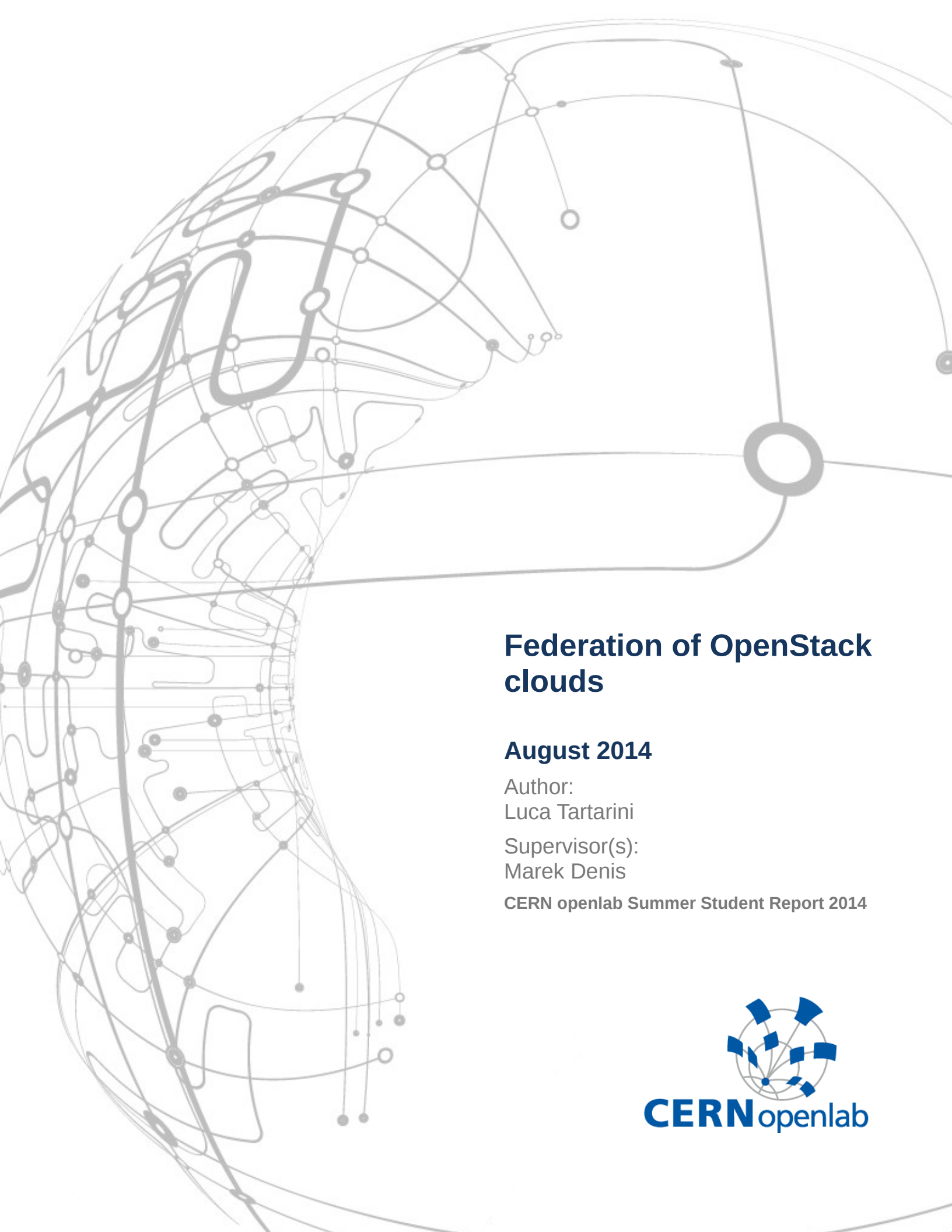

Federation of OpenStack clouds

August 2014

Author:
Luca Tartarini

Supervisor(s):
Marek Denis

CERN openlab Summer Student Report 2014

CERN openlab

Project Specification

Rackspace and CERN are implementing federated identity of OpenStack clouds within the OpenStack cloud project. The project is to enhance the client tools in OpenStack to support the federated identity functionalities, work with the open source community to incorporate these changes into the product and adapt the documentation and testing. The student will learn about the internals of OpenStack, federated identity techniques such as SAML and working with open source communities.

Abstract

The aim of this report is to describe and document the configuration steps of the Openlab Summer Student project.

The main goal of the project was to create a testbed for cloud federation for performing federation tests with multiple Identity Providers at the same time and test them with both browser and CLI (Command Line Interface).

At the beginning this report gives a general overview of the main concepts on which the project is based, with particular reference to Keystone, the OpenStack Identity Service, describing its main features and how it works. Later are described the protocols and the open source solutions and products (Shibboleth SP, Shibboleth IdP, Shibboleth EDS, ADFS) used for the creation and testing of the testbed.

In the following chapters is documented step by step the testbed's configuration. This is the main part of the report and it gives the details of how to install OpenStack and configure its Identity Service running in HTTPD with Shibboleth Service Provider. Then is described how to federate each Identity Provider with Keystone Identity Service providing the main configuration files.

The last part describes the testing in which each Identity Provider has been tested both via CLI (ECP - Enhanced Client or Proxy) and via web browser in order to receive from Keystone a token with which the end user could perform some OpenStack operations.

Table of Contents

1 Introduction

1.1 Cloud Computing

Cloud computing refers to a set of technologies and resources that provide computing services to the end users through the network. Cloud resources can be both hardware and software and are distribuited and virtualized over the network in a typical Client-Server architecture.

