

---

# **KeyRock Documentation**

***Release 6.0.0***

**Universidad Politecnica de Madrid**

**Mar 14, 2019**



---

## Installation and Administration

---

<b>1</b>	<b>Important information</b>	<b>3</b>
<b>2</b>	<b>Identity Manager - Keyrock Overview</b>	<b>5</b>
<b>3</b>	<b>Installation and Administration Guide</b>	<b>11</b>
<b>4</b>	<b>Production Set Up Guide</b>	<b>21</b>
<b>5</b>	<b>Private Clouds Federation</b>	<b>27</b>
<b>6</b>	<b>Using the FIWARE LAB instance (OAuth2)</b>	<b>31</b>
<b>7</b>	<b>Two Factor Authentication</b>	<b>37</b>
<b>8</b>	<b>User and Programmers Guide</b>	<b>39</b>
<b>9</b>	<b>Endpoints Management Dashboard (admin-only)</b>	<b>49</b>
<b>10</b>	<b>Developers and contributors Guide</b>	<b>55</b>



This project is part of [FIWARE](#). You will find more information about this FIWARE GE at the [Catalogue](#).



# CHAPTER 1

---

## Important information

---

This is an archive of the code repository for the previous Horizon + Keystone-based version of the FIWARE Identity Manager project. The current code repository for this project can be found at: <https://github.com/ging/fiware-idm>





---

### Identity Manager - Keyrock Overview

---

- *Introduction*
  - *Requirements*
- *How to Build & Install*
  - *Installing the back-end*
  - *Installing the front-end*
  - *Other Installation options*
    - \* *Docker*
    - \* *VM Image*
    - \* *Chef*
- *API Overview*
- *Changes introduced in 5.x*
- *Advanced Documentation*

## 2.1 Introduction

This project is part of [FIWARE](#). You may find more information about this FIWARE GE [here](#).

- You may find the source code of this project in GitHub [here](#)
- You may find the documentation of this project in Read the Docs [here](#)

Welcome to the main repository for the UPM's implementation of the FIWARE Identity Manager Generic Enabler. This repository acts as an entry point and holds the documentation and some automated tools for installation and management. The IdM is composed of two independent components: a RESTful back-end and web front-end.

If you want to see the code for each of the components of the IdM and more specific documentation please head to each component's repository:

- Horizon based front-end [ging/horizon](#)
- Keystone based back-end [ging/keystone](#)

You can see a working installation in the FIWARE Lab sandbox environment <https://account.lab.fiware.org/>

## 2.1.1 Requirements

Identity Manager - KeyRock requires Ubuntu 12.04 or greater.

Both Horizon, for the front-end, and Keystone, for the back-end, must be installed in order for the generic enabler to run correctly. They can be installed in the same machine or in two separated ones. If you choose to separate them, the two machines must be able to communicate to each other through the network.

## 2.2 How to Build & Install

The IdM is made up of two components: the web-based front-end and the restful back-end. You can check specific documentation in their respective repositories.

### 2.2.1 Installing the back-end

1. Install the Ubuntu dependencies

```
$ sudo apt-get install python python-dev python-virtualenv libxml2-dev_  
↪ libxslt1-dev libsasl2-dev libssl-dev libldap2-dev libffi-dev libsqlite3-  
↪ dev libmysqlclient-dev python-mysqldb
```

2. Get the code from our [GitHub repository](#)

```
$ git clone https://github.com/ging/keystone && cd keystone
```

3. Install the python dependencies

```
$ sudo python tools/install_venv.py
```

4. Create a configuration file

```
$ cp etc/keystone.conf.sample etc/keystone.conf
```

5. Create the tables and populate the database

```
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=oauth2  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=roles  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=user_  
↪ registration  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=two_  
↪ factor_auth
```

(continues on next page)

(continued from previous page)

```
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --  
↪extension=endpoint_filter  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --populate
```

6. Finally, you can run keystone from the console

```
$ sudo tools/with_venv.sh bin/keystone-all -v
```

You may now log into the web (if you have Horizon installed) using the administrative account (by default, user is *idm* and the password is the one you entered during the populate step).

---

**Note:** If you want to run the Keystone backend in the background you can *install it as a service*.

---

Now, head on to the [configuration instructions](#).

(You can read more in-depth documentation at the [Installation & Administration Guide](#))

## 2.2.2 Installing the front-end

1. Install the Ubuntu dependencies

```
$ sudo apt-get install python python-dev python-virtualenv libssl-dev libffi-  
↪dev libjpeg8-dev
```

2. Get the code from our [GitHub repository](#)

```
$ git clone https://github.com/ging/horizon && cd horizon
```

3. Create a configuration file

```
$ cp openstack_dashboard/local/local_settings.py.example openstack_dashboard/  
↪local/local_settings.py
```

4. Install the python dependencies

```
$ sudo python tools/install_venv.py
```

You can now check that everything went OK by running the development server, but you won't be able to log in until you install the backend.

```
$ sudo tools/with_venv.sh python manage.py runserver localhost:8000
```

---

**Note:** If you want to run the Horizon frontend in the background you can install it as a service or, for a production environment, run it under Apache.

---

Now, head to the [configuration instructions](#).

(You can read more in-depth documentation at the [Installation & Administration Guide](#))

## 2.2.3 Other Installation options

## Docker

We provide a Docker image to facilitate you the building of this GE.

- [Here](#) you will find the Dockerfile and the documentation explaining how to use it.
- In [Docker Hub](#) you will find the public image.

## VM Image

We provide an installation script that can be run within a Ubuntu virtual machine. This script installs the whole Identity Manager, and sets it up to run in background.

You can find the installation script and a verification script [here](#).

This image contains the following settings as defaults, but you can change any of them after installation, as you can see in the [horizon](#) and the [keystone](#) configuration instructions:

Setting	Value
idm user	i dm
idm password	i dm
Horizon port	8000
Keystone port	5000

## Chef

We also provide a Chef Cookbook, which you can find [here](#).

## 2.3 API Overview

Keyrock back-end is based on Openstack [Keystone](#) project, so it exports all the Keystone API. However, Keyrock implements some custom extensions that have their own REST APIs. Furthermore, to facilitate the access to some identity resources we have enabled an [SCIM 2.0](#) API.

Finally, one of the main uses of Keyrock is to allow developers to add identity management (authentication and authorization) to their applications based on FIWARE identity. This is posible thanks to [OAuth2](#) protocol.

- [Keystone API](#)
- [Keyrock extensions API](#)
- [SCIM 2.0 API](#)
- [OAuth2 API](#)

You will find the full API description [here](#)

## 2.4 Changes introduced in 5.x

This section is for users of the 4.x versions. They biggest change introduced in 5.x is the removal Fabric tasks. The functionality in the tasks has been moved elsewhere, converted to a CLI or removed completely.

- A new CLI tool to help with admin tasks. Documentation [here](#)

- The installation now is always done step by step.
- The population script for the keystone database is now part of keystone.

Check the release notes for a full list of changes and new features.

## 2.5 Advanced Documentation

- [User & Programmers Manual](#)
- [Installation & Administration Guide](#)
- [Production set-up guide](#)
- [How to run tests](#)
- [Using the FIWARE LAB instance \(OAuth2\)](#)
- [Developers and contributors Guide](#)



- *Introduction*
  - *Requirements*
- *Step by Step Installation*
  - *Installing Horizon*
    - \* *1. Installation*
    - \* *2. Configuration*
    - \* *3. Django settings*
    - \* *4. Running a development server*
  - *Installing Keystone*
    - \* *1. Installation*
    - \* *2. Configuration*
    - \* *3. Run Keystone*
    - \* *4. Configuring Keystone as a service*
    - \* *5. Running tests*
- *System Administration*
  - *CLI tools*
  - *White and black lists*
- *Sanity Check Procedures*
  - *End to End testing*
  - *List of Running Processes*

- *Network interfaces Up & Open*
- *Databases*
- *Diagnosis Procedures*
  - *Resource availability*
  - *Remote Service Access*
  - *Resource consumption*
  - *I/O flows*

## 3.1 Introduction

Welcome to the Installation and Administration Guide for the Identity Management - KeyRock Generic Enabler. This section will cover how to install, configure and administrate a working instance of KeyRock.

If you want to deploy it in a production environment, take a look at the *Production set up Guide*.

### 3.1.1 Requirements

Identity Manager - KeyRock requires Ubuntu 12.04 or greater.

Both Horizon, for the front-end, and Keystone, for the back-end, must be installed in order for the generic enabler to run correctly. They can be installed in the same machine or in two separated ones. If you choose to separate them, the two machines must be able to communicate to each other through the network.

## 3.2 Step by Step Installation

The IdM is composed of two separated services, that interact with each other. The web portal is based on OpenStack's Dashboard, Horizon. The back-end is a REST service based on OpenStack's Identity Provider, Keystone.

They can be installed both on the same machine (or docker container) or in separated ones. If separated machines is the preferred option, make sure there is connectivity between them, as Horizon needs to be able to consume Keystone's REST API.

---

**Note:** To be able to log into the IdM, you will need a working Keystone backend. Please complete all the steps in this page in order to have a complete and working IdM.

---

### 3.2.1 Installing Horizon

#### 1. Installation

1. Install the Ubuntu dependencies

```
$ sudo apt-get install python python-dev python-virtualenv libssl-dev libffi-  
→dev libjpeg8-dev
```

2. Get the code from our [GitHub repository](#)



```
$ git clone https://github.com/ging/horizon && cd horizon
```

### 3. Create a configuration file

```
$ cp openstack_dashboard/local/local_settings.py.example openstack_dashboard/
↪ local/local_settings.py
```

### 4. Install the python dependencies

```
$ sudo python tools/install_venv.py
```

You can now check that everything went OK by running the development server, but you won't be able to log in until you install the backend.

```
$ sudo tools/with_venv.sh python manage.py runserver localhost:8000
```

**Note:** If you want to run the Horizon frontend in the background you can install it as a service or, for a production environment, run it under Apache.

## 2. Configuration

To configure Horizon, the configuration file can be found in `openstack_dashboard/local/local_settings.py`. This file holds sensible defaults for a common installation but you might need to tweak them to fit your use case.

If you are running Keystone on your own machine the address will be `http://localhost:5000/v3`. If Keystone is configured to run on a different port and/or address you should set this accordingly.

```
OPENSTACK_HOST = "Keystone server IP address"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

## Email

Configure this for your outgoing email host or leave the default values for the console email backend. More details on how to configure this can be found in the [Django docs](#) and in the *Production Set up Guide*.

## Keystone Account for the IdM to perform tasks like user registration

```
OPENSTACK_KEYSTONE_ADMIN_CREDENTIALS = {
    'USERNAME': 'the_username',
    'PASSWORD': 'the_password',
    'PROJECT': 'the_projectname',
}
```

## User Registration settings

This setting enables email domain filtering on user registration. Set to 'whitelist', 'blacklist' or comment it out for no filtering.

```
EMAIL_LIST_TYPE = 'blacklist'
```

More info [here](#).

### noCAPTCHA reCAPTCHA

---

**Note:** If you want to disable the captcha, set `USE_CAPTCHA` to `False`.

---

**Warning:** Don't deploy KeyRock in a public domain with CAPTCHA disabled.

Get your keys [here](#). More documentation in [the captcha package repository](#).

