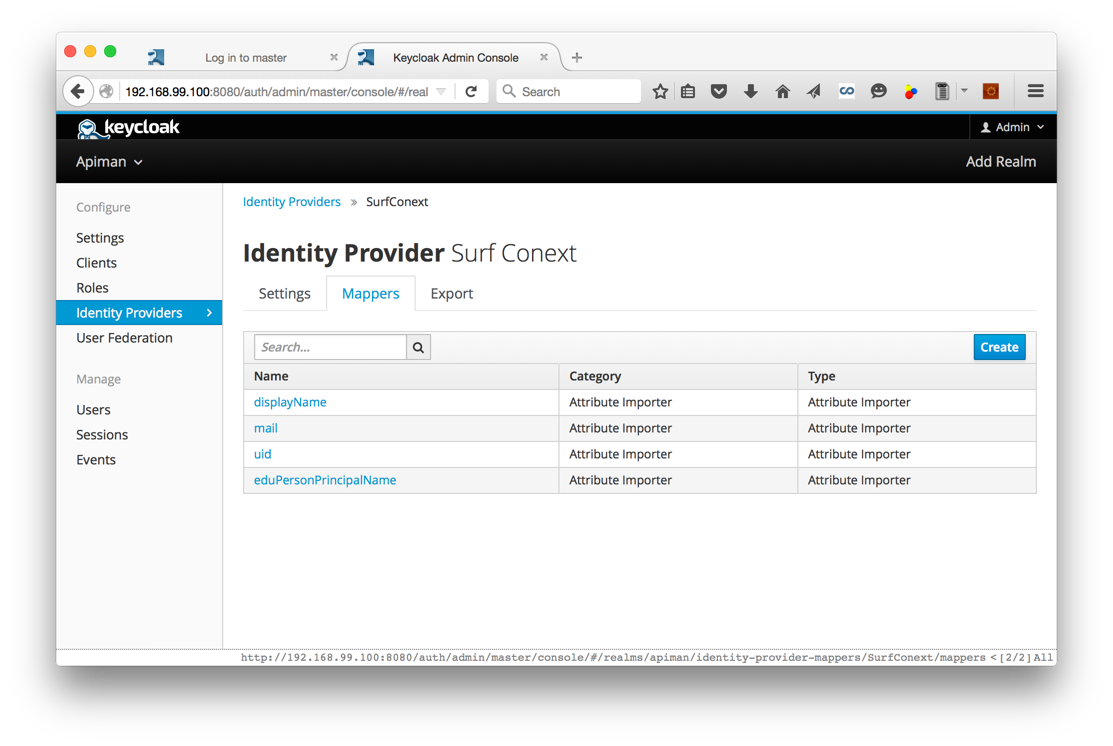
Aan de kant van SURFconext diende de metadata van de realm geconfigureerd te worden. De gebruikte versie van Keycloak biedt geen standaard URL voor het aanbieden van die metadata. Wel bleek er al een service te bestaan die deze data oplevert. Om deze via een URL beschikbaar te stellen is een kleine webapplicatie toegevoegd.

### Attribute Mapping

Om attributen die door een IdP worden doorgegeven op te nemen in een token is het nodig dat deze attributen zowel bij de IdP als bij de client geconfigureerd worden. De IdP mapping zorgt er voor dat attributen worden overgenomen uit de SAML assertions in het interne Keycloak user object. De client mapping maakt het mogelijk om attributen uit het user object op te nemen in het ID token en/of het access token. Daarnaast is het ook mogelijk om attributen naar rollen te mappen.

#### Configureren IdP Mappers

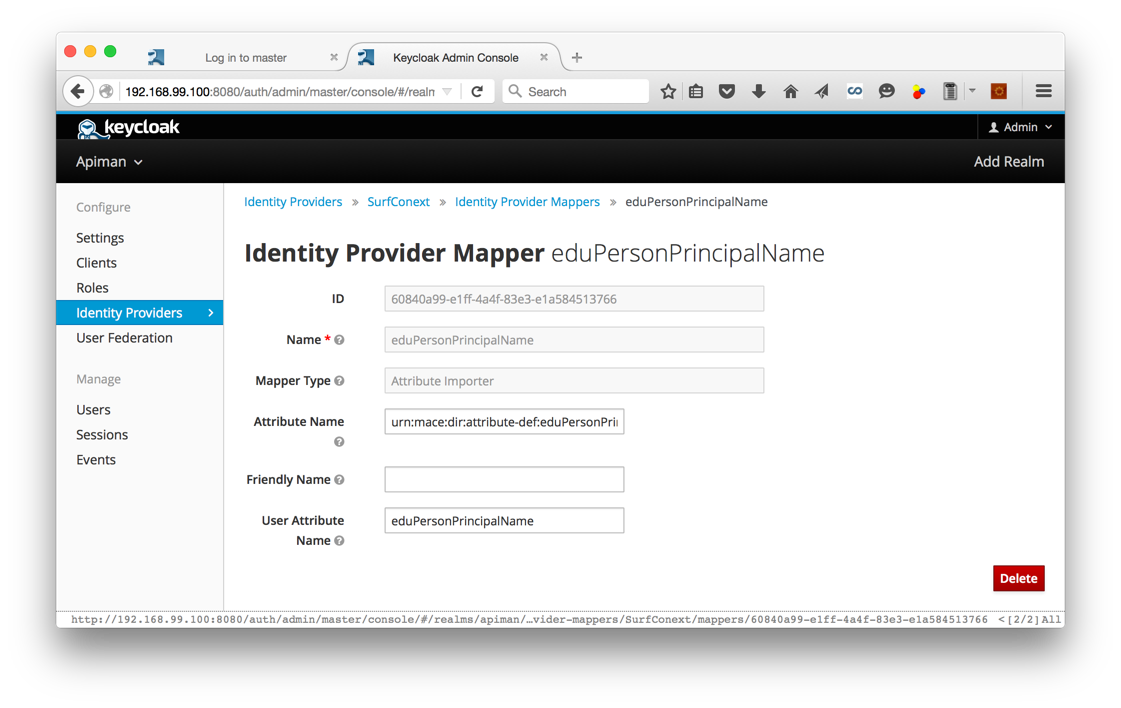
Default stelt Keycloak bin de IdP geen mappers in. Deze dienen dus te worden toegevoegd:



Bij de IdP worden de volgende SAML-attributen gemapt:

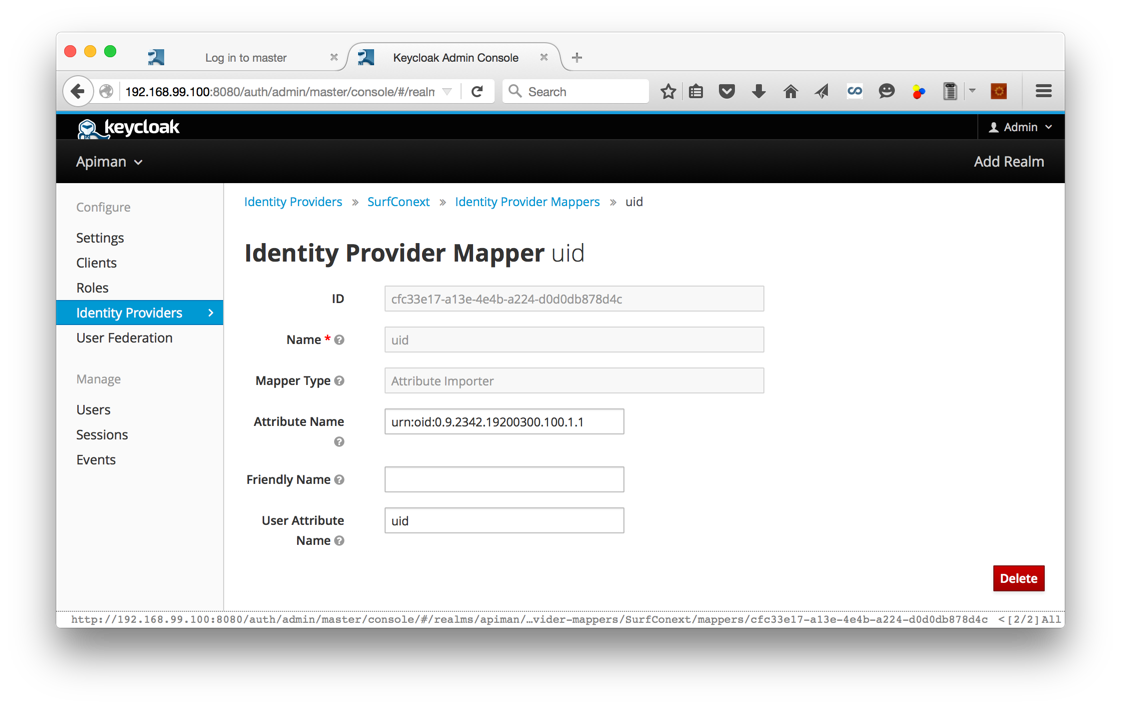
##### eduPersonPrincipalName

Gebruikersidentificatie inclusief het domein waarbinnen de gebruiker geauthentiseerd werd:



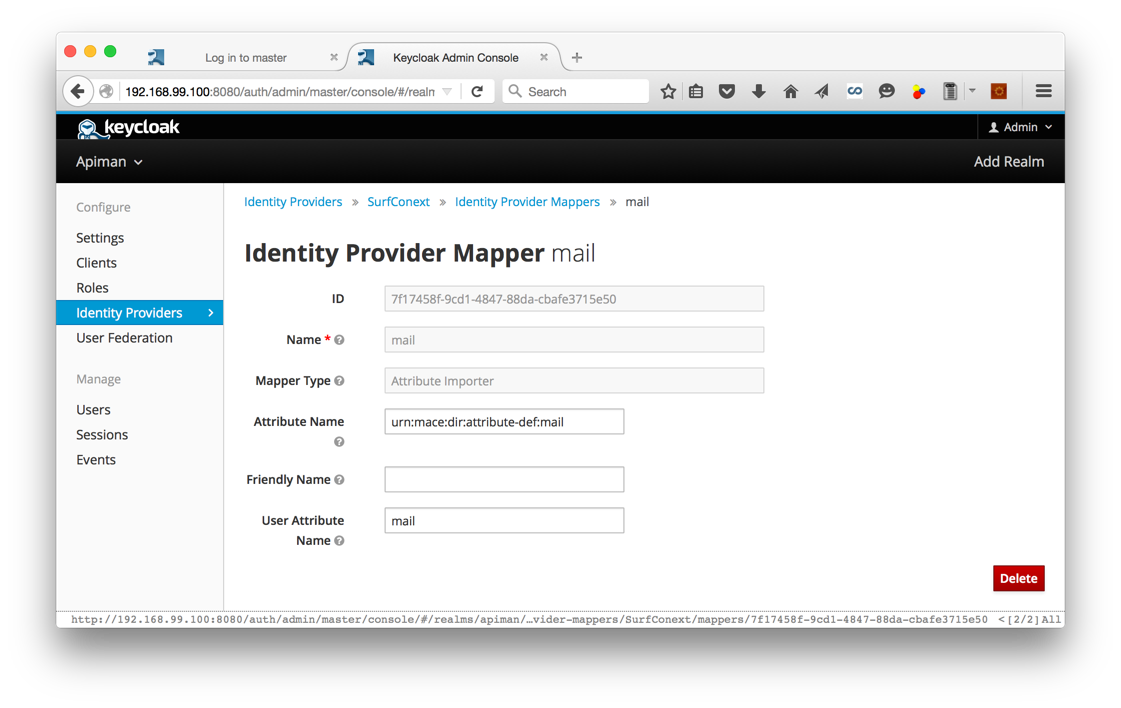
##### uid

Gebruikersidentificatie zonder het domein waarbinnen de gebruiker geauthentiseerd werd:



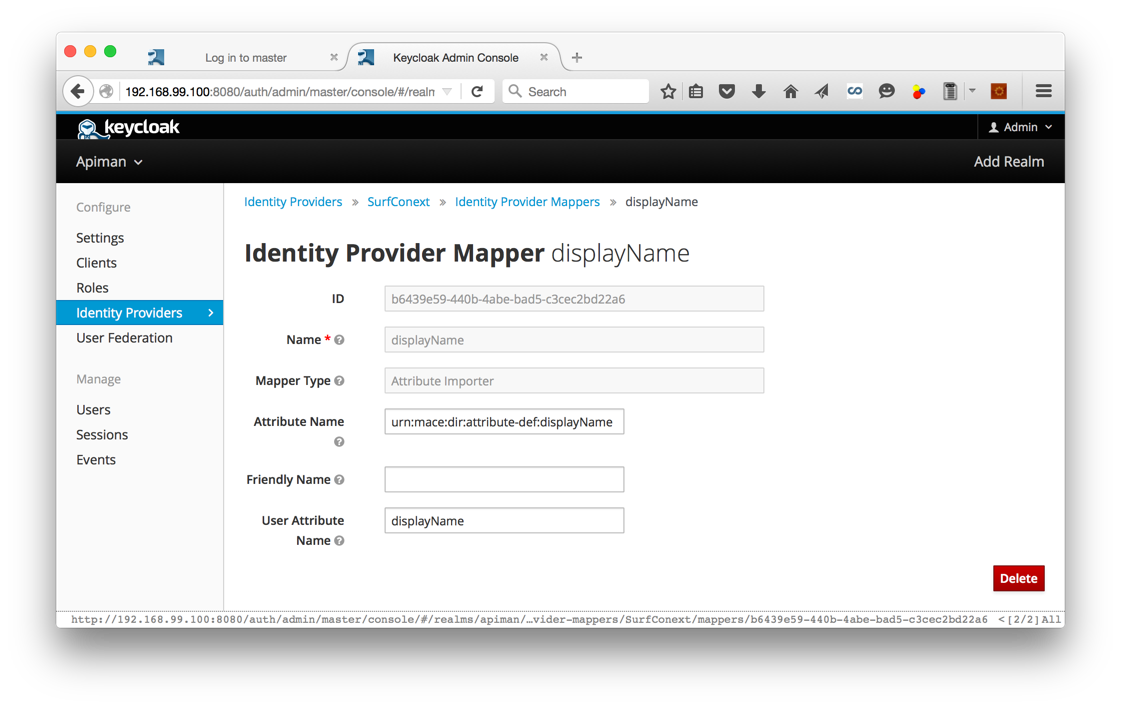
##### mail

Emailadres van de gebruiker zoals dat bij de IdP bekend is.



##### displayName

Te tonen naam van de gebruiker zoals dat bij de IdP bekend is.

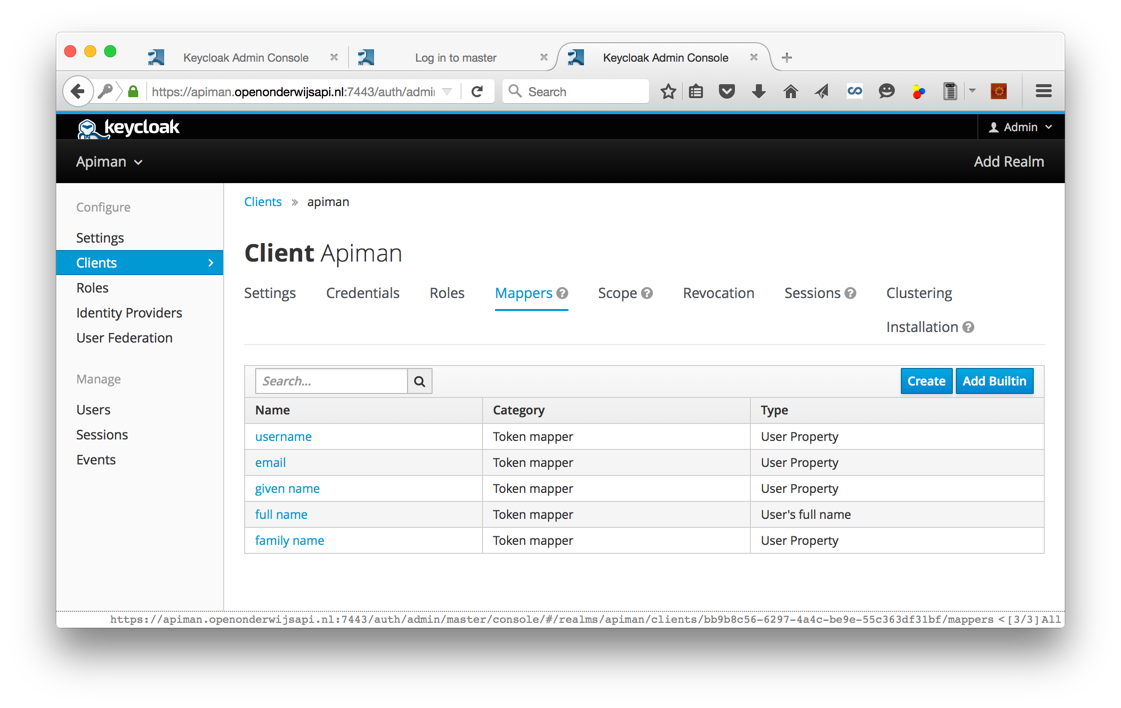


#### Configureren Client Mappers

Keycloak definieert default een aantal mappers die user properties mappen naar claims in de tokens. User properties maken deel uit van de UserModel interface van Keycloak. De waardes daarvan worden default door Keycloak ingesteld, maar zijn overschrijfbaar door attribuut mappers toe te voegen.

