Keyrock to Keystone Federation

Requirements Analysis Document

Authors

Name and Surname	Email	Company
Alessandro M. Martellone	amartellone@create-net.org	Create-Net
Álvaro Alonso	aalonsog@dit.upm.es	UPM-DIT
Cyril Dangerville	cyril.dangerville@thalesgroup.com	Thales Services

Document history

Version number	Date	Notes	Authors	Reviewer
1	22/10/2015	First version	Alessandro Martellone (Create-Net) Álvaro Alonso (UPM-DIT)	Silvio Cretti (Create-Net)
2	05/11/2015	Improved “Functional requirements” and “Our solution” sections.	Alessandro Martellone (Create-Net) Álvaro Alonso (UPM-DIT)	Cyril Dangerville (Thales Services)

3	06/11/2015	New "OpenID Connect" section in State of the art, as alternative to SAML-based solution	Alessandro Martellone (Create-Net) Álvaro Alonso (UPM-DIT) Cyril Dangerville (Thales Services)	Alessandro Martellone (Create-Net)
4	22/11/2015	Reviewed by architects	Alessandro Martellone (Create-Net) Álvaro Alonso (UPM-DIT) Cyril Dangerville (Thales Services)	Alessandro Martellone (Create-Net)

Introduction

In the current architecture, the FIWARE Lab provides a centralized authentication and authorization system shared among all federation nodes. In particular we have recognized the following four main areas of intervention:

1. Make more reliable and fault tolerant the authentication and authorization system.
2. Allow a commercial/private cloud selected during the FI-Core Open Call or, more generally, a node of the FIWARE Lab federation to have its own Keystone and consequently its own set of users.
3. Allow a FIWARE community user to access a commercial/private cloud or, more generally, a node of the FIWARE Lab federation[1] limiting the SPOF due to the central authentication system (keystone)

[1] In the following we will use the words “remote node” to refer to either a commercial cloud node or a generic node of the FIWARE Lab federation where the central authorization and authentication system is not installed.

4. Resolve the privacy concern, enabling a node to store the user accounts in a local database.

Purpose

In this document the second and third points are tackled, proposing a solution for them, taking into account the functional and non-functional requirements below described. Furthermore the document presents, in the state of the art section, a brief analysis on a federated Keystone solution developed at CERN[2] and another one based on OpenID[3].
Out from the scope of document is designing a detailed solution for the first point. It will be addressed by multi-site KeyRock database replication, where those sites should be hosted in two different locations in Spain (thus mitigating the latency problem which could occur in a geographic distribution).
The fourth point is considered of lowest priority level then it is postponed in the next release.

Functional requirements

Two (or more) different cloud instances should be considered: a FIWARE Lab node which uses Keyrock 2.0 (its APIs are compatible with Keystone APIs version 2 and 3) for the Authentication and Authorization of users and services (AA) and another one (called “remote node”) which uses a “vanilla” Keystone. We would like to enable a FIWARE Lab user to use the resources of a remote node using the same credentials and avoiding to login again in the remote node. At the same time, a local user of the remote node should be able to use its services without limitations or drawbacks, continuing to use both the dashboard (e.g standard Horizon) and CLI. Nevertheless a remote node user can not use the FIWARE Lab resources, but he/she needs a FIWARE Lab account in order to do this.

In other words the FIWARE Lab Federation will continue to use a centralized KeyRock service (highly available and geographically distributed) for AA of all Federation’s nodes but it will support a distributed, independent and trusted local Keystone service which can act on behalf of central KeyRock to deliver cross-clouds authorization.

The solution should aim to minimize the configuration and software components installation activities on the remote node.

[2] http://home.cern/
[3] http://openid.net/

Use cases

Involved actors:

Federation Lab (FL) user: a registered account in FIWARE Lab.

External (Ext) user: a registered account in the remote cloud.

ID	1
Name	FL user uses the Federation services
Actor(s)	FL user
Preconditions	FL user has a valid account into Federation’s KeyRock with one or more associated projects on several regions.
Flow of events	1. FL user enters his/her credentials on the Cloud Portal’s login form. a. Cloud Portal authenticates the user on KeyRock and returns an auth-token. 2. The Cloud Portal shows, for each associated region, the related projects. 3. FL user selects a project and interacts with its OpenStack services. 4. FL user, uses OpenStack services a. Cloud Portal sends a request to the selected OpenStack service. b. The selected OpenStack service validates the related auth-token on KeyRock.
Postconditions	FL user is enabled to use the services provided by the Cloud Portal.