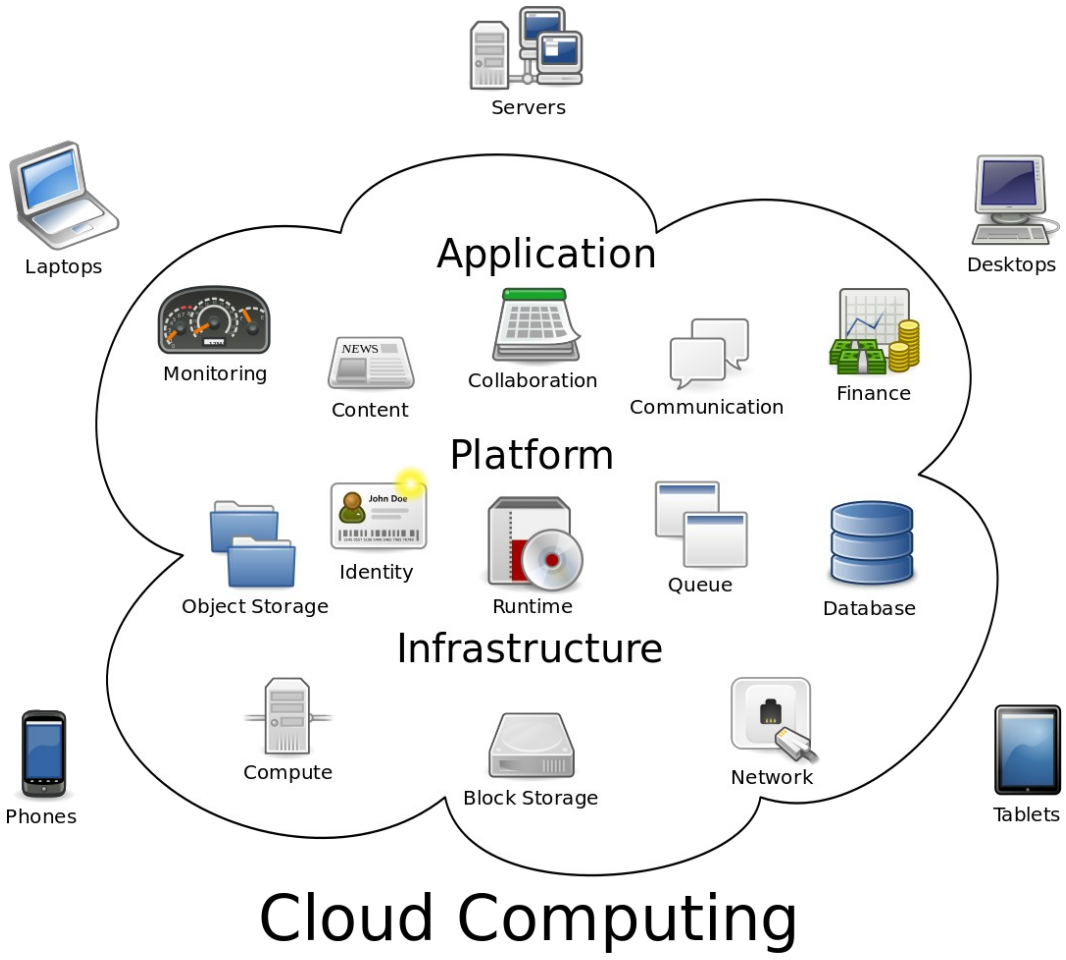

Figure 1. Cloud computing overview

We can distinguish three cloud computing services:

- IaaS (Infrastructure-as-a-Service): use of hardware resources in remote on request at the time when a platform needs them. The resources are typically virtual servers instances (an hypervisor or more often a pool of hypervisors are necessary to run the VMs). Examples of IaaS are OpenStack cloud computing environment and Amazon Web Services.

- PaaS (Platform-as-a-Service): use of provider's platform over the network. Developers can create and run applications on the provider's platform without the cost of buying and managing hardware.

- SaaS (Software-as-a-Service): use of software installed on remote servers (often accessible via web server). The provider supplies the infrastructure and the platform where the application runs.

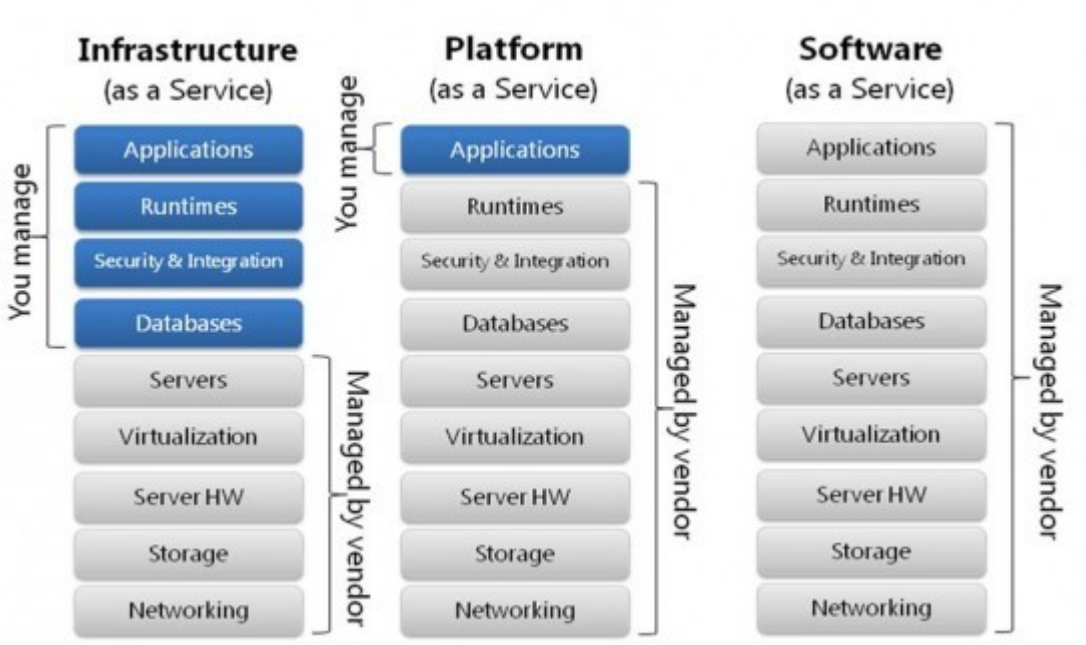

Figure 2. IaaS, PaaS and SaaS service models

Clouds can be classified in three different deployment models:

- Public cloud: a cloud that sells services to the end user over the network (Rackspace public cloud, Amazon Web Services, etc...).

- Private cloud: a proprietary cloud computing infrastructure that supplies services for a single organization (CERN private cloud).

- Hybrid cloud: an interconnection of two or more different cloud computing environments (private or public) through the network.

1.2 OpenStack

OpenStack is a cloud computing platform which provides an IaaS service model. It supplies to the end user processing, networking and storage resources provided by VMs running on the data center's hosts. OpenStack allows customers to manage their own resources and applications transparently, as if they saw a server to their dedicated.

1.2.1 OpenStack Components

The last version of OpenStack (Icehouse) consists in the following main services:

- OpenStack Compute (Nova): manages the VMs lifecycle, their scheduling and security.
- OpenStack Identity Service (Keystone): provides the user authentication and authorization for all OpenStack components.
- OpenStack Image Service (Glance): provides the images for the VMs.
- OpenStack Dashboard (Horizon): provides a web interface for admin and users.
- OpenStack Networking Service (Neutron): allows users to create networks.
- OpenStack Object Storage Service (Swift): a redundant storage system.
- OpenStack Block Storage Service (Cinder): provides persistent block-level storage.