De default mappers zeten de volgende claims in het ID en het Acces Token:

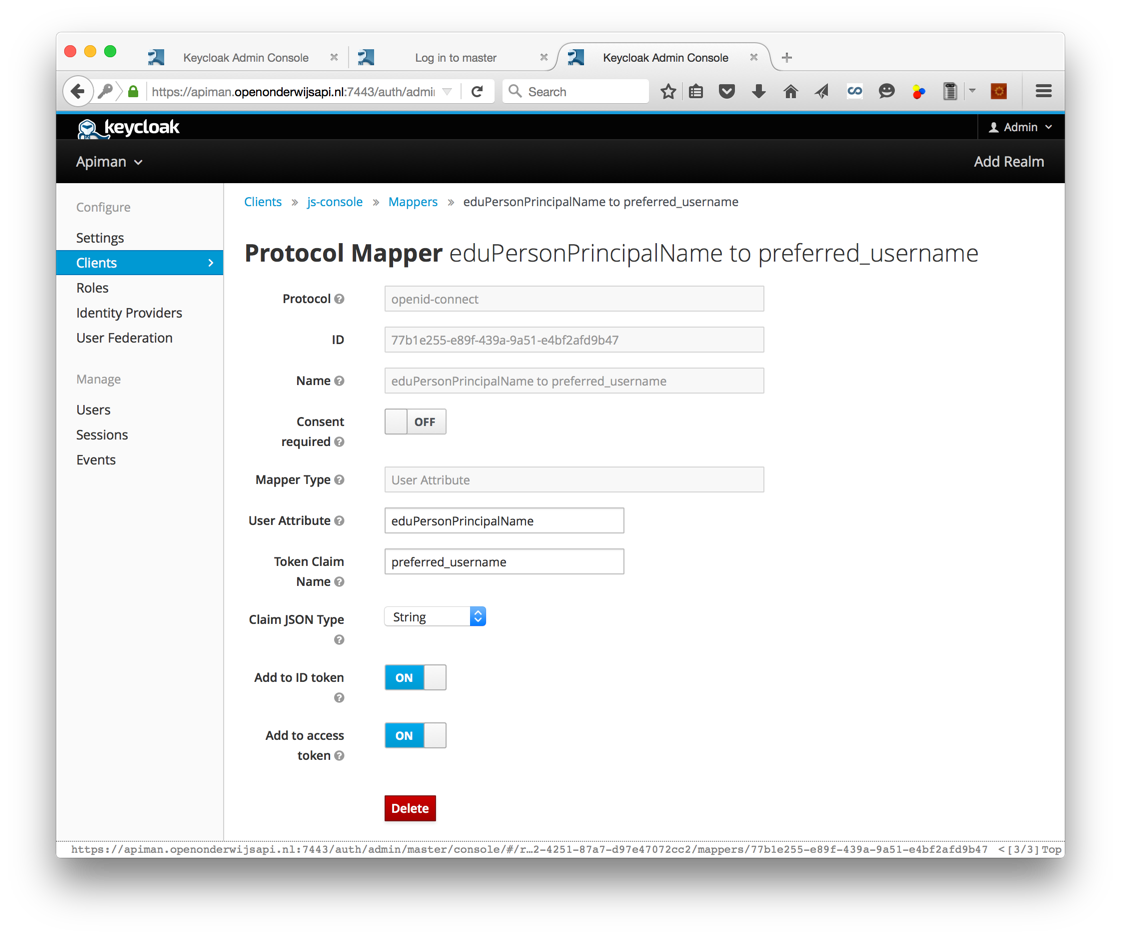
|  |  |
| --- | --- |
| **Claim** | **Waarde** |
| preferred\_username | User Property username |
| email | User Property email |
| given\_name | User Property firstName |
| name | User Properties firstName, lastName |
| family\_name | User Property lastName |



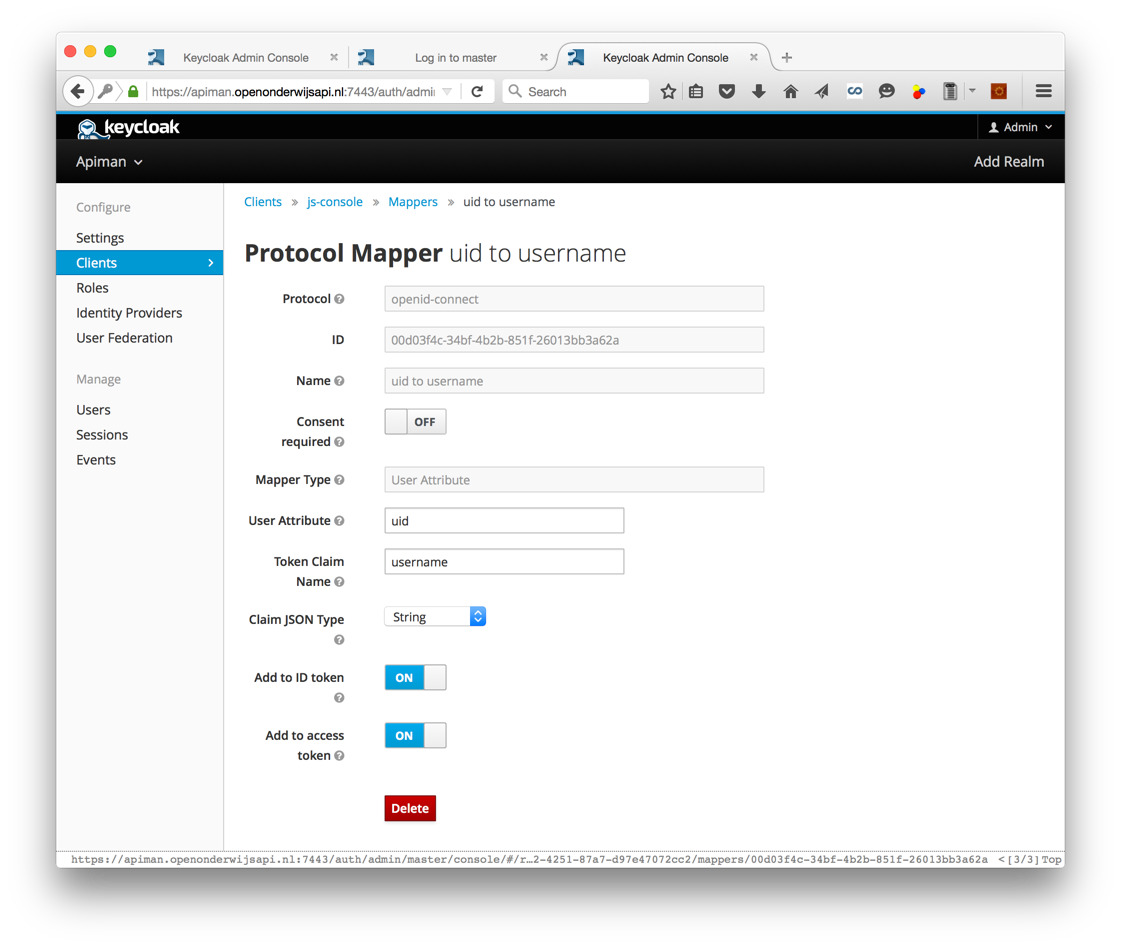
Aan deze default mappers zijn voor de client **js-console** de volgende mappers toegevoegd, deze overschrijven dus de claims die door de default mappers worden gezet:

|  |  |
| --- | --- |
| **Claim** | **Waarde** |
| preferred\_username | Attribuut eduPersonPrincipalName |
| username | Attribuut uid |
| email | Attribuut mail |

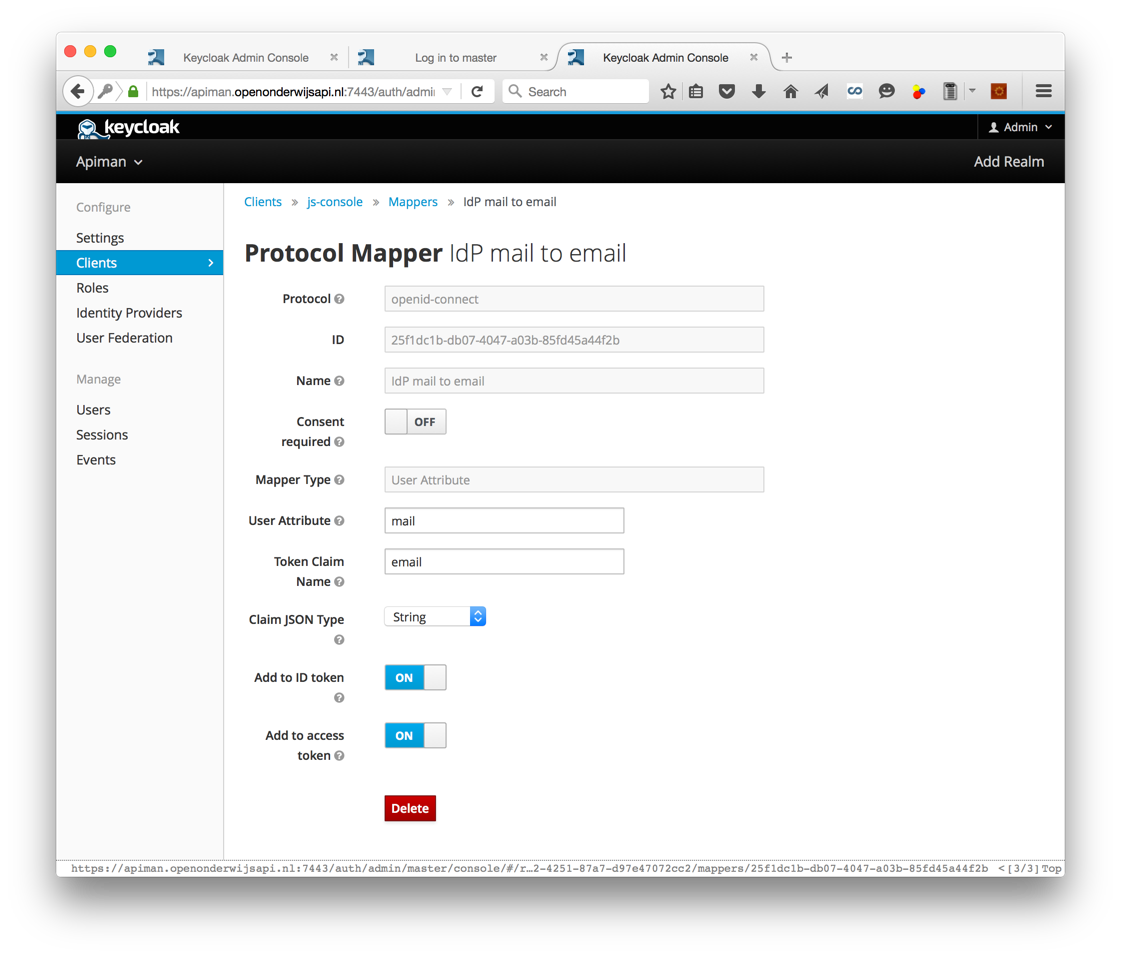
##### eduPersonPrincipalName to preferred\_username