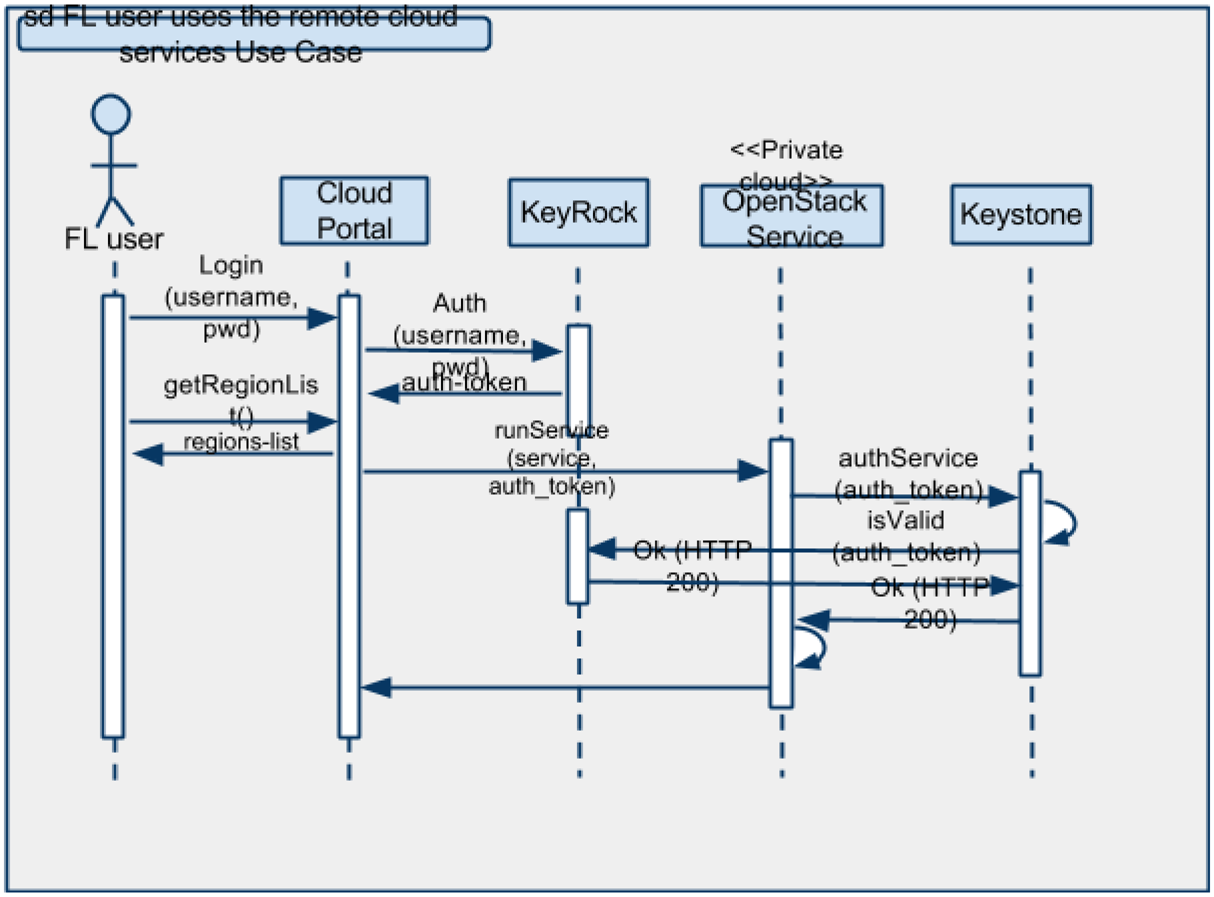
ID	2
Name	Ext user uses the remote cloud services
Actor(s)	Ext user

Preconditions	Ext user has a valid account into remote cloud Keystone with one or more associated projects on a single region.
Flow of events	1. Ext user enters his/her credentials on the cloud remote Horizon login form. a. Horizon authenticates the Ext user in the local KeyStone and returns an auth-token. 2. Horizon shows all projects associated to the user. 3. Ext user selects a project and interacts with its OpenStack services. 4. Ext user, uses OpenStack services a. Horizon sends a request to the selected OpenStack service. b. The selected OpenStack service validates the related auth-token on the local keystone
Postconditions	Ext user is enabled to use the services provided by the remote cloud.