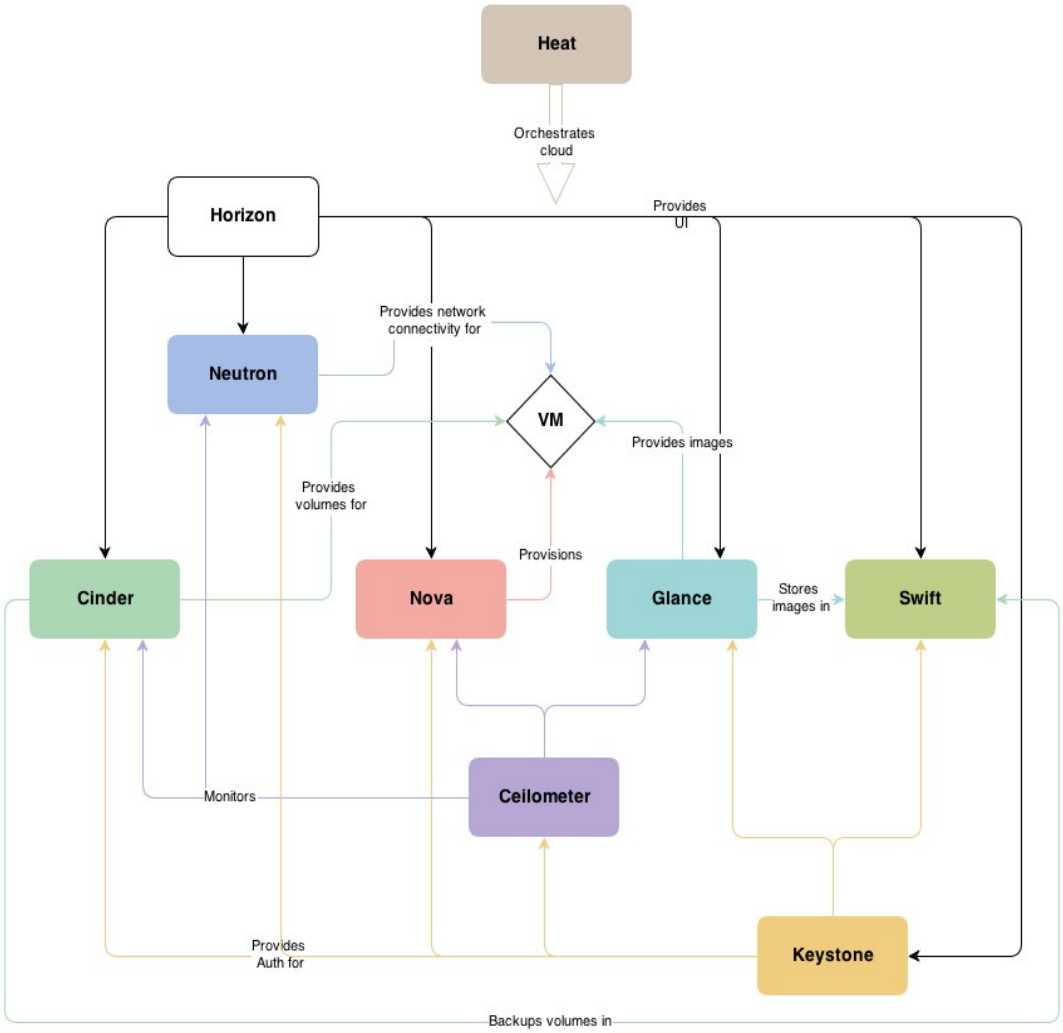

Figure 3. OpenStack architecture

1.2.2 Keystone

Keystone is a OpenStack service that provides token, policies and catalog via OpenStack API. It incorporates all information about users and their capabilities across other services. The main functions are:

- user authentication;
- grant token for authorization;
- policy management;
- manage catalog services.

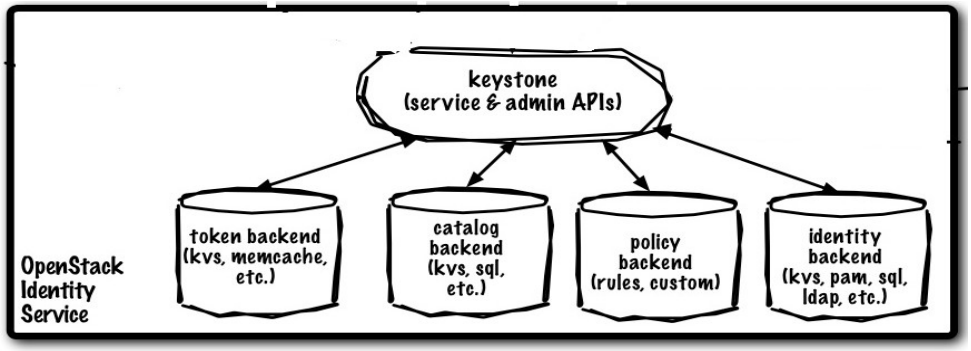

Figure 3. Keystone Identity Service overview

1.2.2.1 How Keystone works

To access some OpenStack service, users provide their credentials to Keystone Identity Service and receive a token with which users can perform the OpenStack commands. A token is given to user by Keystone if a valid username/password has been provided.

For instance, if a user wants to boot a new VM using Nova Service:

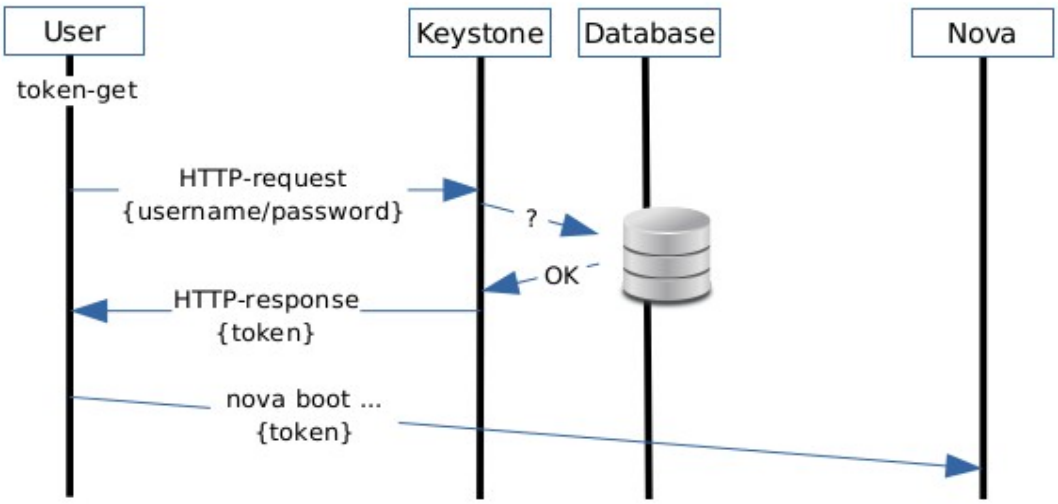

Figure 4. Example of Keystone usage

There are two token approach:

- Universally Unique IDentifier (UUID) token;
- Public Key Infrastructure (PKI) token.

1.2.2.2 UUID token

- User send to Keystone username and password;
- Keystone:
  - generates UUID token;
  - stores UUID token in its backend;
  - sends a copy of UUID token to user.
- User caches the token and sends it along with each API called;
- For each user request the API endpoint sends the UUID token received back to Keystone for validation;
- Keystone validates the UUID token received from API endpoint with the UUID token previously stored;
- Keystone returns “success” or “failure” message to the API endpoint.