##### uid to username

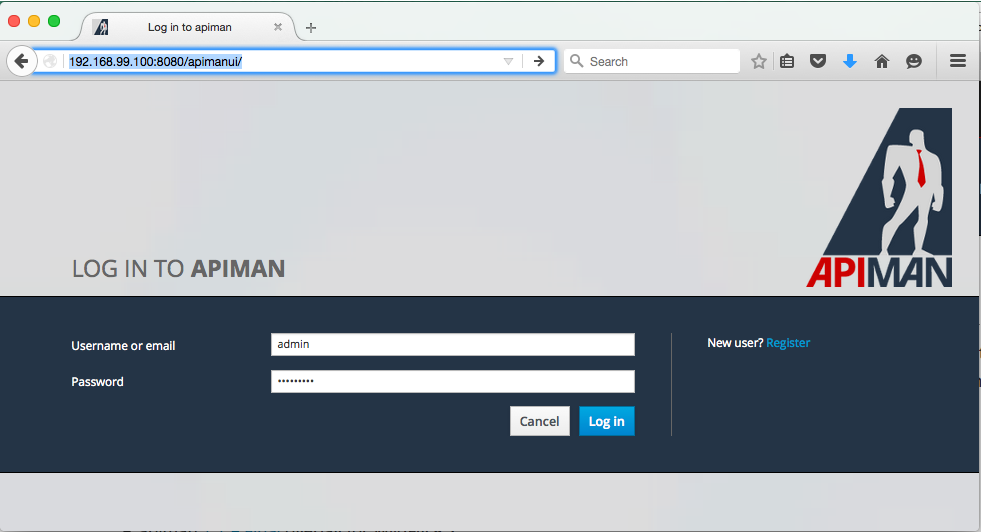


##### IdP mail to email

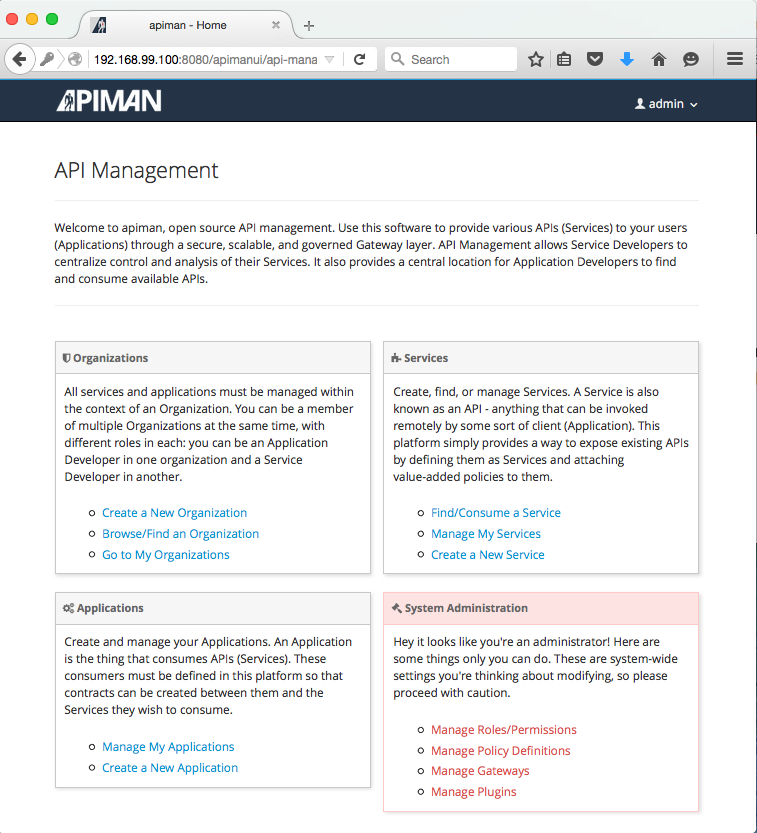


## Configuratie JBoss Apiman

Voor configuratie van JBoss Apiman kan men inloggen met de standaard username/wachtwoord admin/admin123! op de URL: <http://localhost:8080/apimanui/>.



Na het inloggen komt men in het overzicht scherm van JBoss Apiman. De beschikbare opties in de scherm zijn afhankelijk van de rollen en rechten.

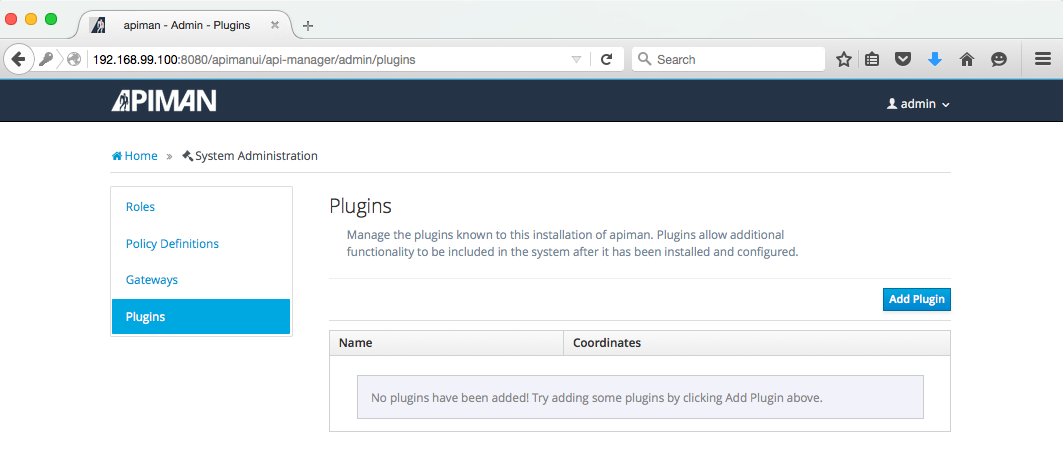


### Installatie extra plugins

JBoss Apiman biedt ondersteuning voor plugins voor het toevoegen van extra functionaliteit. Standaard zijn een aantal officiële plugins beschikbaar die niet meegeleverd worden met de standaard distributies, maar die men wel eenvoudig kan toevoegen.

Kies in het overzichtsscherm van JBoss Apiman voor *Manage Plugins*.

Het volgende scherm verschijnt:



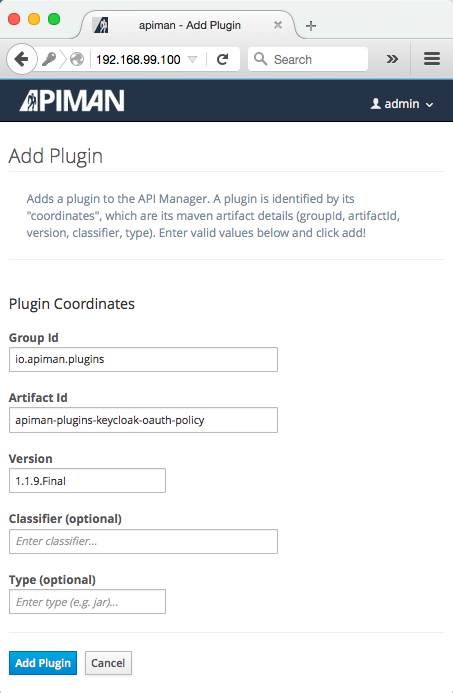
#### Installatie Keycloak OAuth2 Policy plugin

Kies in het Plugins scherm voor *Add Plugin*. Een formulier verschijnt welke als volgt moet worden ingevuld:

* Group Id : io.apiman.plugins
* Artifact Id : apiman-plugins-keycloak-oauth-policy
* Version : 1.1.9.Final

De overige velden worden leeg gelaten. Het *version* veld moet altijd gelijk zijn aan de versie van JBoss Apiman die gebruikt wordt!

Het ingevulde formulier ziet er dan als volgt uit:



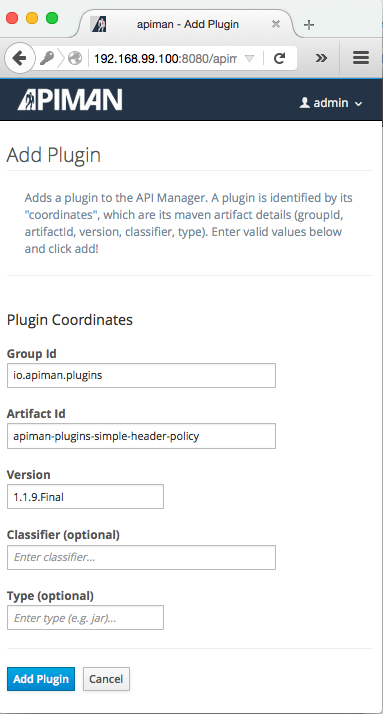
#### Installatie Simple Header Policy plugin

Kies in het Plugins scherm voor *Add Plugin*. Een formulier verschijnt welke als volgt moet worden ingevuld:

* Group Id : io.apiman.plugins
* Artifact Id : apiman-plugins-simple-header-policy
* Version : 1.1.9.Final

De overige velden worden leeg gelaten. Het *version* veld moet altijd gelijk zijn aan de versie van JBoss Apiman die gebruikt wordt!