ID	3
Name	FL user uses the remote cloud services
Actor(s)	FL user
Preconditions	FL user has a valid account into Federation's KeyRock with one or more associated projects on several regions. The remote

	cloud node is registered as node (region) into KeyRock. The KeyRock instance acts as a “trusted” Identity Provider.
Flow of events	1. FL user enters his/her credentials on the Cloud Portal’s login form. a. Cloud Portal authenticates the FL user on KeyRock and returns an auth-token. 2. The Cloud Portal shows, for each associated region, the related projects. 3. FL user selects the remote cloud node (region). 4. FL user selects a project and interacts with its OpenStack services. 5. All OpenStack services requested by the FL user (via Cloud Portal) are authorized (using the auth-token) sending a request to the selected remote Cloud OpenStack service. 6. The remote cloud OpenStack service validates the request (using the auth-token) on KeyRock.
Postconditions	FL user is enabled to use the services provided by the remote cloud.

The following sequence diagram describes the flow of events which occur in the use case 3.



Constraints, Non-Functional requirements

The designed solution should minimize the configuration procedure on the remote node avoiding to diverge from the current FIWARE Lab federation and limiting the updating impact on the remote Cloud.
Furthermore, the solution must not degrade the overall security of the existing implementation of both the FIWARE Lab and private clouds.

State of the art

Currently, there are two major federation protocols proposed for Keystone federation[4]:

- SAML;
- OpenID Connect.

[4] http://docs.openstack.org/developer/keystone/configure_federation.html

SAML

An SAML-based solution to federate different cloud instances has been proposed by the CERN [5] where they would like to move cloud workloads between multiple public cloud service providers ("CSP").
Starting from Kilo release a first implementation of Keystone to Keystone (K2K) has been released.

The solution is based on SAML[6] protocol and it considers the following simplified workflow:

1. A CERN user provides his credentials to Keystone
2. A token is returned back with a service catalog showing CERN openstack services (nova, swift, etc) and the CSP service catalog
3. Attempt to use the token against CSP nova service
4. CSP nova service calls CSP Keystone (no change)
5. CSP keystone deciphers the token belonging to the CERN Keystone IdP (which it sees as being setup as a trusted identity provider with attribute mappings we need to use)
6. CSP Keystone calls CERN Keystone (SAML or other federation protocol request and negotiation)
7. CERN Keystone shows CSP setup as a trusted service provider (with attribute mappings it should expect)
8. CERN Keystone returns back a SAML (or other federation protocol) assertion to CSP Keystone
9. CSP Keystone deciphers the assertion and provisions a temporary user. The token is retained valid and stored in CSP Keystone for future validation calls until expiration.

OpenID Connect

A solution based on the OAuth/OpenID Connect[7] standard is also proposed in the Keystone documentation[8]. In terms of workflow, this is similar to the SAML-based solution above. However, instead of complex XML/SOAP-based technologies (XML encryption/signature in particular), OpenID Connect uses simpler OAuth and JSON/REST standards with JOSE[9] and JWT[10] for token format, encryption and signature.

---

[5] https://blueprints.launchpad.net/keystone/+spec/keystone-to-keystone-federation
[6] https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language
[7] http://openid.net/connect/faq/
[8] General concepts: http://docs.openstack.org/developer/keystone/configure_federation.html. Concrete example using ForgeRock OpenAM IdP (OAuth/OpenID Connect provider): https://blueprints.launchpad.net/keystone/+spec/oath-openid-connect
[9] https://datatracker.ietf.org/wg/jose/documents/
[10] http://jwt.io/

The other benefit of OpenID Connect over SAML is quite relevant to FIWARE: as OpenID Connect extends OAuth and therefore uses OAuth at its core, it is a better fit for an OAuth provider like FIWARE KeyRock Identity Provider.

Evaluation of both SAML and OpenID Connect solutions

In both Keystone federation alternatives, the user entity authentication (the identity provider, or IdP) is separated from the cloud resources which the user wants to access (the service provider, or SP).

Furthermore, these solutions allow to use several external identity services, such as based on LDAP, Active Directory[11], SAML or OpenID Connect. As a result, if an external IdP service has a trusted relationship with the SP cloud, the latter can offer access to its resources, without the user needs to have explicitly registered an SP cloud account.