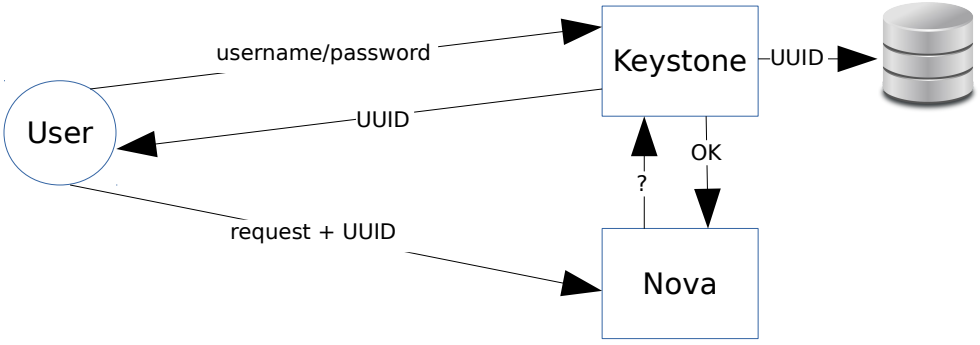

Figure 5. UUID token

1.2.2.3 PKI token

PKI token can be verified offline using the Keystone's public signing key. Each API endpoint holds a copy of Keystone's:

- Signing certificate;
- Revocation list;

- CA certificate.

The API endpoints use these information to validate the user requests without the need to validate each request to Keystone as in the case of UUID token.

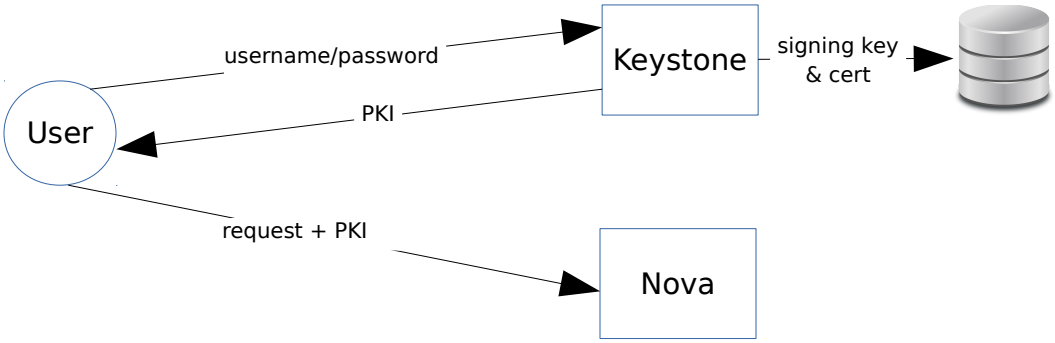

Figure 6. PKI token

2 Cloud Federation

Cloud federation refers to the interconnection of different service providers' cloud environments through the network. The main objective is to perform the login only once (SSO – Single Sign-On) with transparent access in multiple cloud computing environments in order to manage resources (move the workload between clouds, demand new resources, etc...) reducing the administrative overhead of distributing multiple authentication tokens to the user. Once authenticated by an external Identity Provider, the user can perform operations in the cloud environment in which is authenticated.

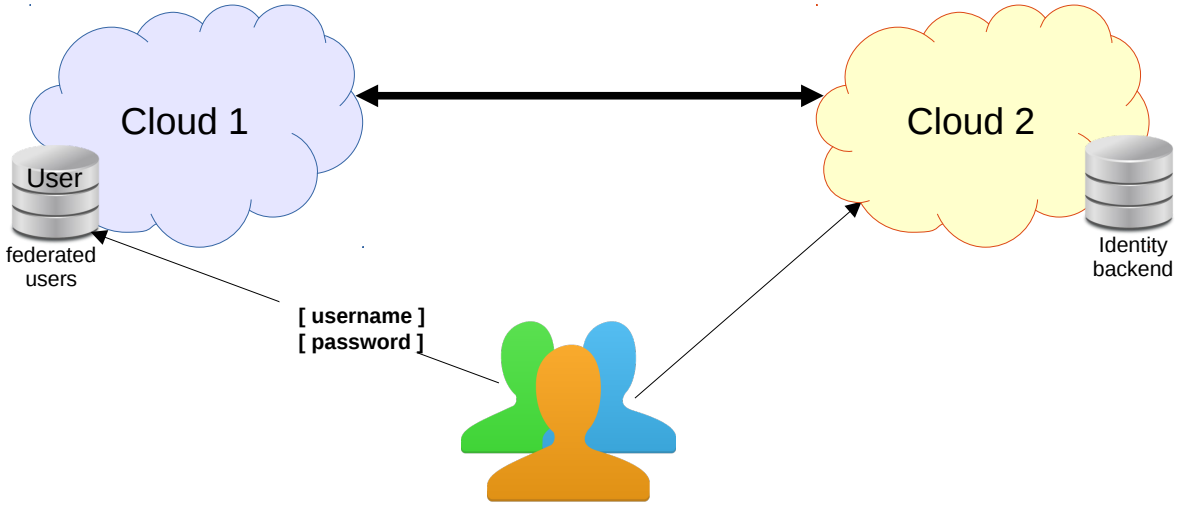

Figure 7. Cloud Federation

2.1 SAML2

It is an XML based protocol that implements the SSO's idea in different security domains for exchanging authentication and authorization data (assertions) between parties:

- Identity Provider: entity that stores information about users;
- Service Provider: entity that provides services to the end users;
- User (“principal”): entity that wants to authenticate to access a service.

The user need to be registered with at least on Identity Provider, the Identity Provider must provide to authenticate the user and the Service Provider relies on the Identity Provider to identify the user.

At user's request (phase 3 in the Figure 8 below), the Identity Provider must send a SAML assertion (it contains information about the user that wants to be authenticated) to the Service Provider specifying whether allow or deny the access to services for the user.