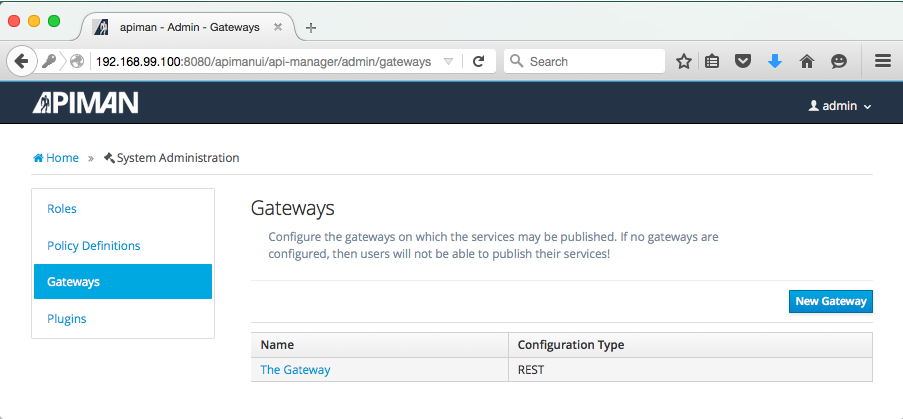
Het ingevulde formulier ziet er dan als volgt uit:



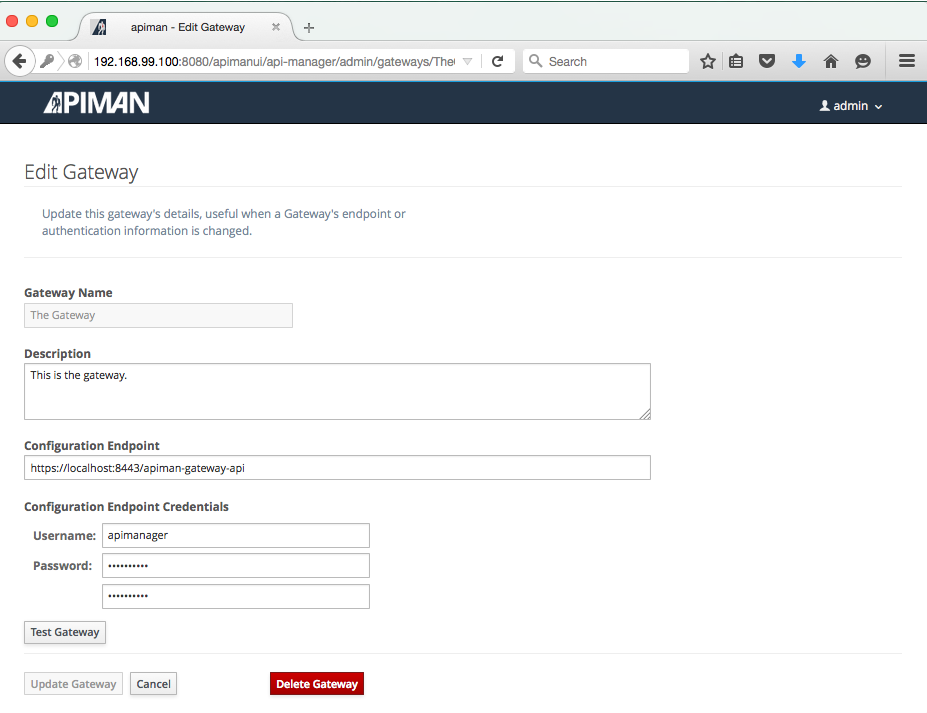
### Configureren Gateway Endpoint

Kies in het overzichtsscherm van JBoss Apiman voor *Manage Gateways*.

Het volgende scherm verschijnt:

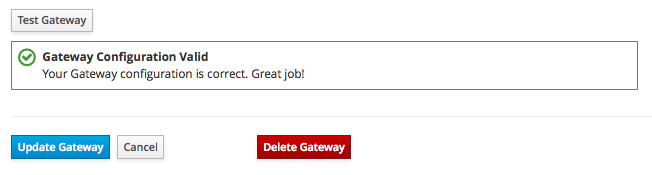


Kies in dit scherm voor *The Gateway*. Het volgende scherm verschijnt:



Verander in dit scherm de waarde van het veld *Configuration Endpoint* door de URL van <https://localhost:8443/apiman-gateway-api> aan te passen naar de externe URL van de demo omgeving, bijvoorbeeld: <https://apiman.openonderwijsapi.nl:7443/apiman-gateway-api>. Voor de meeste omgevingen volstaat het vervangen van localhost met de domainnaam van de omgeving. Tenzij in JBoss Wildfly de HTTPS binding op een andere poort beschikbaar wordt gesteld en/of er een reverse proxy voor de omgeving geplaatst wordt.

Kies vervolgens voor *Test Gateway*. Indien de URL valide en bereikbaar is verschijnt de melding:

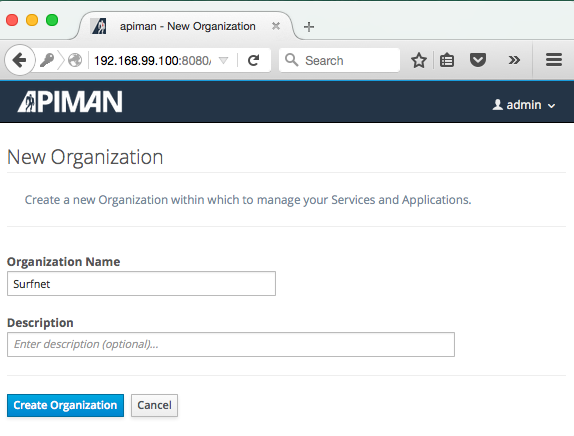


Pas zo nodig de URL aan voer de test opnieuw uit.

Kies vervolgens voor *Update Gateway* voor het definitief maken van de wijziging.

### Aanmaken nieuwe organisatie

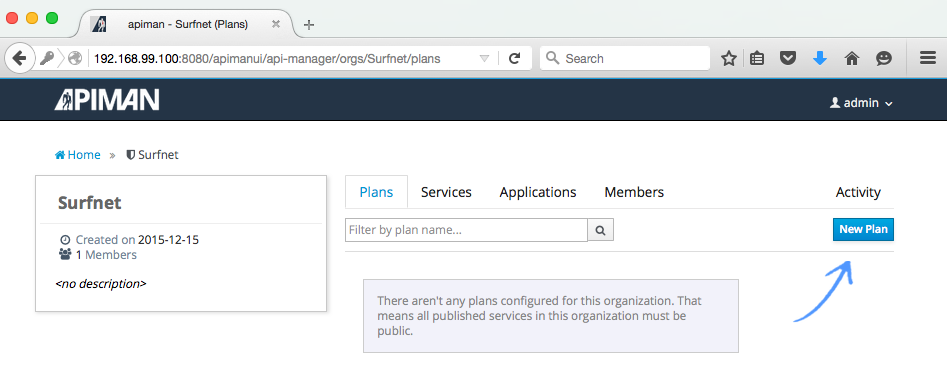
Kies in het overzichtsscherm van JBoss Apiman voor *Create a New Organization*. Het volgende scherm verschijnt:



Vul tenminste het veld *Organization Name* en optioneel een omschrijving.

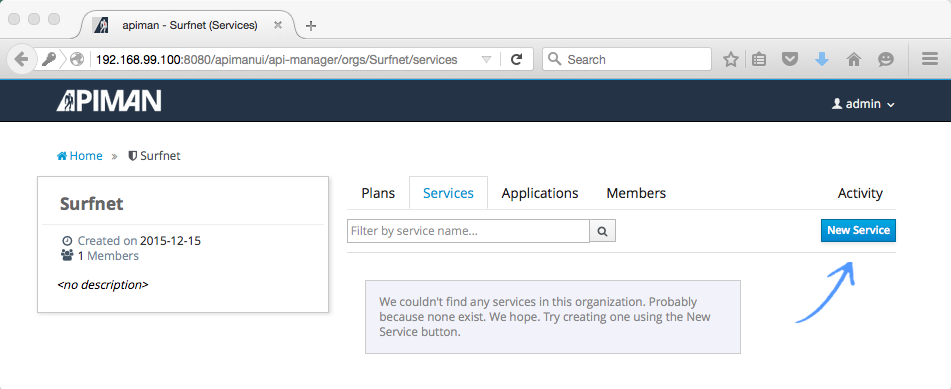
Kies vervolgens voor *Create Organization*.