A good starting point in order to evaluate these solutions is the following documentation:

- http://docs.openstack.org/developer/keystone/configure_federation.html
- http://specs.openstack.org/openstack/keystone-specs/specs/juno/keystone-to-keystone-federation.html
- https://developer.rackspace.com/blog/keystone-to-keystone-federation-with-openstack-ansible/

After a first analysis the following pros and cons have been identified:

- Pros
    - a comprehensive set of use cases;
    - multiple Identity manager backends;
    - a solution supported by the community.
- Cons
    - for the Kilo release of OpenStack, federation is only partially supported[12];
    - does not support a reliable and tested backward compatibility among past OpenStack releases;
    - the IdP installation and configuration could have a strong impact on the current architecture (mainly on the Spanish node).

[11] https://msdn.microsoft.com/en-us/library/bb897402.aspx
[12] http://docs.openstack.org/developer/openstack-ansible/install-guide/configure-federation.html

Our solution

As introduced above we need to find a solution that covers the three use cases we have described:

1. A FL user uses the FIWARE Lab resources
2. An Ext user uses the remote Cloud resources
3. A FL user uses the remote Cloud resources

As we can see the two first cases are the way in which both environments work before the Private cloud joins the Federation. Thus, what we need to provide here is a solution that enables the third use case without modifying the two first ones. And this should be done in the most transparent way for both sides.

We propose to introduce a Keystone token driver that, installed in the remote cloud Keystone, to allow the token validation from the Openstack services remotely with the Keyrock. Thus, when a request from the Cloud Portal (authenticated with a token from the Keyrock) is sent to a remote cloud Openstack service, the service tries to validate the token in the local Keystone and, if it fails, the validation of the FL user token is delegated transparently to the Keyrock.

Currently there are several kind of tokens that could be used[13]:

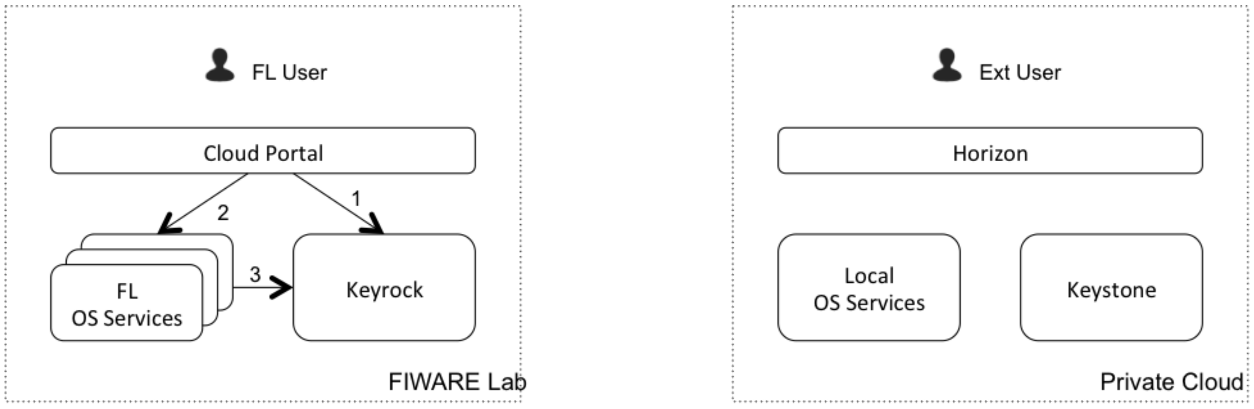
- UUID tokens are online validated by Keystone. It can be either persistently stored in a database (producing a lot of database traffic) or in memory through Memcached. Furthermore, the UUID token size is limited.
- PKI[14] tokens (solution based on a public/private certificate pair using X.509 technology) are non persistent cryptographic tokens, offline validated by each OpenStack service (through the Keystone middleware module included in). It is larger in size than a UUID token because each token includes an encrypted metadata containing the whole service catalog for each region, roles and the revocation lists.
- Fernet tokens are non persistent cryptographic based tokens and online validated by the Keystone service. Fernet tokens are more lightweight than PKI tokens and have a fixed size. It is supported by OpenStack Kilo release and above.

In this first stage we will design a solution based on UUID tokens which does not impact on the current system architecture.

In the following figures we can see the detailed interactions that take part in the solution.