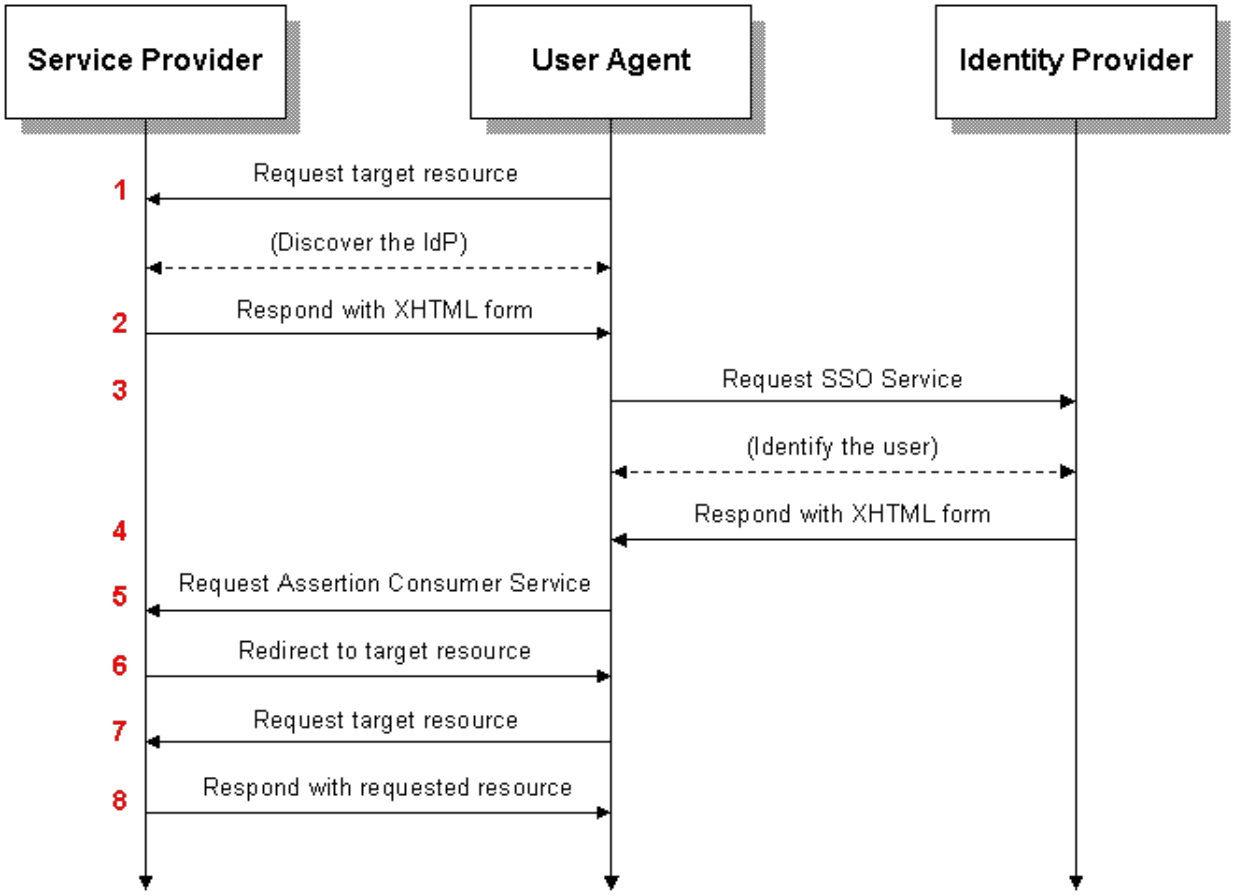

Figure 8. SAML2 Web Browser SSO

2.1.1 ECP (Enhanced Client or Proxy)

It is a CLI (Command Line Interface) extension with the particular feature that the client is not a browser. ECP is and adaptation of the SAML2 protocol used by web browser and it allows users to directly contact the Identity Providers avoiding the Identity Provider Discovery and Redirection by the Service Provider as in the case of the web browser.

The ECP is an intermediary between the Service Provider and the Identity Provider allowing the Service Provider to perform an authentication request without knowing the Identity Provider. ECP must support PAOS (reverse SOAP binding) which lets the Service Provider to obtain the assertion through the ECP, which is always directly accessible, unlike the Identity Provider.

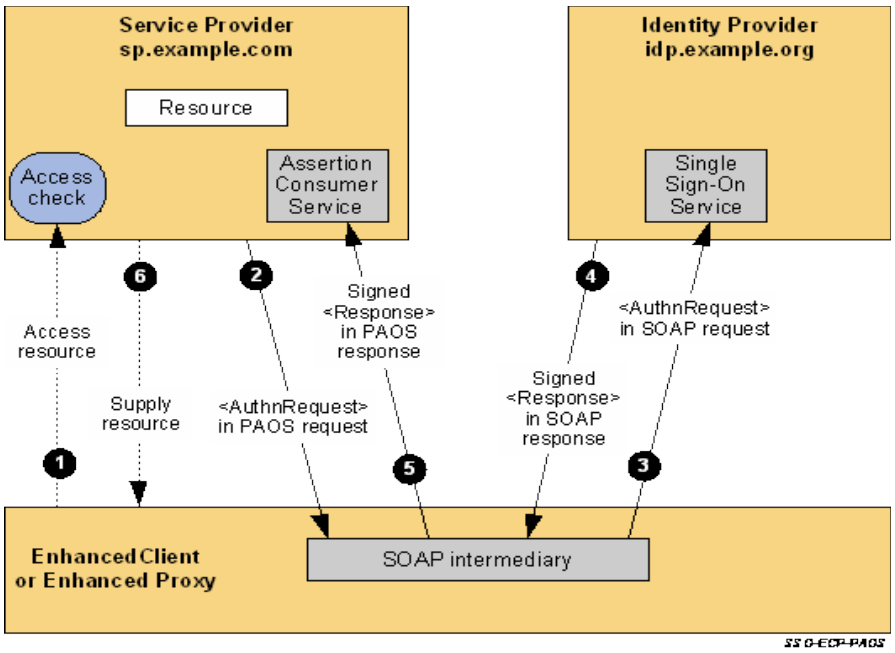

Figure 9. SAML2 ECP SSO

2.2 Shibboleth

Shibboleth is an open source federated identity solution built on SAML providing the Single Sign-On capabilities. It is a web-based technology that implements the HTTP/POST, artifact and attributes of SAML. The key concepts of Shibboleth are:

- Federated authentication;
- Access control based on attributes;
- Privacy management;
- Trust (implemented with metadata);
- Framework for multiple federations.

The Shibboleth's products used in the project are:

- Shibboleth Service Provider;
- Shibboleth Identity Provider (Testshib and IDPOPEN);
- Shibboleth Embedded Discovery Service.

In particular it has been used the Shibboleth 2.x version built on SAML 2.0 standards.

3 Testbed for Federation Tests

In the following sections I will describe how to configure a testbed for performing federation tests with multiple Identity Providers at the same time.

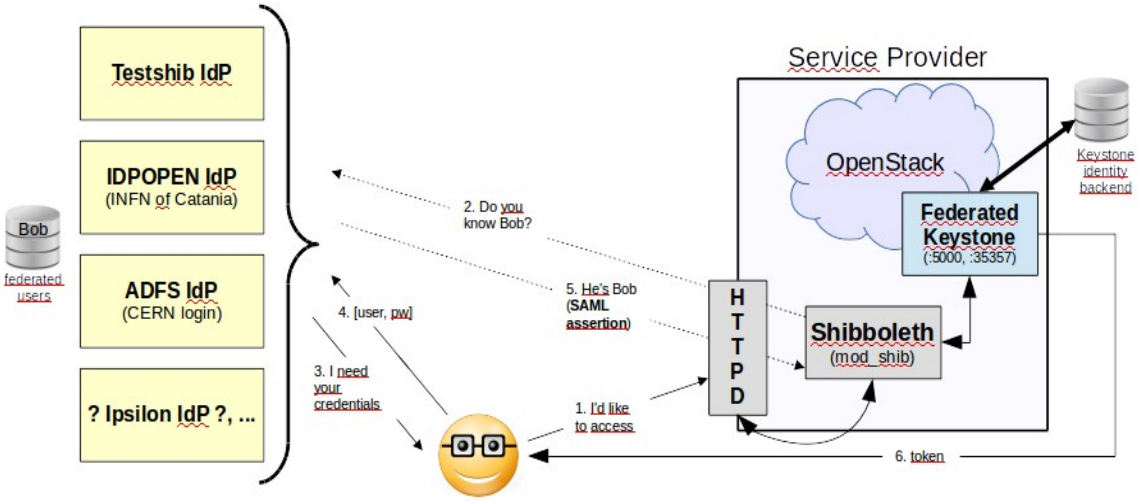

Figure 10. Testbed overview

The configuration has been performed on a OpenStack Virtual Machine running SLC6 CERN Server – x86_64, with OpenStack deployed via Packstack and Apache 2.2.15.