```
USE_CAPTCHA = False
NORECAPTCHA_SITE_KEY = '6LeIxAcTAAAAAJcZVRqyHh71UMIEGNQ_MXjiZKhI'
NORECAPTCHA_SECRET_KEY = '6LeIxAcTAAAAAGG-vF1lTnRWxMZNfUojJ4WifJWe'
```

### FIWARE Applications and Roles

These settings map applications used in the FIWARE-Lab environment and are needed for automated tasks, for example granting the **Purchaser** role in the **Store** to any created organization. Depending on your use case you might need or want to modify them, but normal installations in a *fiware-like* environment won't need to change the following code. Keep in mind that if your use case differs too much you might need to change the code to prevent some of these operations. If you are not using the scripts you will need to check the ids in through the API or in the database yourself.

```
FIWARE_PURCHASER_ROLE_ID = 'id'
FIWARE_PROVIDER_ROLE_ID = 'id'
FIWARE_IDM_ADMIN_APP = 'idm'
FIWARE_CLOUD_APP = 'Cloud'
FIWARE_DEFAULT_CLOUD_ROLE_ID = 'id'
FIWARE_DEFAULT_APPS = [
    'Store',
]
```

### Keystone roles

These settings map to normal Keystone roles that are used by the IdM. As with the FIWARE Applications and Roles settings, they depend on your use case and , if you are not using the installation scripts, you will have to create them yourself.

```
KEYSTONE_OWNER_ROLE = 'owner'
KEYSTONE_TRIAL_ROLE = 'trial'
KEYSTONE_BASIC_ROLE = 'basic'
KEYSTONE_COMMUNITY_ROLE = 'community'
MAX_TRIAL_USERS = 100
OPENSTACK_KEYSTONE_ADMIN_ROLES = [
    KEYSTONE_OWNER_ROLE,
```

(continues on next page)

(continued from previous page)

```
'admin',
]
```

## AuthZForce GE Configuration

These settings configure the connection to an [Authorization PDP GE](#) instance to create permissions to your applications. If the AZF instance is secured by a [PEP Proxy GE](#) you can also set a magic key to bypass the policy enforcement point.

```
# ACCESS CONTROL GE
ACCESS_CONTROL_URL = 'http://azf_host:6019'
ACCESS_CONTROL_MAGIC_KEY = 'azf_pep_key'
```

## Endpoints Management Dashboard

This admin-only dashboard requires some settings before it can be used. The Keystone project to which all services accounts are given admin permissions must be provided in the `SERVICE_PROJECT` setting. The `AVAILABLE_SERVICES` setting contains the set of services whose endpoints can be managed from the Dashboard. Both `type` and `description` are mandatory, while the `extra_roles` setting is optional and has to do with special roles being assigned to the given service account, either in a domain or in a project.

```
# ENDPOINTS MANAGEMENT DASHBOARD
SERVICE_PROJECT = 'service'
AVAILABLE_SERVICES = {
    'swift': {'type': 'Object storage',
              'description': 'Stores and retrieves arbitrary unstructured data objects,
↪via a RESTful, HTTP based API. \
                           It is highly fault tolerant with its data replication and
↪scale out architecture. Its \
                           implementation is not like a file server with mountable
↪directories.'},
    'nova': {'type': 'Compute',
             'description': 'Manages the lifecycle of compute instances in an OpenStack
↪environment. Responsibilities \
                           include spawning, scheduling and decommissioning of
↪machines on demand.'},
    'cinder': {'type': 'Block storage',
              'description': 'Provides persistent block storage to running instances.
↪Its pluggable driver architecture \
                           facilitates the creation and management of block storage
↪devices.',
              'extra_roles': [{'role': 'cinder-role', 'domain': 'cinder-domain'}]
    },
}
```

## 3. Django settings

The settings for all the Django configuration are located at `horizon/openstack_dashboard/settings.py`. Here we added some django apps, middleware, etc. You can check the file for reference but there is no extra configuration needed here.

## 4. Running a development server

To run a simple server to try out and check the IdM installation or for developping purposes you can use Django's development server that comes with the IdM installation, which will automatically run in port 8000:

```
$ sudo tools/with_venv.sh python manage.py runserver
```

You can also explicitly run::

```
$ sudo tools/with_venv.sh python manage.py runserver IP:PORT
```

For more documentation about this server, head on to [django docs](#)

**Warning:** As the Django documentation states: DO NOT USE THIS SERVER IN A PRODUCTION SETTING. It has not gone through security audits or performance tests. For a production setting, follow the [Production Set up Guide](#).

## 3.2.2 Installing Keystone

### 1. Installation

1. Install the Ubuntu dependencies

```
$ sudo apt-get install python python-dev python-virtualenv libxml2-dev_  
↪ libxslt1-dev libsasl2-dev libssl-dev libldap2-dev libffi-dev libsqlite3-  
↪ dev libmysqlclient-dev python-mysqldb
```

2. Get the code from our [GitHub repository](#)

```
$ git clone https://github.com/ging/keystone && cd keystone
```

3. Install the python dependencies

```
$ sudo python tools/install_venv.py
```

4. Create a configuration file

```
$ cp etc/keystone.conf.sample etc/keystone.conf
```

5. Create the tables and populate the database

```
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=oauth2  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=roles  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=user_  
↪ registration  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=two_  
↪ factor_auth  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --  
↪ extension=endpoint_filter  
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --populate
```

6. Finally, you can run keystone from the console

```
$ sudo tools/with_venv.sh bin/keystone-all -v
```

You may now log into the web (if you have Horizon installed) using the administrative account (by default, user is *idm* and the password is the one you entered during the populate step).

---

**Note:** If you want to run the Keystone backend in the background you can *install it as a service*.

---

## 2. Configuration

After creating the default configuration file, the following lines must be uncommented and set to your custom values.

```
admin_token=ADMIN
admin_port=35357
public_port=5000
```

## 3. Run Keystone

To run Keystone, we must either run it as a service or in a console with the following command:

```
$ sudo tools/with_venv.sh bin/keystone-all -v
```

## 4. Configuring Keystone as a service

If you want to add the Keystone to `init.d` to run it as a service there are several possibilities. You can try to reuse the scripts provided with Keystone or you can add a `.conf` file to `etc/init`. Here is a basic example:

Create the following file at: `/etc/init/keystone_idm.conf`

```
# keystone_idm - keystone_idm job file
description "Service conf file for the IdM backend based in Keystone"
author "Enrique Garcia Navalon <garcianavalon@gmail.com>"
start on (local-filesystems and net-device-up IFACE!=lo)
stop on runlevel [016]
# Automatically restart process if crashed
respawn
setuid root
script
cd $absolute_keystone_path
#activate the venv
. .venv/bin/activate
#run keystone
bin/keystone-all
end script
```

To run Keystone, you can now run it with the following command:

```
$ sudo service keystone_idm start
```

## 5. Running tests

In order to test, we use the Keystone built in system: **tox** and **testr**.

To execute all tests:

```
$ sudo tox
```

To Execute the extension tests (in this case for oauth2):

```
$ sudo tox -e py27 -- keystone.tests.test_v3_oauth2
```

---

**Note:** To debug during test, add the following parameter to the command: `-e debug`

---

## 3.3 System Administration

### 3.3.1 CLI tools

A set of commands is provided to help with some common tasks like updating endpoints and regions, a console to execute python against Keystone API, etc.

To install them

```
$ git clone https://github.com/ging/fiware-idm imd-admin && cd imd-admin
$ sudo pip install -r requirements.txt
$ sudo python setup.py install
```

Get to know how to use them and the available commands by using the following

```
$ idm-admin --help
```

### 3.3.2 White and black lists

As administrator of IdM KeyRock you can manage white and black lists in order to allow and deny access to users by their email domains.

There is a file for each of the list which you can find at `/horizon/openstack_dashboard/fiware_auth/blacklist.txt` or `whitelist.txt`.

- *Whitelist*: add a line for each of the domains that are allowed. No other domain will be allowed to register users.
- *Blacklist*: add a line for each of the domains that are not allowed. If a user has an email from this domain, they will not be able to register.

## 3.4 Sanity Check Procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

### 3.4.1 End to End testing

1. Verify that the host address of IdM can be reached. By default, web access will show a Login Page.
2. Acquire a valid username and password and access with those credentials. The resulting web page is the landing page of the IdM KeyRock Portal.
3. Verify that you can view the list of applications, organizations, etc.

### 3.4.2 List of Running Processes

In you have run the Horizon and Keystone run commands without errors, the portal is up and running.

### 3.4.3 Network interfaces Up & Open

- TCP port 80 should be accessible to the web browsers in order to load the IdM Portal (8000 for development).
- Ports 5000 and 35357 are Keystone's public and admin port respectively.

## 3.5 Databases

If you have correctly populated the database when installing the GE, the connection with it is up and running.

The databases and tables needed are:

```
+-----+
| Tables_in_keystone |
+-----+
| access_token_oauth2 |
| assignment           |
| authorization_code_oauth2 |
| consumer_credentials_oauth2 |
| consumer_oauth2      |
| credential            |
| domain               |
| endpoint              |
| endpoint_group        |
| group                 |
| id_mapping            |
| migrate_version       |
| permission_fiware     |
| policy                |
| project               |
| project_endpoint      |
| project_endpoint_group |
| region                |
| revocation_event      |
| role                  |
| role_fiware           |
| role_organization_fiware |
| role_permission_fiware |
| role_user_fiware      |
| service               |
| token                 |
```

(continues on next page)

(continued from previous page)

trust	
trust_role	
user	
user_group_membership	
user_registration_activation_profile	
user_registration_reset_profile	
+-----+	

## 3.6 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

### 3.6.1 Resource availability

- Verify that 2.5MB of disk space is left using the UNIX command ‘df’

### 3.6.2 Remote Service Access

Please make sure port 80 is accessible (port 8000 in development mode).

### 3.6.3 Resource consumption

Typical memory consumption is 100MB and it consumes almost the 1% of a CPU core of 2GHz, but it depends on user demand.

### 3.6.4 I/O flows

Clients access the KeyRock Interface through the client’s Web Browser. This is simple HTTP traffic. It makes requests to the local database.



- *MySQL*
  - *Install MySQL*
  - *Configure Keystone*
  - *Populate Database*
- *Web Server (Apache + mod\_wsgi)*
  - *Install apache and mod\_wsgi*
  - *Configure Apache*
  - *Collect Static Assets*
- *NO CAPTCHA reCAPTCHA*
- *Email Configuration*

This section covers how to set up the IdM for production, covering topics like email sending, No CAPTCHA reCAPTCHA support or how to serve static and media files. Some topics, for example HTTPS, are beyond the scope of this documentation and only some pointers to related documentation are provided as a starting point.

Make sure to also check the documentation for the respective parts of the IdM for more in-depth information of the components.

- Back-end [ging/keystone](#)
- Front-end [ging/horizon](#)

## 4.1 MySQL

If you have installed the IdM using the automated tools the back-end (Keystone) will be configured to use a SQLite database. This is NOT recommended for production, we strongly advise to switch to a production-ready SQL database.

This guide covers how to configure MySQL but any other database compatible with [SQLAlchemy](#) would probably work too.

### 4.1.1 Install MySQL

```
sudo apt-get install mysql-server
```

### 4.1.2 Configure Keystone

Edit **keystone/etc/keystone/keystone.conf** and change the [database] section.