[13] http://docs.openstack.org/developer/keystone/configuration.html
[14] https://wiki.openstack.org/wiki/PKI

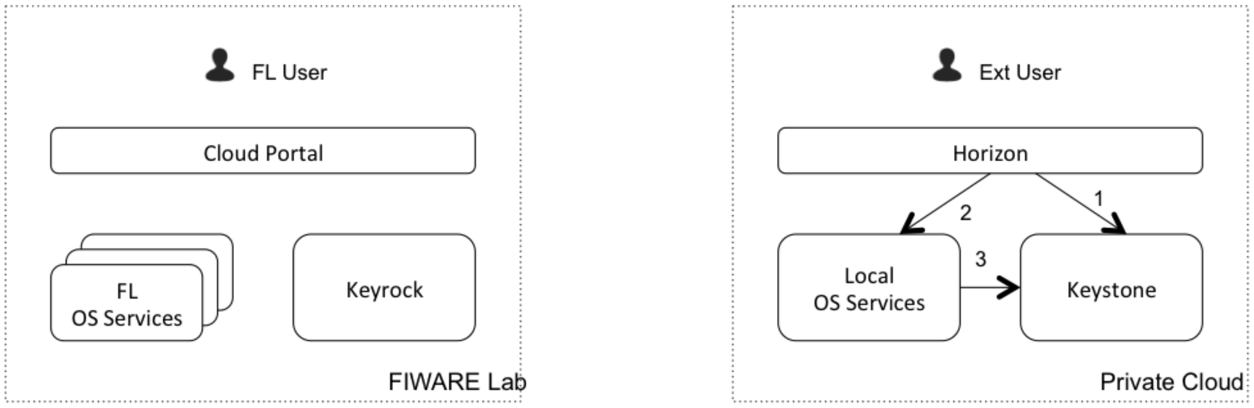
A FL user uses the FIWARE Lab resources



When a FL user uses the Federated Cloud resources, the process is the same as before:

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

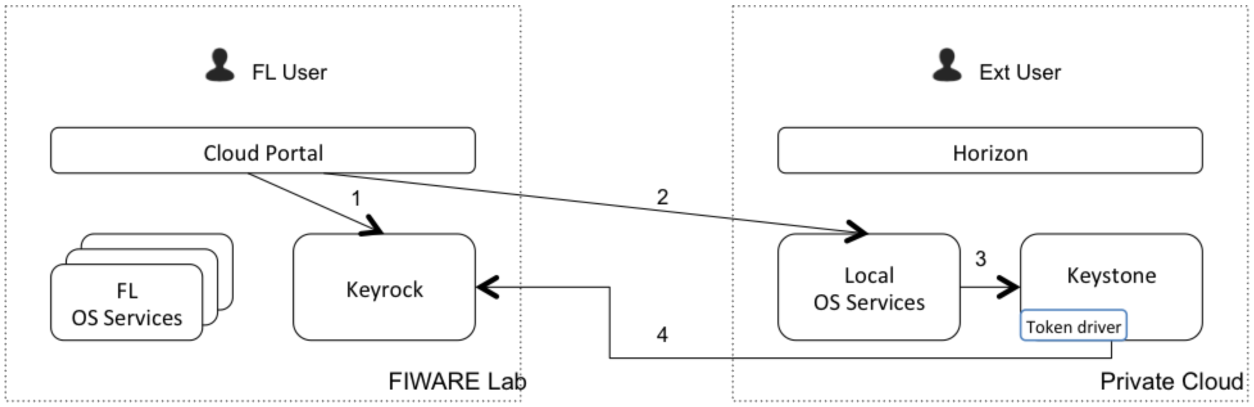
An Ext user uses the remote cloud a resources



When a Ext user uses the remote cloud resources, the process is the same as before:

1. Horizon authenticates the user in Keystone
2. Horizon sends a request to a local Openstack service
3. The Openstack service validates the token with Keystone

A FL user uses the remote cloud resources



When a FL user uses the remote Cloud resources, the new driver comes into play:

1. Cloud Portal authenticates the user in Keyrock
2. Cloud Portal sends a request (via APIs) to a remote Cloud Openstack service
3. Remote cloud Openstack service tries to validate the token in Keystone
4. As the token validation does not succeed (the token is not stored in remote Keystone), the remote Keystone validates it with Keyrock acting as a gateway, and sends the response to the remote Cloud OS Service.

Note: If the token validation succeeds, the remote Keystone stores the token locally (in cache), so the next times the step 4 is not required. Furthermore, for each service, the authorization is done locally (not using central KeyRock authorization), i.e. based on the local policy.json file and the user roles from the Keystone token. We consider that the basic roles and permissions are consistently configured into related policy.json files belonging to FIWARE Lab and Private Cloud nodes. In particular, if the remote Keystone admin creates, renames or removes roles locally, it is critical to keep them consistent with FIWARE Lab.