3.1 Install OpenStack via Packstack

After launching a new Virtual Machine via CLI (nova boot --flavor <flavor_id> --image <image_id> --key-name <key> <vm_name>) or Dashboard and connected to it via SSH (ssh -i <key.pem> username@<vm_name>.cern.ch), I installed OpenStack via Packstack following these configuration steps:

- setenforce 0
- /etc/init.d/iptables stop
- yum install -y http://rdo.fedorapeople.org/rdo-release.rpm
- yum install -y openstack-packstack

- For a single node OpenStack deployment:
  - packstack –allinone

Note:

- could be necessary to modify the priorities in /etc/yum.repos.d/ to allow the downloading of all the packages for Packstack:
  - modify the priority for each .repo;
  - yum clean-all
  - yum distro-sync
  - yum update
- during the installation something could go wrong, in case some workarounds may be useful:
  - http://openstack.redhat.com/Workarounds_2014_01
  - http://openstack.redhat.com/Workarounds_2014_02

Then we are ready to initialize the database:

- keystone-manage db_sync
- keystone-manage pki-setup –keystone-user keystone –keystone-group keystone

3.2 Configuring Keystone for Federation

The main concepts are:

- Keystone as a Service Provider;
- Keystone running in HTTPD with Shibboleth (mod_shib);
- Keystone consumes SAML assertions provided by an external Identity Provider.

Federated users are not stored in the Keystone database. An external Identity Provider is responsible for authenticating users, and communicates the result of authentication to Keystone using SAML assertions. Keystone will maps the SAML assertions received from the Identity Provider to Keystone user groups and assignments created in Keystone.

3.2.1 Keystone running in HTTPD with mod_shib

- Stop Keystone service:
  - service openstack-keystone stop

- chkconfig openstack-keystone off

- Install httpd (Apache 2.x);
- Install mod_nss;
- Install mod_wsgi;
- Install Shibboleth SP:
  - yum install shibboleth
  - Configure Shibboleth with 1+ IdPs (see Chapter 4)
  - service httpd restart
  - service shibd start
- Configure Keystone running in Apache's vhost listening standard port 5000 and 35357:
  - Create the wsgi-keystone.conf file;
  - set the alias that will match with the Federation URI: '/v3/OS-FEDERATION/identity_providers/.*?/protocols/.*?/auth';
  - copy it in /etc/httpd/conf.d/

```
WSGISocketPrefix /var/run/httpd

Listen 5000
<VirtualHost *:5000>
        WSGIScriptAliasMatch ^(/v3/OS-FEDERATION/identity_providers/.*?/protocols/.*?/auth)$ /var/www/cgi-bin/keystone/main/$1
  ##############################
  WSGIScriptAlias /secure /home/www/wsgi-scripts/wsgi_app.py
  <Directory /home/www/wsgi-scripts>
    Order allow,deny
    Allow from all
  </Directory>
  ##############################
  WSGIScriptAlias / /var/www/cgi-bin/keystone/main
          WSGIDaemonProcess keystone-public user=keystone group=keystone processes=3 threads=10 home=/usr/share/keystone
  WSGIProcessGroup keystone-public
  WSGIApplicationGroup %{GLOBAL}
  ErrorLog /var/log/httpd/keystone/keystone.log
  LogLevel debug
  CustomLog /var/log/httpd/keystone/access.log combined
  SSLEngine on
  SSLCertificateFile    /etc/ssl/certs/ca.crt
  SSLCertificateKeyFile /etc/ssl/private/ca.key
  <Directory /var/www/cgi-bin>
    Options FollowSymLinks
    AllowOverride All
    Order allow,deny
```

```
    Allow from all
  </Directory>
</VirtualHost>

Listen 35357
<VirtualHost *:35357>
  WSGIScriptAlias / /var/www/cgi-bin/keystone/admin
        WSGIDaemonProcess keystone-admin user=keystone group=keystone processes=3 threads=10
home=/usr/share/keystone
  WSGIProcessGroup keystone-admin
  WSGIApplicationGroup %{GLOBAL}
  ErrorLog /var/log/httpd/keystone/keystone.log
  LogLevel debug
  CustomLog /var/log/httpd/access.log combined
  <Directory /var/www/cgi-bin>
    Options FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from all
  </Directory>
</VirtualHost>

<Location /Shibboleth.sso>
                                        SetHandler shib
</Location>

<LocationMatch /v3/OS-FEDERATION/identity_providers/.*?/protocols/saml2/auth>
ShibRequestSetting requireSession 1
############################################################
#SSLRequireSSL   # The modules only work using HTTPS
AuthType shibboleth
ShibRequireSession On
ShibRequireAll On
ShibExportAssertion Off

Require valid-user
#Require ADFS_GROUP "Some Users Group" "Some Other Users Group"
############################################################

</LocationMatch>

<LocationMatch /v3/OS-FEDERATION/websso>
ShibRequestSetting requireSession 1
############################################################
#SSLRequireSSL   # The modules only work using HTTPS
AuthType shibboleth
ShibRequireSession On
ShibRequireAll On
ShibExportAssertion Off

Require valid-user
#Require ADFS_GROUP "Some Users Group" "Some Other Users Group"
############################################################

</LocationMatch>
```

- Hardlink the file:
    - /usr/share/openstack-puppet/modules/keystone/files/httpd/keystone.py

- in the /var/www/cgi-bin/keystone/ directory:
  - ln /usr/share/openstack-puppet/modules/keystone/files/httpd/keystone.py admin
  - ln /usr/share/openstack-puppet/modules/keystone/files/httpd/keystone.py main

3.2.2 OS-FEDERATION extension

- Enable the Federation extension in etc/keystone/keystone.conf file:

```
[federation]
driver = keystone.contrib.federation.backends.sql.Federation

[auth]
methods = external,password,token,saml2
saml2 = keystone.auth.plugins.saml2.Saml2
```

- Add the Federation extension in /usr/share/keystone/keystone-dist-paste.ini file:

```
[pipeline:api_v3]
pipeline = access_log sizelimit url_normalize token_auth admin_token_auth
xml_body json_body ec2_extension s3_extension federation_extension
service_v3
```

- Create the federation extension tables in Keystone database:
  - keystone-manage db_sync --extension federation

In the next chapter I am going to describe how to create Keystone groups and roles required for the mapping between the SAML assertions and the authorized federated users (mapped into local groups and roles) for each Identity Provider (see Chapter 4.4).