```
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://keystone:KEYSTONE_DBPASS@MYSQL_ADDRESS/keystone
```

Use the password that you set previously to log in as root. Create a keystone database user:

```
# mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY
↳ 'KEYSTONE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'KEYSTONE_
↳ DBPASS';
```

### 4.1.3 Populate Database

You need to create the database tables and populate them.

```
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=oauth2
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=roles
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=user_
↳ registration
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --extension=two_
↳ factor_auth
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --
↳ extension=endpoint_filter
$ sudo tools/with_venv.sh bin/keystone-manage -v db_sync --populate
```

You can find additional help for setting up Keystone + MySQL [here](#).

## 4.2 Web Server (Apache + mod\_wsgi)

The web server used by the tools is a development server that should NOT be used for a production setting. There are several servers and configurations to serve a Django (Python) web application but only Apache + mod\_wsgi will be covered here. Take a look at the [official Django documentation](#) for other options available and further information on this topic.

### 4.2.1 Install apache and mod\_wsgi

```
sudo apt-get install apache2 libapache2-mod-wsgi
```

### 4.2.2 Configure Apache

The details on how to correctly configure Apache or set up HTTPS are beyond the scope this document, check the [Django documentation](#) and [Apache HTTPS documentation](#) for a starting point. Make sure that the following elements are present: (take special care with the venv)

```
WSGIPassAuthorization On
WSGIScriptAlias / [PATH_TO_HORIZON]/horizon/openstack_dashboard/wsgi/django.wsgi
WSGIProxyPath [PATH_TO_HORIZON]/horizon/openstack_dashboard:[PATH_TO_HORIZON]/
↳horizon/.venv/lib/python2.7/site-packages
```

If you want to serve your static and media files from Apache itself, also make sure to create the Alias

```
Alias /media/ /root/horizon/media/
Alias /static/ /root/horizon/static/
Alias /assets/ /root/horizon/static/fiware/
<Directory [PATH_TO_HORIZON]/horizon/static>
    Require all granted
</Directory>
<Directory [PATH_TO_HORIZON]/horizon/media>
    Require all granted
</Directory>
```

As reference, here you can see a full Apache configuration file using HTTPS

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName foo
    ServerAdmin bar

    WSGIScriptAlias / /home/someone/horizon/openstack_dashboard/wsgi/django.wsgi

    <Directory /home/someone/horizon/openstack_dashboard/wsgi>
        Order allow,deny
        Allow from all
    </Directory>

    Alias /media/ /home/someone/horizon/media/
    Alias /static/dashboard/fonts /home/someone/horizon/openstack_dashboard/static/
↳dashboard/fonts
    Alias /static/dashboard/img /home/someone/horizon/openstack_dashboard/static/
↳dashboard/img
    Alias /static/dashboard/css /home/someone/horizon/static/dashboard/css
    Alias /static/dashboard/js /home/someone/horizon/static/dashboard/js

    <Directory /path/to/foo/static>
        Require all granted
    </Directory>

    <Directory /path/to/foo/media>
        Require all granted
```

(continues on next page)

(continued from previous page)

```

</Directory>

ErrorLog ${APACHE_LOG_DIR}/error.log

# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
LogLevel debug

CustomLog ${APACHE_LOG_DIR}/ssl_access.log combined

#    SSL Engine Switch:
#    Enable/Disable SSL for this virtual host.
SSLEngine on

SSLCertificateFile      /etc/ssl/private/someplace.org/somecert.crt
SSLCertificateKeyFile   /etc/ssl/private/someplace.org/*.somepem.pem
SSLCertificateChainFile /etc/ssl/private/someplace.org/chain.crt

<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
    SSLOptions +StdEnvVars
</Directory>

BrowserMatch "MSIE [2-6]" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
# MSIE 7 and newer should be able to use keepalive
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
</IfModule>

#rdeirection to the secure version
<VirtualHost 0.0.0.0:80>
    ServerName foo2
    Redirect permanent / foo
</VirtualHost>

```

### 4.2.3 Collect Static Assets

Now, go to the folder you have installed Horizon and run

```

sudo tools/with_venv.sh python manage.py collectstatic
sudo tools/with_venv.sh python manage.py compress --force

```

Edit the `local_settings.py` file and set

```

DEBUG = False
ALLOWED_HOSTS = [
    'your.domain.com',
    'another.domain.es'
]
SECRET_KEY = 'arandomstringhere' # DON'T LEAVE THIS SAMPLE STRING

```

**Warning:** Please set your SECRET\_KEY. A known SECRET\_KEY is a huge security vulnerability.

More information [here](#)

## 4.3 NO CAPTCHA reCAPTCHA

**Warning:** Don't deploy KeyRock in a public domain with CAPTCHA disabled.

Get your keys [here](#). More documentation in the [captcha package repository](#).

```
USE_CAPTCHA = False
NORECAPTCHA_SITE_KEY = '6LeIxAcTAAAAAJcZVRqyHh71UMIEGNQ_MXjiZKhI'
NORECAPTCHA_SECRET_KEY = '6LeIxAcTAAAAAGG-vFl1TnRWxMZNfuojJ4WifJWe'
```

## 4.4 Email Configuration

The IdM can't send emails by itself, you must set up a SMTP server to send it. This section covers how to set up a mail server using [POSTFIX](#) and connect the front-end to it. Further information can be found in the [Django documentation](#).

Install and configure [POSTFIX](#), [Ubuntu guide](#).

```
sudo apt-get install postfix
```

Go to the folder where you have installed the front-end and edit local\_settings.py

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
```

This will get the settings from the default SMTP server in your host (it should be POSTFIX after installing it) If you are not running POSTFIX in the same host or want to use a different configuration, make use of the following settings

```
# Configure these for your outgoing email host
EMAIL_HOST = 'smtp.my-company.com'
EMAIL_PORT = 25
EMAIL_HOST_USER = 'djangomail'
EMAIL_HOST_PASSWORD = 'top-secret!'
EMAIL_URL = 'your-webstie-domain.com'
DEFAULT_FROM_EMAIL = 'your-no-reply-address'
EMAIL_SUBJECT_PREFIX = '[Prefix for emails subject]'
```



---

### Private Clouds Federation

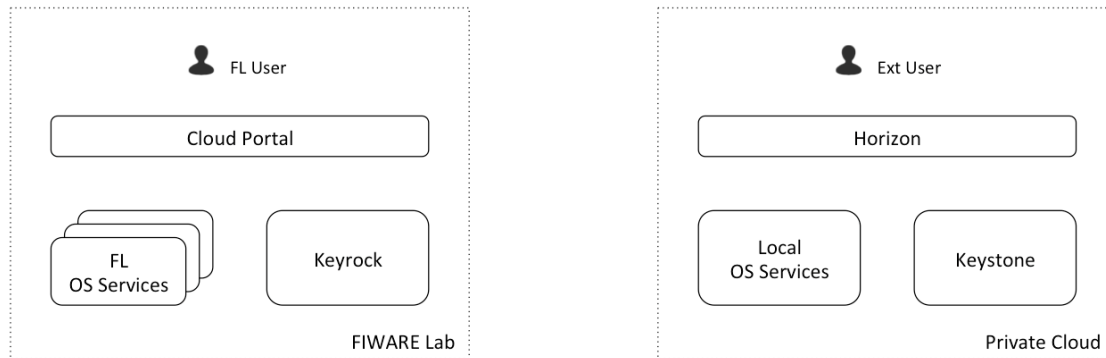
---

- *Main Concepts*
  - *The scenario*
  - *Espected Behaviour*
  - *FL User using FIWARE Lab resources*
  - *Ext User using Local resources*
  - *FL User using Private Cloud resources*
- *Installation and Configuration*

This section provides a guide of how to Federate a private Openstack Cloud with a central Keyrock-based FIWARE Environment. Using this kind of configuration, an external Openstack Cloud can offer part of its resources to the FIWARE Lab users.

## 5.1 Main Concepts

### 5.1.1 The scenario



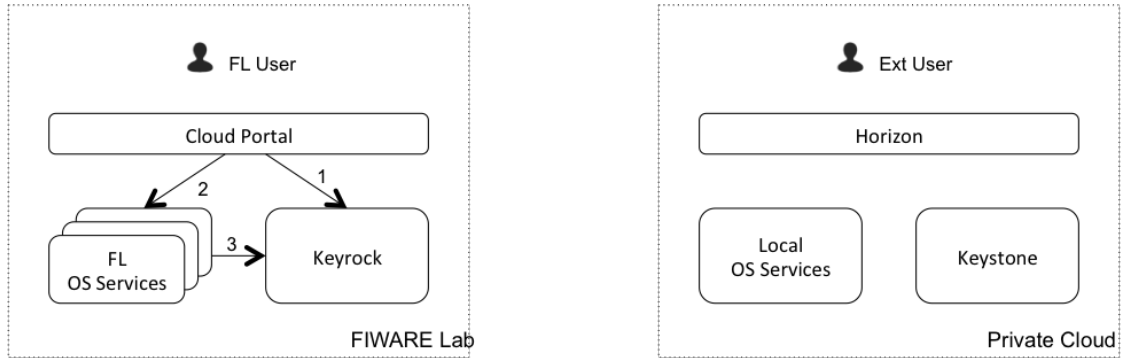
- FL user represents a user with a registered account in FIWARE Lab
- In FIWARE Lab environment, FL OS Services represent the services of all the Federated nodes
- Private Cloud wants to offer some of its resources (part of Local OS Services) to be available in FIWARE Lab as a new node.
- Private Cloud has their own users registered in its local Keystone (Ext User is one of them) and using Cloud resources deployed in Local OS Services

### 5.1.2 Expected Behaviour

- Ext User can continue using his deployed resources in Local OS Services using Horizon
- FL User (if he has the correct rights) can deploy resources in Private Cloud Local OS Services using Cloud Portal
- In Cloud Portal, Private Cloud node appears as a new node. It is accessible for FIWARE Lab users with quotas in that node (community users assigned to that node)
- Private Cloud infrastructure owners can assign quotas of Local OS Services to FIWARE Lab users
- FL User can continue using FL OS Services as before.
- If a Ext User wants to use FIWARE Lab nodes resources, he has to create an account in FIWARE Lab.

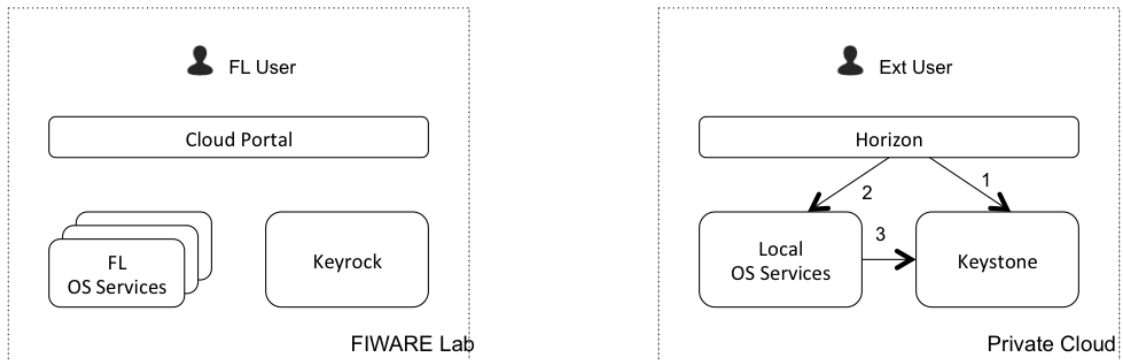


### 5.1.3 FL User using FIWARE Lab resources



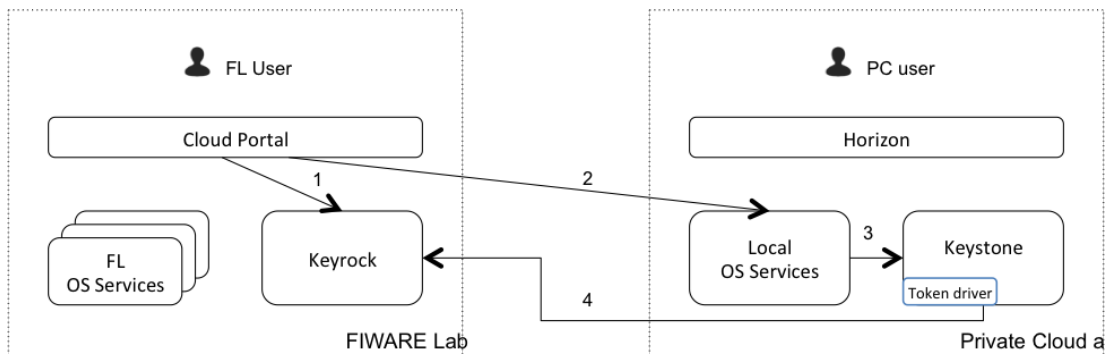
1. Cloud Portal authenticates the user in Keyrock
2. Cloud Portal sends a request to an OS Service
3. OS Service validates the token with Keyrock

### 5.1.4 Ext User using Local resources



1. Horizon authenticates the user in Keystone
2. Horizon sends a request to an OS Service
3. OS Service validates the token with Keystone

### 5.1.5 FL User using Private Cloud resources



1. Cloud Portal authenticates the user in Keyrock
  2. Cloud Portal sends a request to a Private Cloud OS Service
  3. Private Cloud OS Service tries to validate the token in Keystone
  4. As the validation doesn't success (the token is not stored in Keystone), Keystone validates it with Keyrock acting as a gateway and sending the response to Private Cloud OS Service
- \*. If the validation success, Keystone stores the token locally (in cache), so the next times the step 4 is not required.

## 5.2 Installation and Configuration

To have a compatible Keystone in your Private node you have to install a modified version of this component. This version is available [here](#)

This modified Keystone includes an implementation of the described Token Driver. You can install it as a regular Keystone. To configure the Token Driver you have only to add the information about the central FIWARE Lab Keyrock instance in the Keystone configuration file:

```
[simplefederation]
idp=http://user1:password1@idp1.provider1.test:35357
idp=http://user2:password2@idp2.provider2.test:35357
```

The configured users needs admin permissions in the central Keystone to be able to validate tokens there.

A detailed installation and configuration guide can be found [here](#)

---

### Using the FIWARE LAB instance (OAuth2)

---

- *Register your user account*
- *Register your application*
- *OAuth2 Authentication*
  - *Authorization Code Grant*
    - \* *Authorization Request*
    - \* *Authorization Response*
    - \* *Access Token Request*
    - \* *Access Token Response*
  - *Implicit Grant*
    - \* *Authorization Request*
    - \* *Access Token Response*
  - *Resource Owner Password Credentials Grant*
    - \* *Access Token Request*
    - \* *Access Token Response*
  - *Client Credentials Grant*
    - \* *Access Token Request*
    - \* *Access Token Response*
- *Get user information and roles*

There is already a deployed instance of the FIWARE IdM available at <https://account.lab.fiware.org/>

## 6.1 Register your user account

In order to start using the FIWARE IdM, you must first register your own account.

## 6.2 Register your application

The next step is [registering your own application](#). The `Callback URL` attribute is a mandatory parameter used in OAuth2 authentication. The IdM provides you with a `Client ID` and a `Client Secret` which are used in OAuth2.

## 6.3 OAuth2 Authentication

The FIWARE IdM complies with the OAuth2 standard described in [RFC 6749](#) and supports all four grant types defined there.

The `Authorization Basic` header is built with the `Client ID` and `Client Secret` credentials provided by the FIWARE IdM following the [standard](#). So the string will be

```
base64(client_id:client_secret)
```

The `redirect_uri` parameter must match the `Callback URL` attribute provided in the application registration.

### 6.3.1 Authorization Code Grant

The authorization code is obtained by using an authorization server (the IdM) as an intermediary between the client (the registered application) and resource owner (the user). Instead of requesting authorization directly from the resource owner, the client directs the resource owner to an authorization server (via its user-agent as defined in [RFC2616](#)), which in turn directs the resource owner back to the client with the authorization code.

#### Authorization Request

```
GET /oauth2/authorize?response_type=code&client_id=1&state=xyz
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcallback_url HTTP/1.1
Host: account.lab.fiware.org
```

The `response_type` attribute is mandatory and must be set to `code`. The `client_id` attribute is the one provided by the FIWARE IdM upon application registration. The `redirect_uri` attribute must match the `Callback URL` attribute provided to the IdM within the application registration. `state` is optional and for internal use of you application, if needed.

#### Authorization Response

```
HTTP/1.1 302 Found
Location: https://client.example.com/callback_url?code=Splxl0BeZQQYbYS6WxSbIA&
↪state=xyz
```

## Access Token Request

```
POST /oauth2/token HTTP/1.1
Host: account.lab.fiware.org
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=Sp1xl0BeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcallback_url
```

## Access Token Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token": "2YotnFZFEjrlzCsicMWpAA",
  "token_type": "bearer",
  "expires_in": 3600,
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA",
}
```

## 6.3.2 Implicit Grant

The implicit grant is a simplified authorization code flow optimized for clients implemented in a browser using a scripting language such as JavaScript. In the implicit flow, instead of issuing the client an authorization code, the client is issued an access token directly (as the result of the resource owner authorization). The grant type is implicit, as no intermediate credentials (such as an authorization code) are issued (and later used to obtain an access token).

## Authorization Request

```
GET /oauth2/authorize?response_type=token&client_id=1&state=xyz
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcallback_url HTTP/1.1
Host: account.lab.fiware.org
```

The `response_type` attribute is mandatory and must be set to `token`.

The `client_id` attribute is the one provided by the FIWARE IdM upon application registration.

The `redirect_uri` attribute must match the `Callback URL` attribute provided to the IdM within the application registration.

`state` is optional and for internal use of you application, if needed.

## Access Token Response

See *Authorization Code Grant*

### 6.3.3 Resource Owner Password Credentials Grant

The resource owner password credentials (i.e., username and password) can be used directly as an authorization grant to obtain an access token.

#### Access Token Request

```
POST /oauth2/token HTTP/1.1
Host: account.lab.fiware.org
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=password&username=demo&password=123
```

#### Access Token Response

See *Authorization Code Grant*

### 6.3.4 Client Credentials Grant

The client can request an access token using only its client credentials.

#### Access Token Request

```
POST /oauth2/token HTTP/1.1
Host: account.lab.fiware.org
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

#### Access Token Response

See *Authorization Code Grant*

## 6.4 Get user information and roles

**Warning:** Be aware that if you used the Client Credentials Grant to obtain the token there is no such thing as an 'authorizing user' because of the nature of this grant. You can still use this endpoint to validate the token, but the JSON (if the token is valid) will be empty.

Request:

```
GET /user?access_token=2YotnFZFEjrlzCsicMWpAA
```

Example response:

```
{
  id: 1,
  displayName: "Demo user",
  email: "demo@fiware.org",
  roles: [
    {
      id: 15,
      name: "Manager"
    },
    {
      id: 7
      name: "Ticket manager"
    }
  ],
  organizations: [
    {
      id: 12,
      name: "Universidad Politecnica de Madrid",
      roles: [
        {
          id: 14,
          name: "Admin"
        }
      ]
    }
  ]
}
```





---

## Two Factor Authentication

---

- *What is two factor authentication*
- *User guide*
  - *Requirements*
  - *Enabling two factor*
  - *Logging in*
  - *Disabling two factor*
  - *What happens if I lose my phone or uninstall the app*

### 7.1 What is two factor authentication

Two Factor Authentication, also known as two step verification, is an extra layer of security for authenticating a user. In any security system, there are three authentication factors which can be used: something the user knows, something that he owns and something that he is. Two factor authentication uses the first two: a combination of username and password (knowledge) and a physical token (possession.)

In KeyRock's implementation of two factor authentication, the physical token is the users' smart phone thanks to an app. This app will, after being correctly set up, generate unique time-based passwords (also known as verification codes) that will authenticate the user in combination with the right username and password. The app needs no internet connection to generate the verification codes after being set up.

## 7.2 User guide

### 7.2.1 Requirements

You will need to install a third party app that implements the Open MFA standards defined in [RFC 4226](#) (HOTP: An HMAC-Based One-Time Password Algorithm) and in [RFC 6238](#) (TOTP: Time-Based One-Time Password Algorithm).

---

**Important:** We recommend [Google Authenticator](#).

---

### 7.2.2 Enabling two factor

To enable it you must log into KeyRock and head to your settings menu. A two factor section is there with all the instructions to follow. In summary, you will need to:

- Provide a question and its answer (keep it secret!)
- Generate a new secret key
- Configure your app with this secret key using the QR Code or manually

### 7.2.3 Logging in

Once two factor authentication is enabled, your logging process will have a new step. After providing your username and password you will be asked for the verification code generated by your app.

---

**Note:** For convenience, you can remember your computer and no verification codes will be asked when you log in from it. Use this option only in trusted computers.

---

### 7.2.4 Disabling two factor

Simply log into your account, head to settings and disable it in its respective section. Once disabled, you can log in normally in all computers.

### 7.2.5 What happens if I lose my phone or uninstall the app

As a security measure in case of lost or theft of the smart phone or the app, we also ask for a security question and a secret answer to be provided on the activation process. This question and answer can be used to disable two factor authentication without need to authenticate.

- *Introduction*
- *Using the web portal of KeyRock*
  - *Logging in*
  - *Registering an application*
  - *Managing roles*
  - *Managing organizations*
- *Programmer Guide*
  - *Users*
    - \* *Get a single user*
    - \* *Get authenticated user*
  - *Applications*
    - \* *Get applications from actor (user or organization)*
  - *SCIM 2.0*
    - \* *Get service provider configuration*
- *Further information*

## 8.1 Introduction

This document describes the user and programming guide for Keyrock Identity Management component. Here you will find the necessary steps for use the Keyrock portal for create an account and manage it. You will also learn about role and applications management.

This User and Programmers Guide relates to the [Identity Management GE](#).

## 8.2 Using the web portal of KeyRock

Although every user of KeyRock will access the web portal with individual credentials, the following description uses a test account. In every KeyRock instance the web portal can be accessed at **\*\*FIWARE Account Portal\*\***.

### 8.2.1 Logging in

Go to “Sign in” if you have previously created an account, otherwise “Sign up” to create a new account:

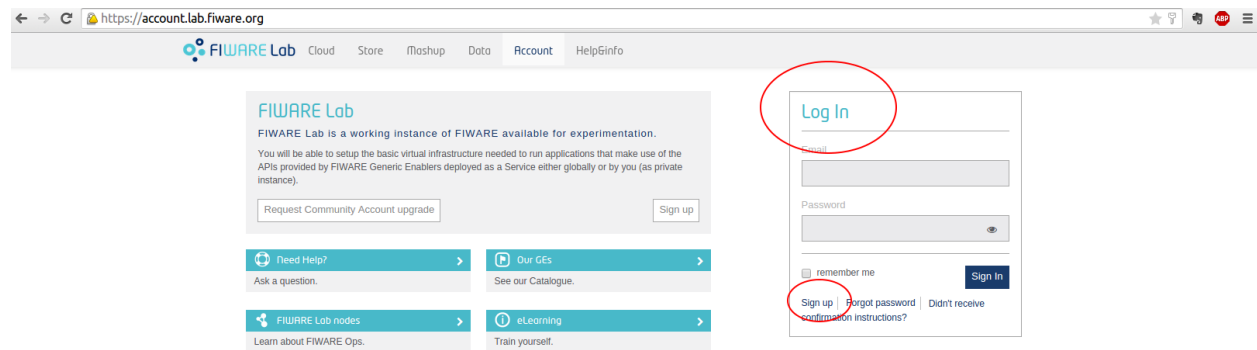


Figure 2 shows the homepage after you log in successfully.

There are two main sections, Applications and Organizations. In the Applications section you can register new application by clicking on “Register”.

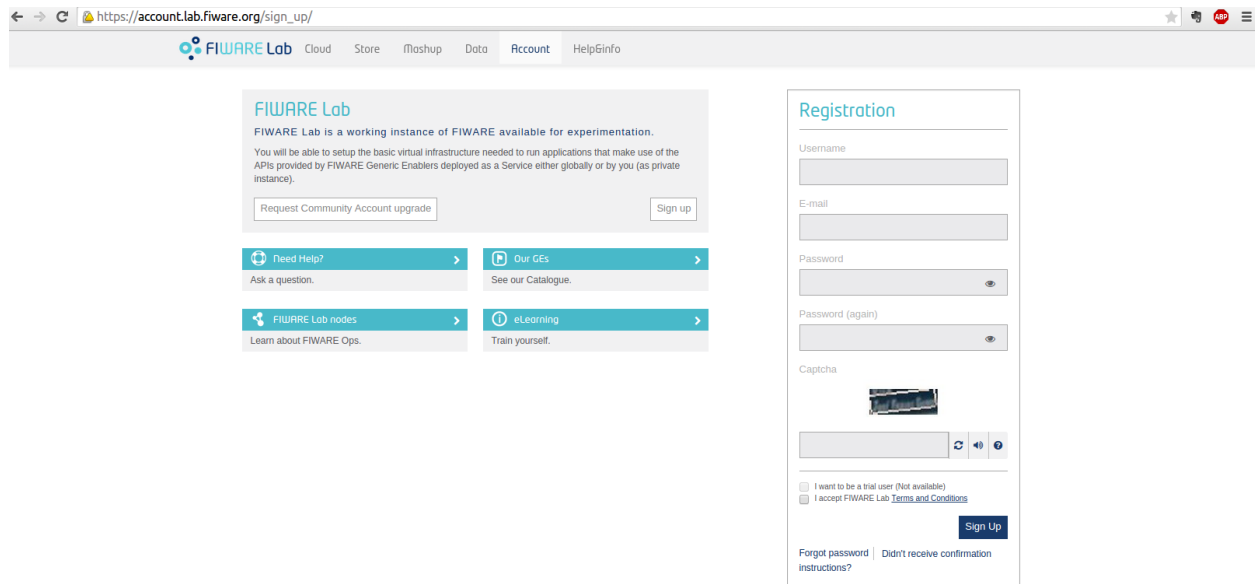
### 8.2.2 Registering an application

In the next step you have to give the application a name, description, URL and callback URL - required by the OAuth 2.0 Protocol.

Click on “Next” (Figure 3).

In the second step the application’s logo will be loaded by selecting a valid file type. You have the option to re-frame the chosen image.

Click on “Crop Image” when you complete this process and then click “Next” as shown on Figure 4.



← → ↻ https://account.lab.fiware.org/sign\_up/ ★ 🔍

FIWARE Lab Cloud Store Mashup Data Account HelpInfo