Vervolgens verschijnt het overzichtsscherm van de aangemaakte organisatie. In deze handleiding is dit het overzichtsscherm van SURFnet:



#### Aanmaken nieuwe service

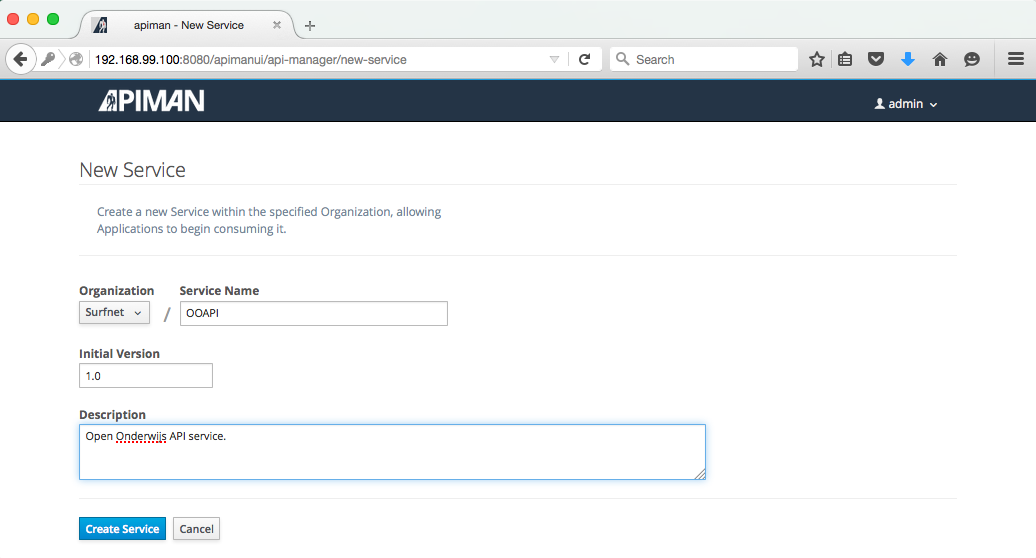
Kies in het overzichtsscherm van de organisatie voor het tabblad *Services*. Het volgende scherm verschijnt:



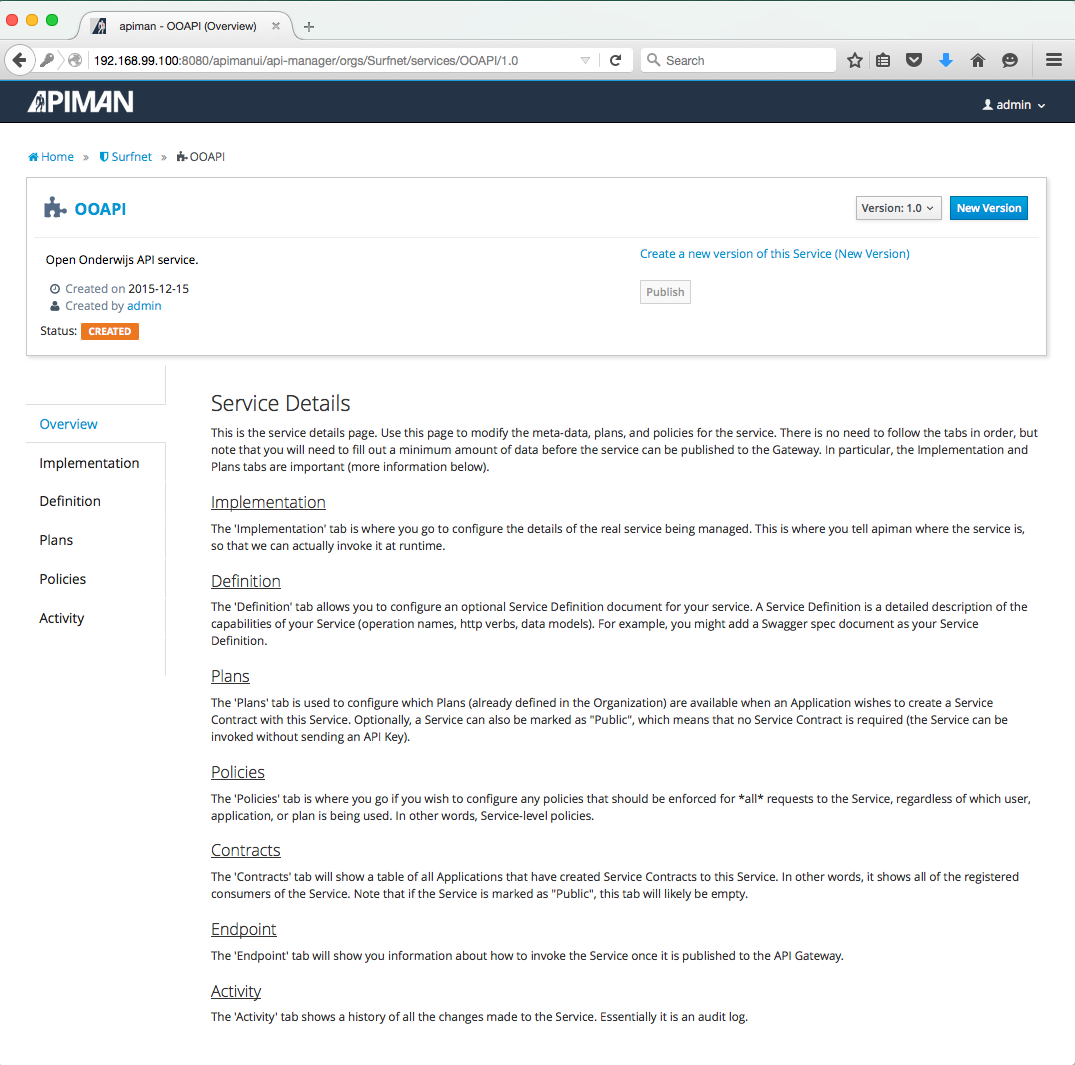
Kies vervolgens voor *New Service*. Er verschijnt een formulier die als volgt ingevuld moet worden:

* Service Name : De naam waaronder JBoss Apiman onze service extern beschikbaar stelt. Is onderdeel van de service URL.
* Initial Version : De begin versie van onze service. Is onderdeel van de service URL.

Het ingevulde formulier ziet er als volgt uit:



Kies vervolgens voor *Create Service*. Het overzichtsscherm van de zojuist aangemaakte service verschijnt waarna deze verder kan worden geconfigureerd.



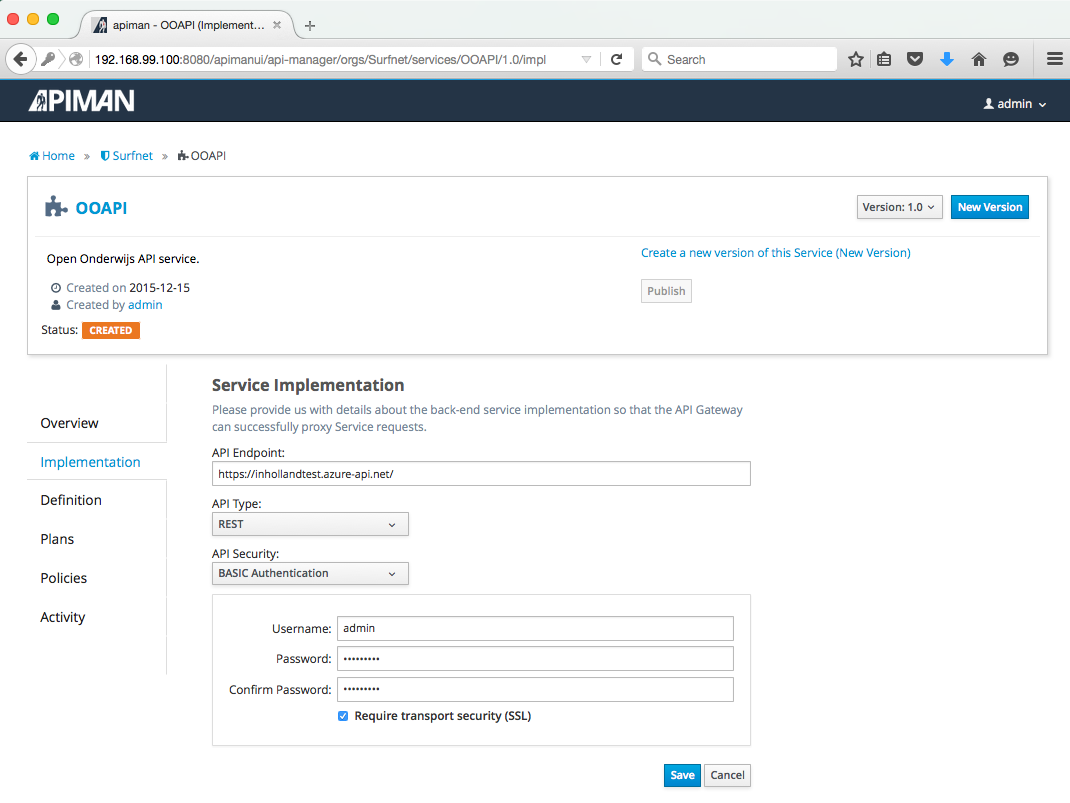
Alle configuratie werkzaamheden moeten uitgevoerd worden voordat een service wordt gepubliceerd. Na publicatie van een service zijn de eigenschappen van een service niet meer aan te passen. Het is wel mogelijk om een nieuwere versie van dezelfde service te maken.