4 Federate Multiple Identity Providers

The following sections will describe how to configure the Service Provider with Keystone running on HTTPD to handle multiple Identity Providers at the same time. The Identity Providers that I federated are:

- Testshib IdP (http://www.testshib.org/);
- IDPOPEN IdP (https://idpopen.garr.it/);
- CERN IdP (ADFS).

The first two are Shibboleth Identity Providers, the last one is ADFS (Active Directory Federation Services) Identity Provider by Microsoft.

4.1 Trust between IdPs and SP (Metadata)

Download the metadata.xml files from the Identity Providers:

- Testshib IdP metadata: http://www.testshib.org/metadata/testshib-providers.xml
- IDPOPEN metadata: https://idpopen.garr.it/metadata/idp-metadata.xml
- CERN metadata: https://login-dev.cern.ch/adfs/XML/ADFS-metadata.xml

Upload the metadata from the Service Provider (in which Keystone is configured) to each Identity Provider creating a trust relationship between them, specify the following URL in the browser:

- https://your.server.name:5000/Shibboleth.sso/Metadata

and download the Metadata of your Service Provider, for example:

```
<!--
This is example metadata only. Do *NOT* supply it as is without review,
and do *NOT* provide it in real time to your partners.
 -->
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
ID="_39a2d033c1a364bc6078b1023e38a447913cdd08" entityID="https://your.server.name:5000/shibboleth">

  <md:Extensions xmlns:alg="urn:oasis:names:tc:SAML:metadata:algsupport">
    <alg:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>
    <alg:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha384"/>
    <alg:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <alg:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha224"/>
    <alg:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha224"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"/>
```

```
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha384"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2009/xmldsig11#dsa-sha256"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha1"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <alg:SigningMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
  </md:Extensions>

                    <md:SPSSODescriptor            protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol
urn:oasis:names:tc:SAML:1.1:protocol                                    urn:oasis:names:tc:SAML:1.0:protocol
http://schemas.xmlsoap.org/ws/2003/07/secext">
    <md:Extensions>
                                    <init:RequestInitiator       xmlns:init="urn:oasis:names:tc:SAML:profiles:SSO:request-init"
Binding="urn:oasis:names:tc:SAML:profiles:SSO:request-init"
Location="https://your.server.name:5000/Shibboleth.sso/Login"/>
    </md:Extensions>
    <md:KeyDescriptor>
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:KeyName>yourKeyName</ds:KeyName>
        <ds:X509Data>
          <ds:X509SubjectName>CN=yourCN</ds:X509SubjectName>
            <ds:X509Certificate>MIIC4jCCAcqgAwIBAgIJAOz9zh27F+WYMA0GCSqGSIb3DQEBBQUAMBQxEjAQ
BgNV
BAMTCWx0YXJ0YXJpMjAeFw0xNDA3MDMwODUxMzBaFw0yNDA2MzAwODUxMzBaMBQx
EjAQBgNVBAMTCWx0YXJ0YXJpMjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
ggEBAL26Nx6EFYqpuE7CxCMveLHAxrs/m8AedOq7tniiSR2Sk8xkjdITbKCCd7jT
xT1LV6q9+4u8p7dUufWyhpxDauluIvad+3ofrx6heeJfBCtB6nnkNugLXkX2RG5n
UndhTzJllal+jbMCBAwuA/Jg+hHrPSYvLZrj0bRDCShojcZjjrT/3ikYhn/hQq56
5b6BcFQJGeO4L68nIRFG06mXG3CuU3UxboYm5QiDHgQQ68W9PlZtLi+9v/Pku8HD
u7U39uYvBYdPyEs1La0FLUjXYWDs8276I81bf6CJyWbJ0stfqEdXOq+yOd4weaxa
oVuDY/W2tjjJAaRRTu9+u2Sozw0CAwEAAaM3MDUwFAYDVR0RBA0wC4IJbHRhcnRh
cmkyMB0GA1UdDgQWBBSDtjZ2u3DokvBYdxqjJkGXtWERGzANBgkqhkiG9w0BAQUF
AAOCAQEAFRIPViFI5vRfUs7kE5Kqc25cDn+qo933/Nbhpngde0ATG2IJB9MHN+63
uYZKFxhha5j3CexEl+7OgUPoH57i/n2AUb2o7WKPe5DAkOL0cFwxYGSrvBeWE1zW
PStIQhlzqe+1DieNKs4KZ0cA17b2tfK4YA3SD7+MH1MA0cy6lsUpToy4HK0lWsGa
+PBNGklqEcOtQ7IhpRFQLha/k93ZNCUWOdIzgR7fWCAekkaQ0FqAEVnyiAmDazPA
K7BsL6YgOeRFqpC1vlFN/FCUYOicm66cPyoTvpQqn35VfYdNsodiyr/WtiuXzZsc
a9D7MdH+KY2VHpszkKGMqGXp+TBuvA==
</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2009/xmlenc11#aes128-gcm"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2009/xmlenc11#aes192-gcm"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes192-cbc"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes256-cbc"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2009/xmlenc11#rsa-oaep"/>
      <md:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
    </md:KeyDescriptor>
                            <md:ArtifactResolutionService        Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://your.server.name:5000/Shibboleth.sso/Artifact/SOAP" index="1"/>
                                <md:SingleLogoutService          Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://your.server.name:5000/Shibboleth.sso/SLO/SOAP"/>
                            <md:SingleLogoutService       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://your.server.name:5000/Shibboleth.sso/SLO/Redirect"/>
                              <md:SingleLogoutService       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://your.server.name:5000/Shibboleth.sso/SLO/POST"/>
                            <md:SingleLogoutService       Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://your.server.name:5000/Shibboleth.sso/SLO/Artifact"/>
```

```
                                <md:SingleLogoutService          Binding="http://schemas.xmlsoap.org/ws/2005/02/trust"
Location="https://your.server.name:5000/Shibboleth.sso/ADFS"/>
                        <md:AssertionConsumerService      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://your.server.name:5000/Shibboleth.sso/SAML2/POST" index="1"/>
          <md:AssertionConsumerService   Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://your.server.name:5000/Shibboleth.sso/SAML2/POST-SimpleSign" index="2"/>
                      <md:AssertionConsumerService      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://your.server.name:5000/Shibboleth.sso/SAML2/Artifact" index="3"/>
                             <md:AssertionConsumerService        Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="https://your.server.name:5000/Shibboleth.sso/SAML2/ECP" index="4"/>
                        <md:AssertionConsumerService      Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
Location="https://your.server.name:5000/Shibboleth.sso/SAML/POST" index="5"/>
                           <md:AssertionConsumerService       Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
Location="https://your.server.name:5000/Shibboleth.sso/SAML/Artifact" index="6"/>
                             <md:AssertionConsumerService       Binding="http://schemas.xmlsoap.org/ws/2003/07/secext"
Location="https://your.server.name:5000/Shibboleth.sso/ADFS" index="7"/>
  </md:SPSSODescriptor>

</md:EntityDescriptor>
```

Then you need to upload the metadata to each the Identity Provider according to its procedure.