**FIWARE Lab**  
FIWARE Lab is a working instance of FIWARE available for experimentation.  
You will be able to setup the basic virtual infrastructure needed to run applications that make use of the APIs provided by FIWARE Generic Enablers deployed as a Service either globally or by you (as private instance).

[Request Community Account upgrade](#) [Sign up](#)

[Need Help?](#) [Our CEs](#)  
Ask a question. See our Catalogue.

[FIWARE Lab nodes](#) [eLearning](#)  
Learn about FIWARE Ops. Train yourself.

**Registration**

Username

E-mail

Password

Password (again)

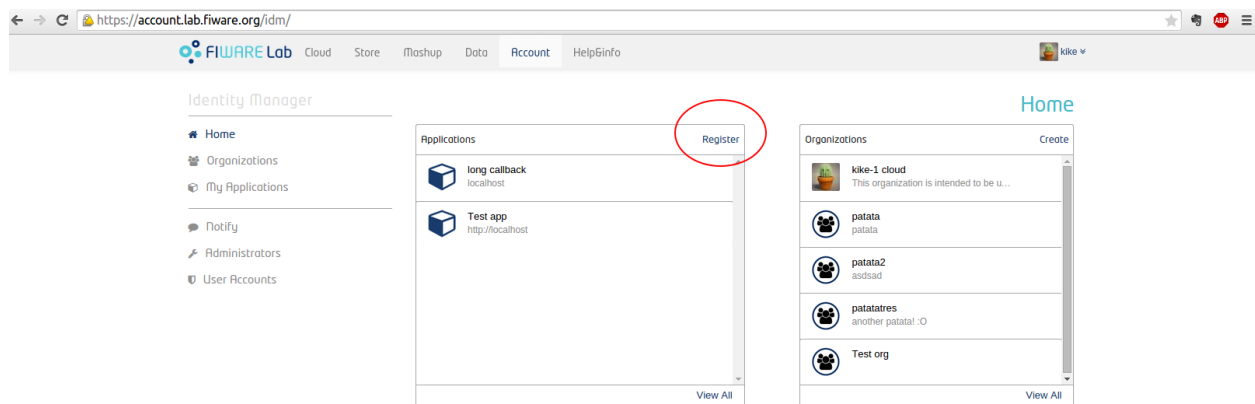
Captcha

☐ I want to be a trial user (Not available)  
☐ I accept FIWARE Lab [Terms and Conditions](#)

[Sign Up](#)

[Forgot password](#) | [Didn't receive confirmation instructions?](#)

Fig. 1: Figure 1: KeyRock Login Page



← → ↻ https://account.lab.fiware.org/idm/ ★ 🔍

FIWARE Lab Cloud Store Mashup Data Account HelpInfo

**Identity Manager**

- Home
- Organizations
- My Applications
- Notify
- Administrators
- User Accounts

**Applications**

[Register](#)

[long callback](#)  
localhost

[Test app](#)  
http://localhost

[View All](#)

**Organizations**

[Create](#)

[kike-1 cloud](#)  
This organization is intended to be u...

[patata](#)  
patata

[patata2](#)  
asdsad

[patatares](#)  
another patata! :D

[Test org](#)

[View All](#)

Fig. 2: Figure 2: KeyRock Home Page

The screenshot shows the 'Identity Manager' interface in a web browser. The URL is <https://account.lab.fiware.org/idm/myApplications/create/>. The page has a navigation bar with 'FIWARE Lab' and links for 'Cloud', 'Store', 'Mashup', 'Data', 'Account', and 'Help/Info'. A user profile 'kike' is visible in the top right. On the left, a sidebar lists 'Home', 'Organizations', 'My Applications' (selected), 'Notify', 'Administrators', and 'User Accounts'. The main content area shows a three-step progress indicator with step 1 highlighted. Below the indicator are input fields for 'Name', 'Description', 'URL', and 'Callback URL'. A 'Next' button is at the bottom right.

Fig. 3: Figure 3: KeyRock Register Application

The screenshot shows the 'Identity Manager' interface in a web browser. The URL is <https://account.lab.fiware.org/idm/myApplications/fdce708f1c504116aabb9b5d945ce65f/step/avatar/>. The page layout is identical to the previous screenshot, but the progress indicator shows step 2 highlighted. The main content area now features a blue cube icon, an 'Image' label, and a 'Choose File' button with the text 'No file chosen' next to it. The 'Next' button remains at the bottom right.

In the third step we set up the roles and permissions of the application. You will find the two possible roles: Provider and Purchaser.

You can edit the permission for each of the roles or create new roles. Click on “New role” and write the name of role, after that click “Save”.

You can configure the permissions for the new role by activating the corresponding check box.

You are also permitted to add up new permissions by clicking on “New Permission”. Here you need to enter the name of the permission, description, HTTP verb (GET, PUT, POST, DELETE) and the Path to that permission, Figure 5.

Click “Create Permission” and “Finish” to finalize with creating the application.

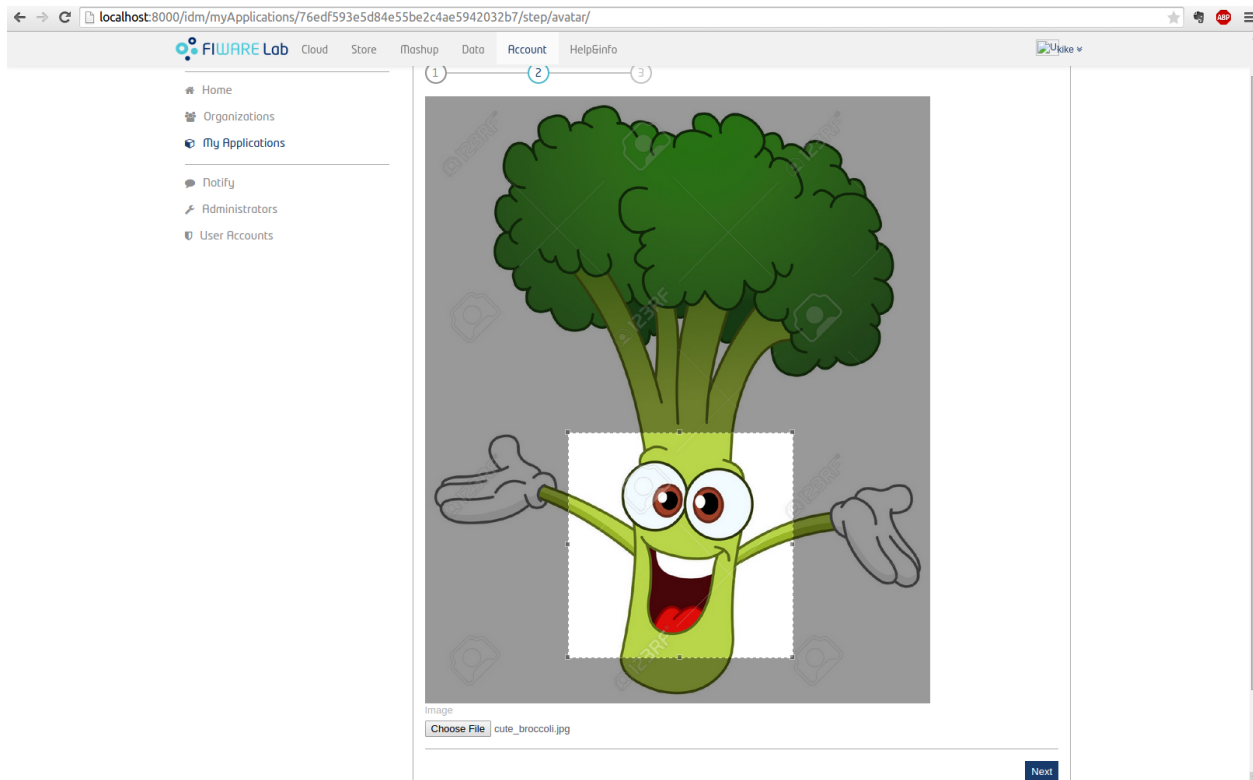
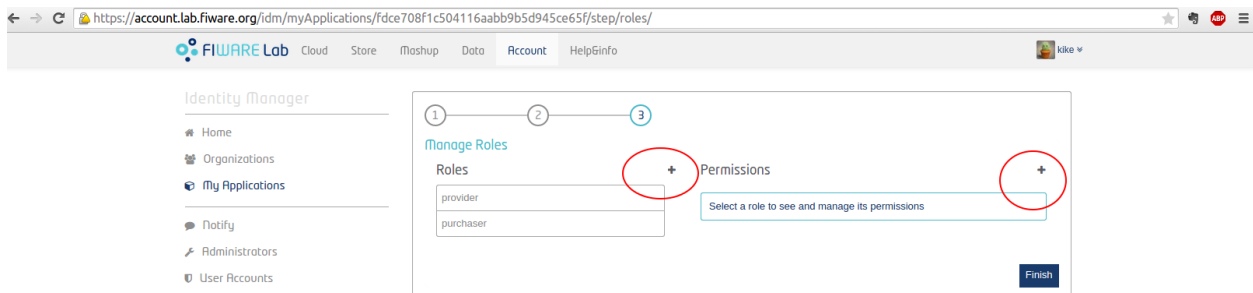


Fig. 4: Figure 4: KeyRock Edit Application Logo



### 8.2.3 Managing roles

Look at the vertical menu on the left (Figure 6). You went from Home to Applications. Here you can see the application you've just created.

At the bottom you can manage the roles of the users. You can add new users on the “Add” button.

It shows a modal where you can manage Users and Groups. You can see the users and their initially assigned roles.

Choose users and groups to add to the application, then choose their initial role. Click “Add”.

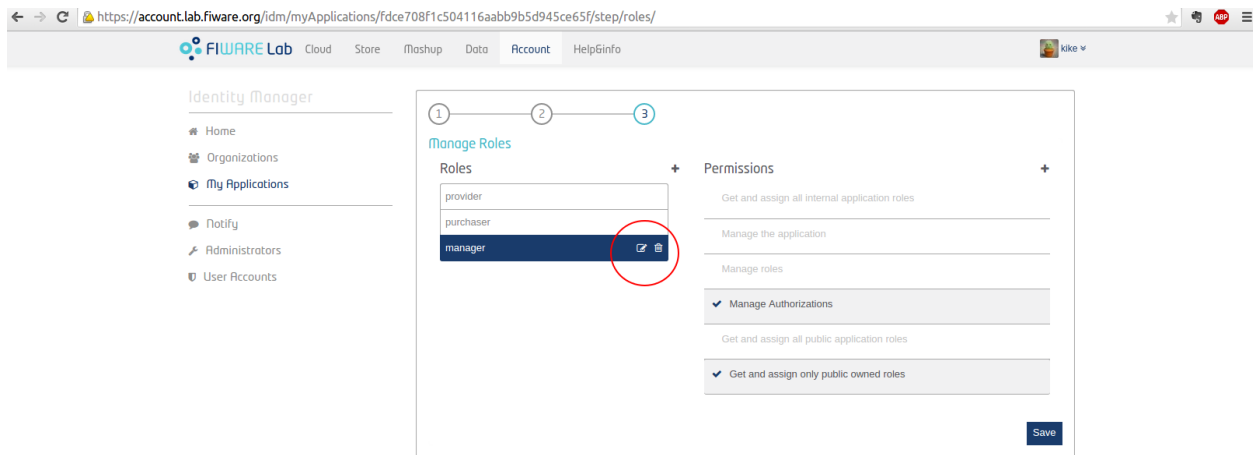
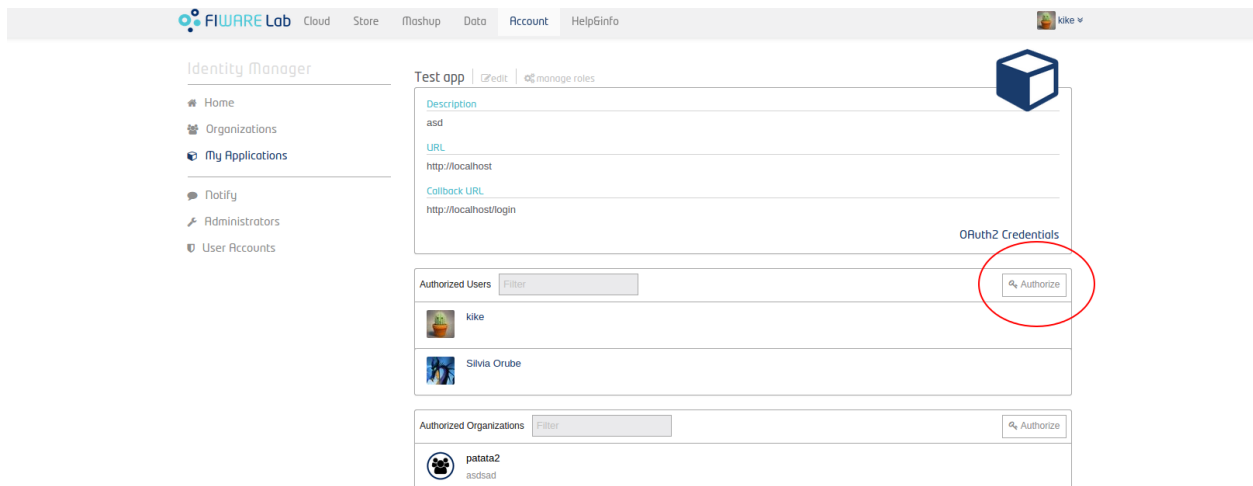


Fig. 5: Figure 5: KeyRock New Roles and Permissions

Note that you can assign roles after the users have been added, by clicking on the roles drop down menu - below the user's icon, as shown on Figure 6.



## 8.2.4 Managing organizations

Next head on to the vertical menu and click “Organizations”. Click “Create Organization” to register a new organization.

Add the name, choose the owner and write the description of the organization. Click “Create Organization”.

You are now redirected to the Home menu on behalf of the newly created organization. Any new application created now, will belong to the organization.

To return to the home of the user go up in the header and click on the name of the organization. Select “Switch



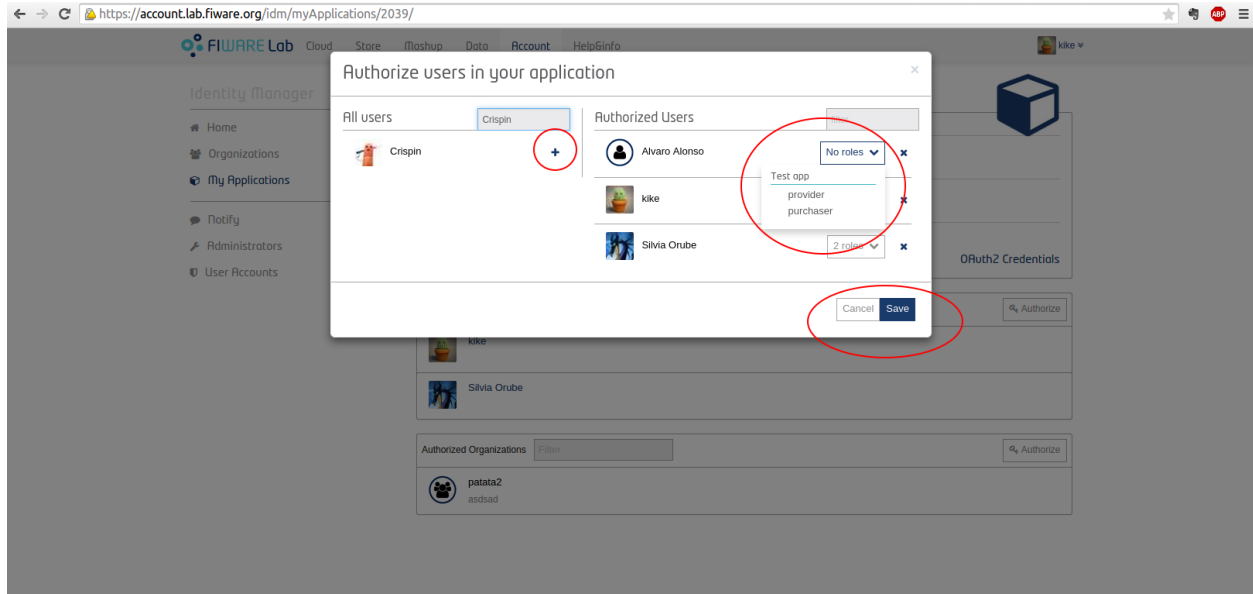


Fig. 6: Figure 6: KeyRock Add Members to Application

session”, Figure 7.

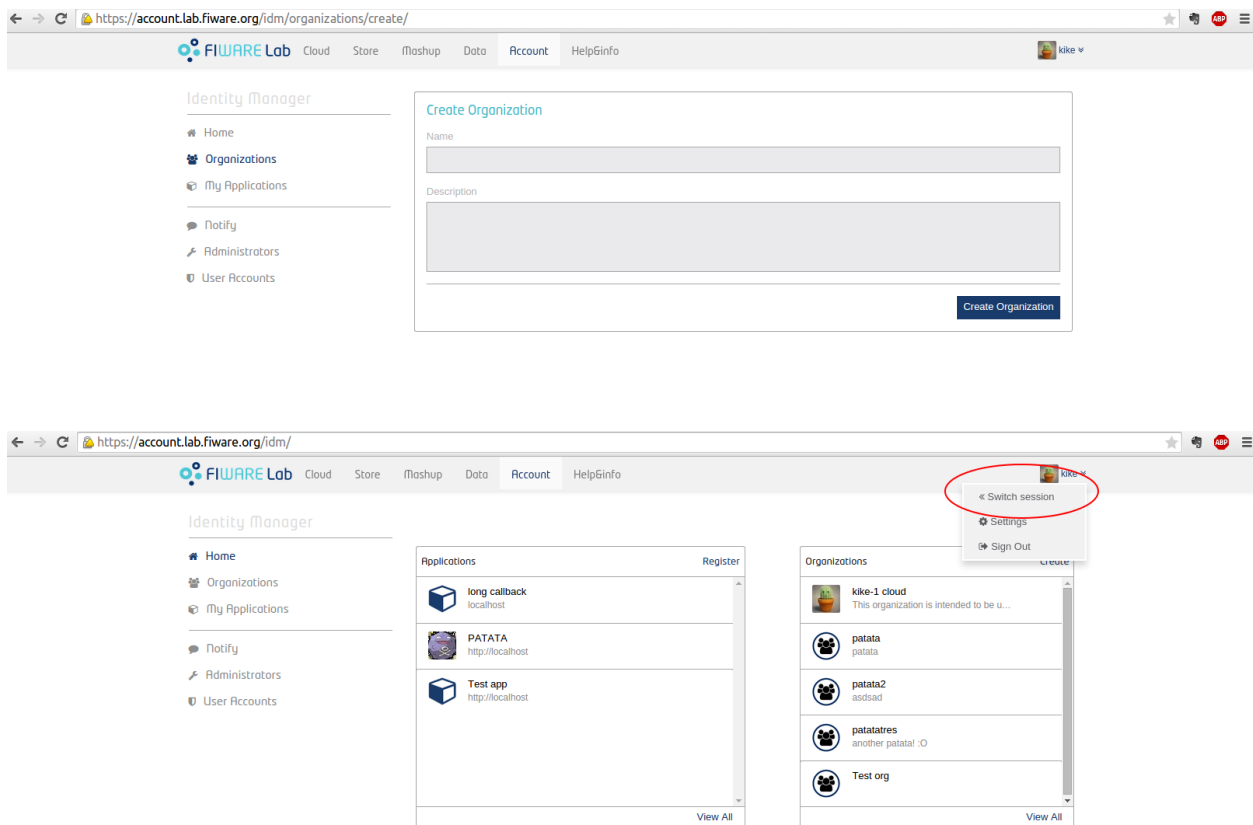


Fig. 7: Figure 7: KeyRock Create Organization

## 8.3 Programmer Guide

Documentation on KeyRock APIs can be found at *API Overview section*

### 8.3.1 Users

#### Get a single user

Request:

```
GET /users/:id
```

Example response:

```
{
  "id": 1,
  "actorId": 1,
  "nickName": "demo",
  "displayName": "Demo user",
  "email": "demo@fiware.eu",
  "roles": [
    {
      "id": 1,
      "name": "Manager"
    },
    {
      "id": 7,
      "name": "Ticket manager"
    }
  ],
  "organizations": [
    {
      "id": 1,
      "actorId": 2,
      "displayName": "Universidad Politecnica de Madrid",
      "roles": [
        {
          "id": 14,
          "name": "Admin"
        }
      ]
    }
  ]
}
```

#### Get authenticated user

Request:

```
GET /user?access_token=12342134234023437
```

### 8.3.2 Applications

### Get applications from actor (user or organization)

Request:

```
GET /applications.json?actor_id=1&access_token=2YotnFZFEjrlzCsicMWpAA
```

Example response:

```
{
  "id": 1,
  "name": "Dummy",
  "description": "fiware demo application",
  "url": "http://dummy.fiware.eu/"
}
```

### 8.3.3 SCIM 2.0

Documentation on KeyRock APIs can be found at [API Overview section](#). We provide bellow an example of API call, to retrieve the service provider documentation.

#### Get service provider configuration

Request:

```
GET /v2/ServiceProviderConfigs
```

Example response:

```
{
  "schemas": [
    "urn:scim:schemas:core:2.0:ServiceProviderConfig"
  ],
  "documentationUrl": "https://tools.ietf.org/html/draft-ietf-scim-core-schema-02",
  "totalUsers": "200",
  "totalOrganizations": "50",
  "totalResources": "250"
}
```

## 8.4 Further information

For further information on KeyRock, please refer to the step-by-step video at [Help & Info Portal](#) choosing “Account”, as **Figure 8\_** shows.

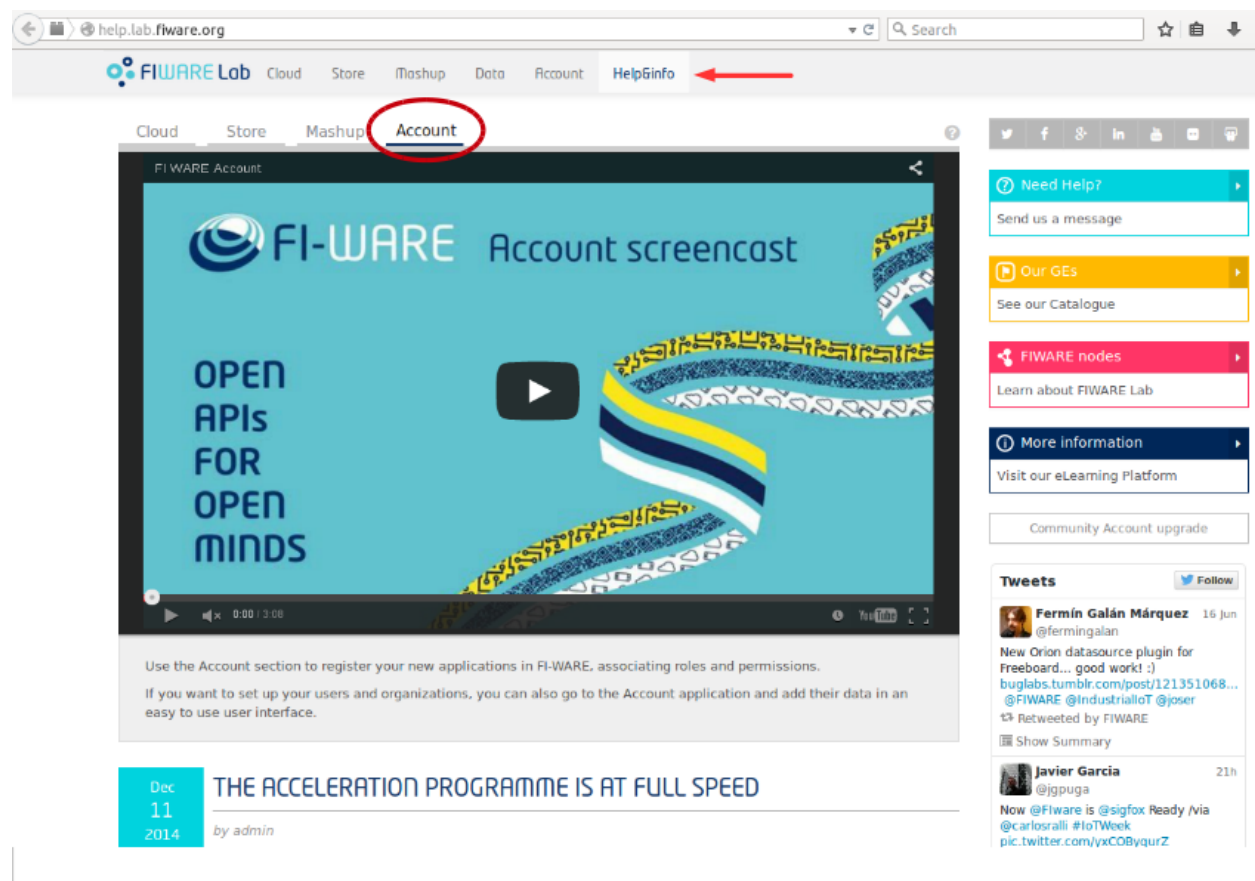


Fig. 8: Figure 8: KeyRock Screencast

---

### Endpoints Management Dashboard (admin-only)

---

- *What is the Endpoints Management Dashboard*
- *User guide*
  - *How to enable and disable services*
  - *How to update a service endpoint*
  - *Managing services accounts*

#### 9.1 What is the Endpoints Management Dashboard

The Endpoints Management Dashboard is a tool that helps node administrators perform CRUD operations regarding the endpoints of OpenStack services. This tool is intended for node administrators at FIWARE Lab, but it could also be used in any other cloud infrastructure. It offers the following functionalities:

- Enabling an OpenStack service for your node, by creating its user account and group and helping you create its endpoints
- Disabling an Openstack service in your node, by deleting both its user account, its endpoint group and its endpoints
- Updating the endpoints of an enabled service in your node
- Getting new credentials for the user account of a certain service in your node

---

**Important:** For security purposes, only admin users can access this dashboard.

---

## 9.2 User guide

In this section, the different functionalities of the Endpoints Management Dashboard are covered. Remember that you can only access the Dashboard if you are a node administrator.

### 9.2.1 How to enable and disable services

The following screenshot depicts the Endpoints Management Dashboard. On the left you will find the list of services which are available in the Keystone Service Catalog.

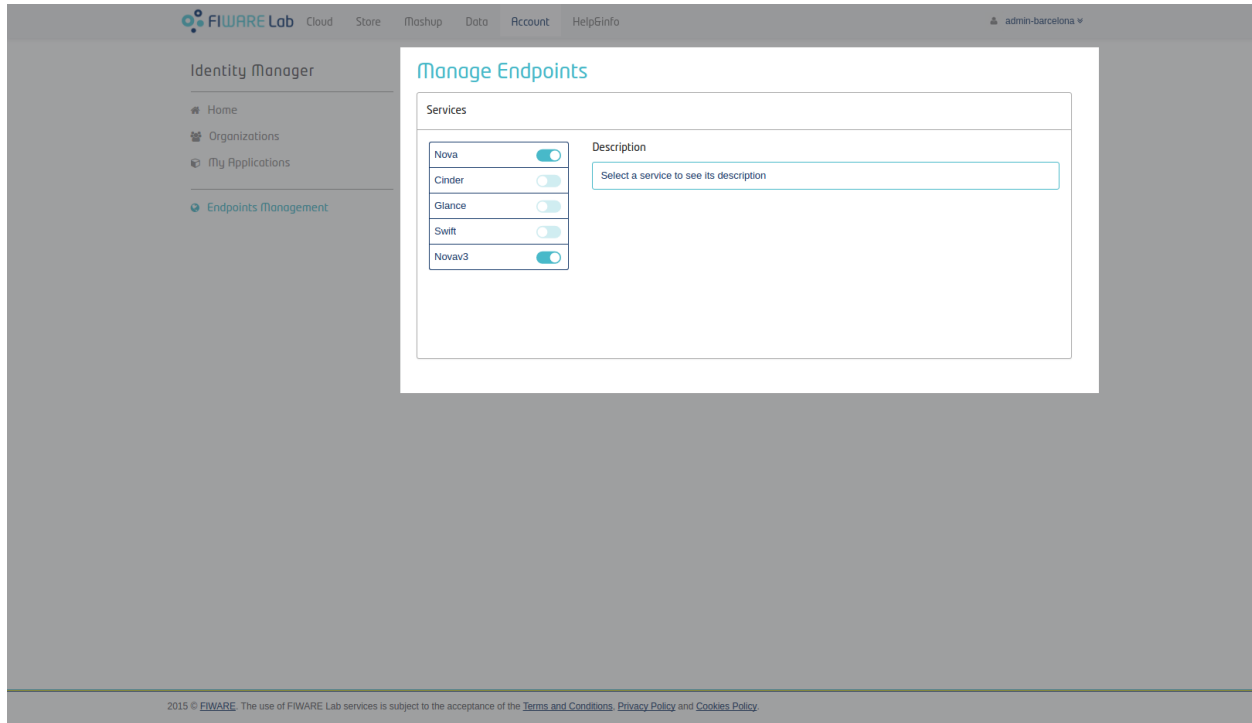


Fig. 1: Endpoints Management Dashboard entry point. The Dashboard has been highlighted.

The switch next to each service name will tell you whether or not the service is enabled for your node. Click on a service name to take a look at its description; endpoints and user account information will be shown too if the service is enabled for your node. If you have permissions to manage more than one region, information of all the regions will be shown.

- To enable a service, click on the switch next to its service name, and provide the endpoints for it. Both of the three interfaces (public, internal & admin are required). When you are finished, click on save to enable the service.
- To disable a service, click on the switch next to its service name. You will be prompted with a confirmation dialog to make sure you want to proceed.

### 9.2.2 How to update a service endpoint

When a service is enabled for your node, clicking on its name in the services menu on the left will show its information. To update any of the endpoints, simply change the one you need. A “Save” button will pop up to let you save your changes. Remember that you can cancel at any time.

## Manage Endpoints

Services

Nova

Cinder

Glance

Swift

Novav3

Block storage

Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.

Endpoints for Barcelona region

Public

Internal

Admin

Endpoints for Barcelona2 region

Public

Internal

Admin

Cancel

Save

Fig. 2: Enabling Cinder service for a certain region/regions. Input fields for endpoints interfaces immediately pop up.

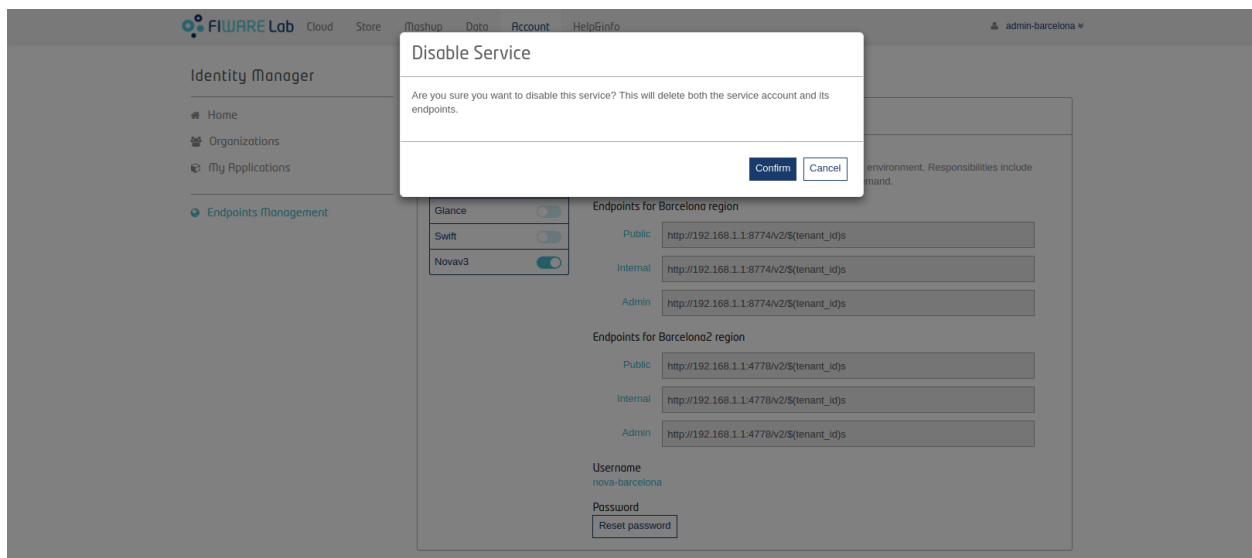


Fig. 3: Disabling Nova service for a certain region/regions.

## Manage Endpoints

Services

Nova

Cinder

Glance

Swift

Novav3

Compute

Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of machines on demand.

Endpoints for Barcelona region

Public

http://192.168.1.1:8774/v2/\$(tenant\_id)s

Internal

this is not a valid URL

Admin

http://192.168.1.1:8774/v2/\$(tenant\_id)s

Endpoints for Barcelona2 region

Public

http://192.168.1.1:4778/v2/\$(tenant\_id)s

Internal

http://192.168.1.1:4778/v2/\$(tenant\_id)s

Admin

http://192.168.1.1:4778/v2/\$(tenant\_id)s

Username

nova-barcelona

Password

Reset password

Cancel

Save

Fig. 4: Updating Nova endpoints. Validation of the input is performed, so as to make sure all endpoints are valid URLs.



### 9.2.3 Managing services accounts

When enabling a service in your node, a user account for it is created. However, for security purposes, the password will only be showed once. If you happen to forget it, just click the “Reset password” button to request a new one. The service account user name will remain the same.

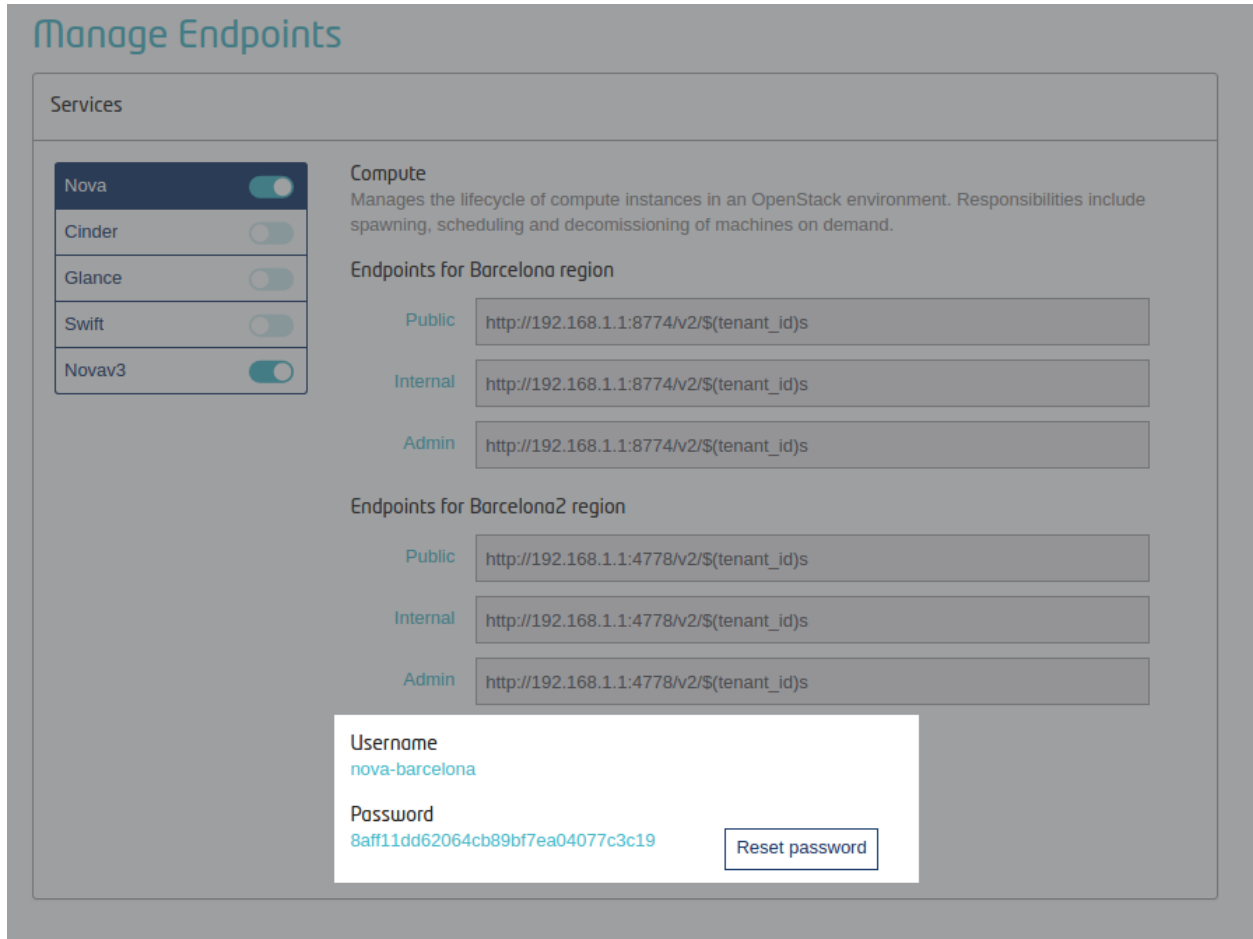


Fig. 5: Use the “Reset password” button to reset the service account. The service account section has been highlighted.



- *Introduction*
- *Horizon*
  - *Settings and Configuration*
    - \* *Local\_settings*
    - \* *Django settings.py*
- *Keystone*
- *django\_openstack\_auth*

## 10.1 Introduction

The intent of this guide is to cover more in-depth the implementation details, settings, problems encountered and their solutions, etc. of KeyRock to help developers that want to contribute or modify the code for their own custom use-cases. Additionally to this, all the components (Keystone, Horizon, the modified KeystoneClient library, etc.) can generate their own specific documentation using Sphinx with autodocs and code-level comments.

## 10.2 Horizon

This section covers all the Horizon related concepts.

### 10.2.1 Settings and Configuration

The base Horizon from OpenStack is a complex project and comes with lots of settings and several settings files. Some of them require configuration for the IdM to work, others are fine with the default values and a lot others are unused.

In this section we are going to cover the ones we need to set, for further reference please take a look at the [official documentation](#)

### Local\_settings

At `openstack_dashboard/local/local_settings.py`

- Identity API v3

We need to configure to use the Identity API v3 in our Keystone. Only matters to us the identity value. For example:

```
OPENSTACK_API_VERSIONS = {
    "data_processing": 1.1,
    "identity": 3,
    "volume": 2
}

OPENSTACK_HOST = "Keystone server IP address"
OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
```

- Email

Configure these for your outgoing email host or leave the default values for the console email backend

```
EMAIL_HOST = 'smtp.my-company.com'
EMAIL_PORT = 25
EMAIL_HOST_USER = 'djangomail'
EMAIL_HOST_PASSWORD = 'top-secret!'
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
```

- IdM account

Account for the IdM to perform tasks like user registration

```
OPENSTACK_KEYSTONE_ADMIN_CREDENTIALS = {
    'USERNAME': 'the_username',
    'PASSWORD': 'the_password',
    'PROJECT': 'the_projectname',
}
```

- FIWARE Applications and Roles.

These settings map to applications used in the FIWARE-Lab environment and are needed for automated tasks, for example granting the **Purchaser** role in the **Store** to any created organization. Depending on your use case you might need or want to modify them but normal installations in a *\*fiware-like\** environment wont need to change anything. Keep in mind that if your use case differs too much you might need to change the code to prevent some of this operations.

```
FIWARE_PURCHASER_ROLE_ID = 'the_id'
FIWARE_PROVIDER_ROLE_ID = 'the_id'
FIWARE_IDM_ADMIN_APP = 'idm'
FIWARE_CLOUD_APP = 'Cloud'
FIWARE_DEFAULT_CLOUD_ROLE_ID = 'the_id'
FIWARE_DEFAULT_APPS = [
    'Store',
]
```

- Keystone roles.

These settings map to normal keystone roles that are used by the IdM. As with the FIWARE Application and Roles settings, they depend on your use case.

```
KEYSTONE_OWNER_ROLE = 'owner'
KEYSTONE_TRIAL_ROLE = 'trial'
KEYSTONE_BASIC_ROLE = 'basic'
KEYSTONE_COMMUNITY_ROLE = 'community'
MAX_TRIAL_USERS = 100
OPENSTACK_KEYSTONE_ADMIN_ROLES = [
    KEYSTONE_OWNER_ROLE,
    'admin',
]
```

## Django settings.py

At `openstack_dashboard/settings.py`

We added some django apps, middleware, etc. You can check the file for reference but there is no configuration to be done there.

## 10.3 Keystone

### 10.4 django\_openstack\_auth