#### Configureren Service Implementation

Kies in het overzichtsscherm van de service voor het tabblad *Implementation*. Er verschijnt een formulier dat als volgt ingevuld moet worden:

* API Endpoint : De URL naar de service die we onder API management willen brengen.
* API Type : Kies hier REST indien de service een REST service is en SOAP indien de service een SOAP service is.
* API Security : Kies hier het security mechanisme waarmee de service beveiligd is. Het betreft hier het security mechanisme waarmee JBoss Apiman toegang verkrijgt tot de service die onder API management gebracht wordt.  
  Configuratie van andere security mechanisme, zoals een lang levend OAuth2 token is te configureren door het toepassen van andere policies. Beschrijving hiervan is niet in scope van dit document!

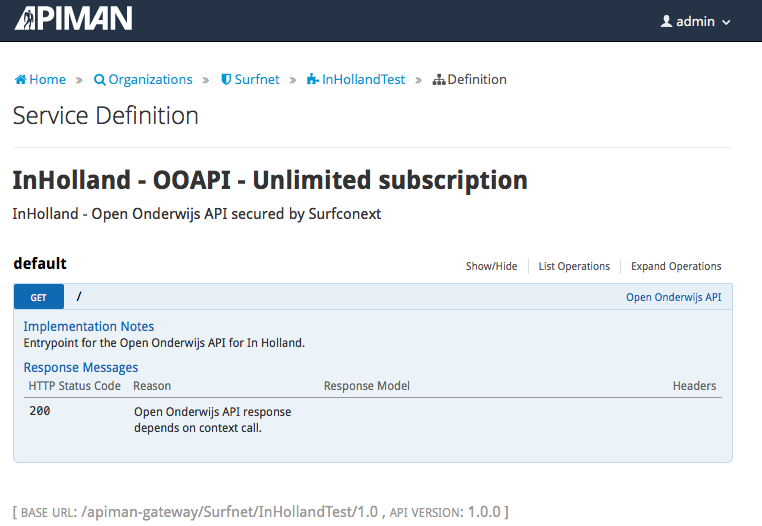
Een ingevuld formulier kan er als volgt uit zien:



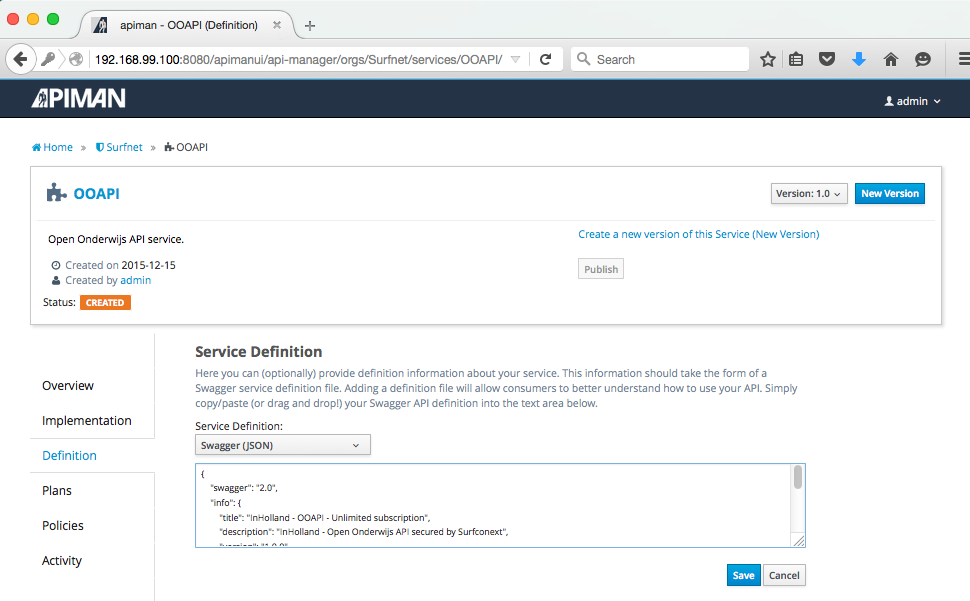
Kies voor *Save* om de wijzigingen op te slaan. Doe dit voordat je naar het volgende tabblad gaat anders gaan de wijzigingen verloren.

#### Toevoegen Swagger Documentatie (optioneel)

JBoss Apiman ondersteunt het toevoegen van een service definitie in Swagger formaat. Deze definitie dient als extra documentatie voor afnemers van services in de GUI van JBoss Apiman.



Kies in het overzichtsscherm van de service voor het tabblad *Definition*. Vervolgens verschijnt er een formulier waar men de mogelijkheid heeft om een service definitie toe te voegen in *Swagger(JSON)* of *Swagger (YAML)* formaat. De Swagger definitie kan afkomstig zijn van de service implementatie, waarna de environment specifieke zaken zoals *host* en *schemes* aangepast moeten worden aan de JBoss Apiman omgeving of kan met de hand opgesteld middels de Swagger editor. Zie [Swagger](http://swagger.io/) voor de mogelijkheden. Een ingevuld formulier ziet er als volgt uit:



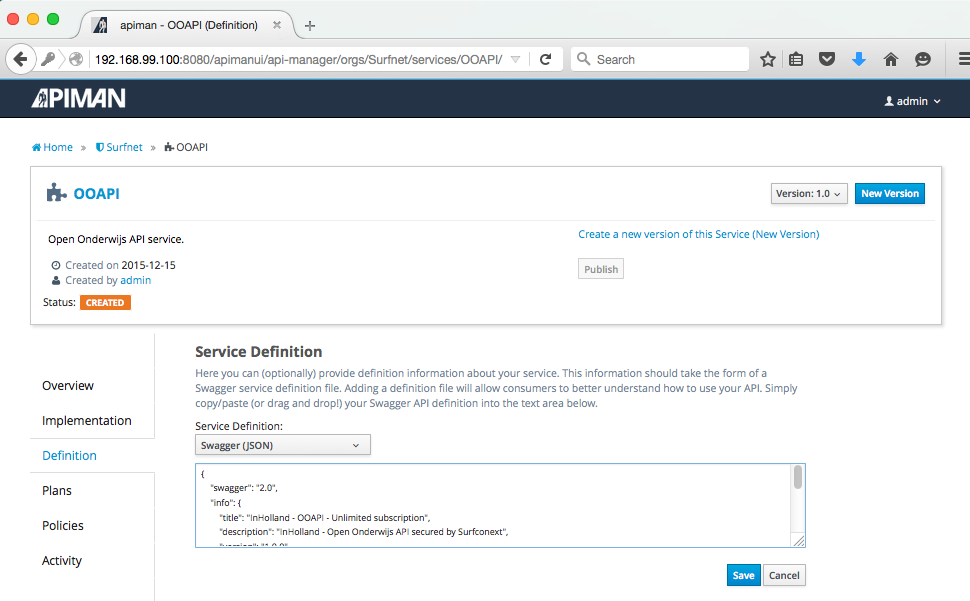
Kies vervolgens voor *Save* om de wijzigingen op te slaan.

#### Service publiek maken

Services in JBoss Apiman kunnen beschikbaar gesteld worden aan afnemers als een publieke service of als een service waarvoor SLA wordt afgesloten tussen applicatie (afnemer) en service (aanbieder) welke middels een API key geïdentificeerd wordt.

Voor de eenvoud van de demo omgeving wordt in deze installatiehandleiding uitgegaan van een publieke service. Het publiek maken van een service moet expliciet geconfigureerd worden.

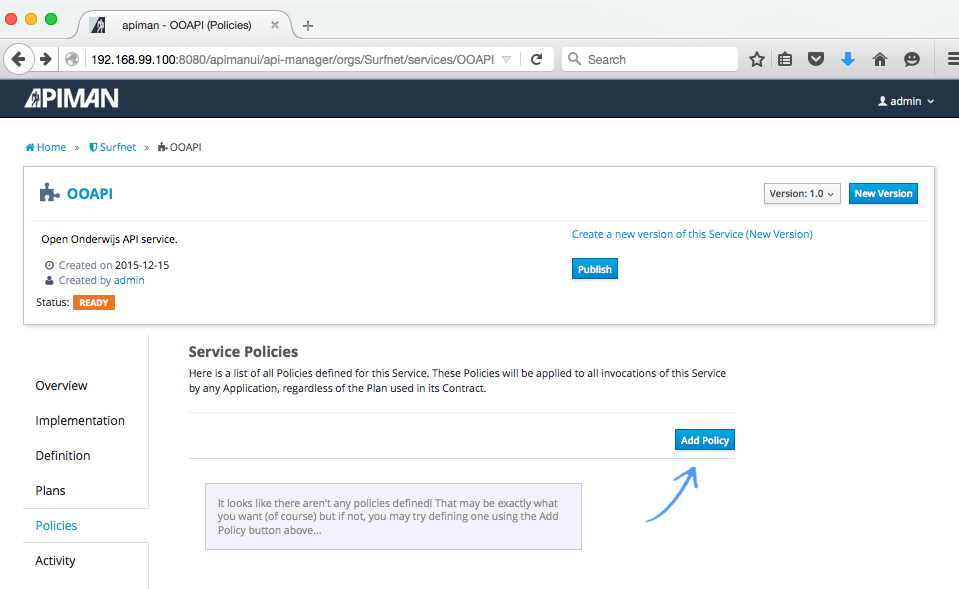
Kies in het overzichtsscherm van de service voor het tabblad *Plans*. Selecteer de optie *Make this service public* en kies vervolgens voor *Save* om de wijziging op te slaan.



#### Toevoegen van Policies

Via policies kan functionaliteit aan de door JBoss Apiman aangeboden services toegevoegd. Zowel standaard beschikbare functionaliteit alsmede extra functionaliteit doormiddel van plugins. Er zijn diverse policies beschikbaar voor security, caching, quota’s en throttling.

Kies in het overzichtsscherm van de service voor het tabblad *Policies* en vervolgens voor *Add Policy*.



Vervolgens verschijnt een formulier waarin de gewenste policy type gekozen kan worden.

#### Toevoegen Keycloak OAuth Policy

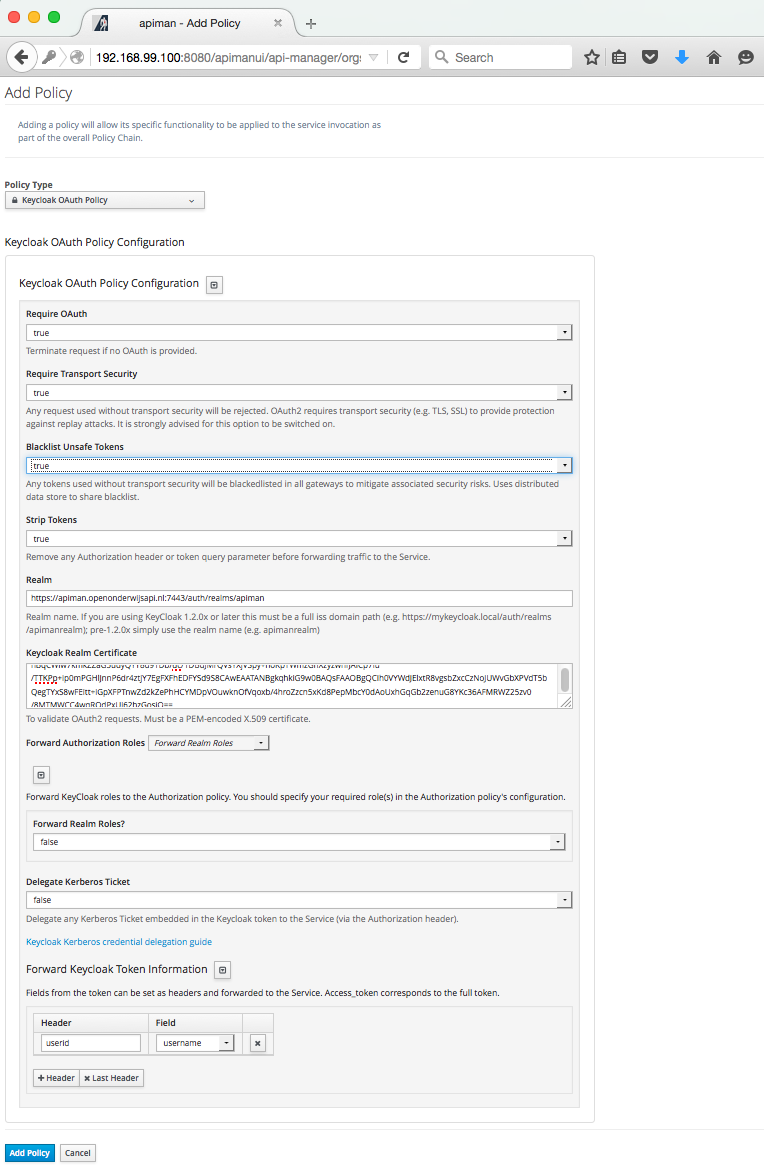
Voor autorisatie op basis van OAuth is de Keycloak OAuth policy beschikbaar. Deze policy verwacht een bearer token in de API call welke is uitgegeven door de Identity Provider JBoss Keycloak.

Kies in het overzichtsscherm van de service voor het tabblad *Policies* en vervolgens voor *Add Policy*. Vervolgens verschijnt er een formulier dat als volgt ingevuld moet worden:

* Policy Type : Keycloak OAuth Policy
* Realm : De locatie van de realm in JBoss Keycloak die gebruikt moet worden om de API calls te beveiligen. Bijvoorbeeld: <https://apiman.openonderwijsapi.nl:7443/auth/realms/apiman>
* Keycloak Realm Certificate : Het certificaat behorende bij het realm. Zie *Opvragen Realm Certificaat* bij de Configuratie van JBoss Keycloak.

De volgende velden kunnen optioneel ingevuld/aangepast worden afhankelijk van de omgeving:

* Blacklist Unsafe Tokens : Kies *true* voor verhoogde beveiliging.
* Strip Tokens : Kies hier altijd *true* tenzij de service implementatie dusdanig geconfigureerd is dat deze ook overweg kan met het OAuth token uitgegeven door JBoss Keycloak. De meeste service implementaties hebben hun eigen authenticatie mechanisme waardoor het kiezen van *true* in de meeste gevallen de juiste keuze is.
* Forward Keycloak Token Information : Forward specifieke onderdelen uit het OAuth token via een HTTP header naar service implementatie. Bijvoorbeeld voor identificatie van de gebruiker op basis username, e-mail, name.



Kies voor *Add Policy* om de policy toe te voegen.

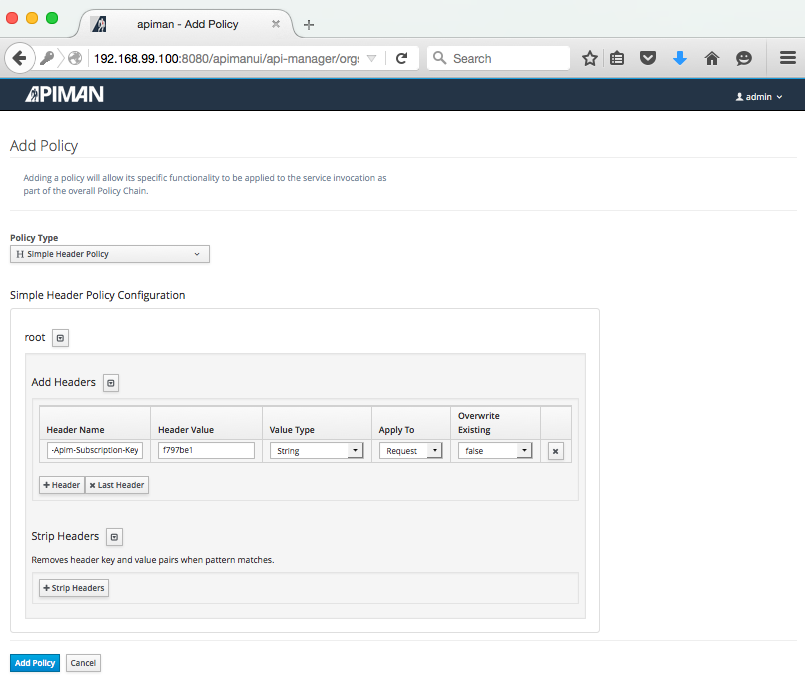
#### Toevoegen Simple Header Policy

Het is mogelijk om extra headers toe te voegen alvorens een HTTP request vanuit JBoss Apiman naar de service implementatie gestuurd wordt en/of alvorens JBoss Apiman het resultaat van de service implementatie teruggestuurd naar de client.

Voor de service van In Holland wordt hiermee de verplichte subscription key meegestuurd.

Kies in het overzichtsscherm van de service voor het tabblad *Policies* en vervolgens voor *Add Policy*. Vervolgens verschijnt er een formulier dat als volgt ingevuld moet worden:

* Policy Type : Simple Header Policy
* Add Headers:
  + Header Name : De naam van de HTTP header die naar de service gestuurd wordt;
  + Header Value : De waarde van de HTTP header die naar de service gestuurd wordt;
  + Value Type : De oorsprong van de header value. Gebruik *String* voor een HTTP header met een vaste waarde.
  + Apply To : Kies *Request* om de header naar de service implementatie te sturen. Kies *Response* om de header de applicatie te sturen welke de API call heeft geïnitieerd.
  + Overwrite Existing : Overschrijf de header indien deze al aanwezig is.

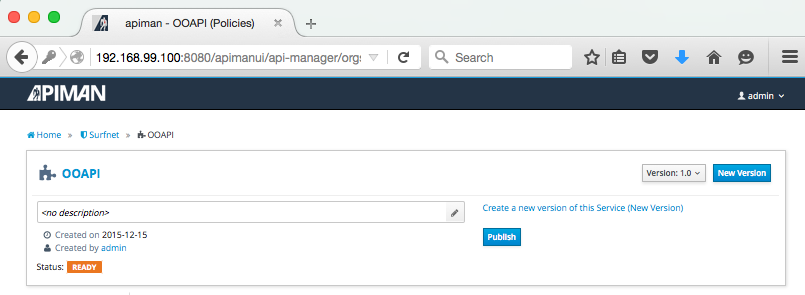


Kies voor *Add Policy* om de policy toe te voegen.

### Publiceren service

Nadat de service volledig is geconfigureerd kan deze gepubliceerd worden. Pas na publicatie is de service beschikbaar voor applicaties. Na publicaties is een service definitief. Wijzigingen aan een service kunnen alleen doorgevoerd worden door een nieuwe versie van de service te maken / kopiëren en deze te publiceren.

Kies in het overzichtsscherm van de service voor *Publish*.



Na publicatie komt er onder ander een tabblad *Endpoint* beschikbaar. Kies het tabblad *Endpoint*. Dit tabblad toont de URL waaronder de API door JBoss Apiman beschikbaar gesteld wordt.