4.2 Configure Shibboleth working with Multiple IdPs via ECP at the same time

In the /etc/shibboleth2.xml file:

- in order to avoid the systematic choice of the attribute(s) to use as REMOTE_USER, in the <ApplicationDefaults> element remove:

```
REMOTE_USER="eppn persistent-id targeted-id"
```

- in order to enable ECP support and to allow > 1 IdPs (this won't affect normal access by browsers), in the <SSO> element remove:

```
entityID="https://idp.example.org/shibboleth"
discoveryProtocol="SAMLDS"
discoveryURL="https://ds.example.org/DS/WAYF"
```

and add:

```
ECP=”true”
```

4.2.1 Multiple Shibboleth IdPs (Testshib and IDPOPEN)

- copy the metadata.xml files previously download in the directory: /etc/shibboleth/
- update the <Handler type=”Session”> element:

```
<Handler type="Session" Location="/Session" showAttributeValues="true"/>
```

the attribute ‘showAttributeValues’ must to be true in order to include the actual attribute values in the session;

- add as many <MetadataProvider> element as there are IdPs:

```
<MetadataProvider type="XML" uri="http://www.testshib.org/metadata/testshib-providers.xml" backingFilePath="/etc/shibboleth/testshib-two-idp-metadata.xml” reloadInterval="180000"/>
<MetadataProvider type="XML" uri="https://idpopen.garr.it/metadata/idp-metadata.xml" backingFilePath="/etc/shibboleth/metadata-idpopen.xml" reloadInterval="180000"/>
```

specifying the URI in which to retrieve metadata and the path in which the metadata.xml files previously download are stored.

4.2.2 ADFS Identity Provider in a multiple IdPs environment

In the /etc/shibboleth2.xml file:

- add the <OutOfProcess> element specifying the extension libraries:

```
<OutOfProcess logger="/etc/shibboleth/shibd.logger">
    <Extensions>
      <Library path="adfs.so" fatal="true"/>
    </Extensions>
</OutOfProcess>
```

- add the <InProcess> element specifying extension libraries and supplementing the native IIS configuration:

```
<InProcess logger="/etc/shibboleth/native.logger">
    <Extensions>
      <Library path="adfs-lite.so" fatal="true"/>
    </Extensions>
    <ISAPI normalizeRequest="true">
      <Site id="1" name="ltartari2.cern.ch:5000"/>
    </ISAPI>
</InProcess>
```

- add the <md:AssertionConsumerService> element to set the handler that is responsible for consuming the SAML assertion:

```
<md:AssertionConsumerService Location="/ADFS" isDefault="true" index="1"
            Binding="http://schemas.xmlsoap.org/ws/2003/07/secext" ResponseLocation="/shibboleth-sp/wsignout.gif"/>
```

- add the <saml:Audience> element to handle SAML products that produce unusual assertions that otherwise can't be processed:

```
<saml:Audience>https://ltartari2.cern.ch:5000/Shibboleth.sso/ADFS</saml:Audience>
```

4.2.3 shibboleth2.xml file

An example of your /etc/shibboleth/shibboleth2.xml (with both Shibboleth IdP and ADFS IdP) may look like:

```
<SPConfig xmlns="urn:mace:shibboleth:2.0:native:sp:config"
   xmlns:conf="urn:mace:shibboleth:2.0:native:sp:config"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
   clockSkew="180">

   <!-- The OutOfProcess section contains properties affecting the shibd daemon. -->
   <OutOfProcess logger="/etc/shibboleth/shibd.logger">
      <Extensions>
                                                <Library path="adfs.so" fatal="true"/>
      </Extensions>
   </OutOfProcess>

   <!-- The InProcess section conrains settings affecting web server modules/filters. -->
   <InProcess logger="/etc/shibboleth/native.logger">
      <Extensions>
                                              <Library path="adfs-lite.so" fatal="true"/>
      </Extensions>
      <ISAPI normalizeRequest="true">
         <!--
         Maps IIS Instance ID values to the host scheme/name/port/sslport. The name is
         required so that the proper <Host> in the request map above is found without
         having to cover every possible DNS/IP combination the user might enter.
         The port and scheme can    usually be omitted, so the HTTP request's port and
         scheme will be used.
         -->
         <Site id="1" name="ltartari2.cern.ch:5000"/>
      </ISAPI>
   </InProcess>

   <!--
   By default, in-memory StorageService, ReplayCache, ArtifactMap, and SessionCache
   are used. See example-shibboleth2.xml for samples of explicitly configuring them.
   -->

   <!--
   To customize behavior for specific resources on Apache, and to link vhosts or
   resources to ApplicationOverride settings below, use web server options/commands.
   See https://spaces.internet2.edu/display/SHIB2/NativeSPConfigurationElements for help.

   For examples with the RequestMap XML syntax instead, see the example-shibboleth2.xml
   file, and the https://spaces.internet2.edu/display/SHIB2/NativeSPRequestMapHowTo topic.
   -->

   <!-- The ApplicationDefaults element is where most of Shibboleth's SAML bits are defined. -->
   <ApplicationDefaults entityID="https://ltartari2.cern.ch:5000/shibboleth">

      <!--
      Controls session lifetimes, address checks, cookie handling, and the protocol handlers.
      You MUST supply an effectively unique handlerURL value for each of your applications.
      The value defaults to /Shibboleth.sso, and should be a relative path, with the SP computing
      a relative value based on the virtual host. Using handlerSSL="true", the default, will force
      the protocol to be https. You should also add a cookieProps setting of "; path=/; secure"
      in that case. Note that while we default checkAddress to "false", this has a negative
      impact on the security of the SP. Stealing cookies/sessions is much easier with this disabled.
```

```
    -->
    <Sessions lifetime="28800" timeout="3600" checkAddress="false" relayState="ss:mem" handlerSSL="false">

        <!--
        Configures SSO for a default IdP. To allow for >1 IdP, remove
        entityID property and adjust discoveryURL to point to discovery service.
        (Set discoveryProtocol to "WAYF" for legacy Shibboleth WAYF support.)
        You can also override entityID on /Login query string, or in RequestMap/htaccess.
        -->
        <SSO ECP="true">SAML2 SAML1</SSO>

        <!-- SAML and local-only logout. -->
        <Logout>SAML2 Local</Logout>

                                                        <!--
        md:AssertionConsumerService locations handle specific SSO protocol bindings,
        such as SAML 2.0 POST or SAML 1.1 Artifact. The isDefault and index attributes
        are used when sessions are initiated to determine how to tell the IdP where and
        how to return the response.
        -->
                        <md:AssertionConsumerService Location="/ADFS" isDefault="true" index="1"
                                                                Binding="http://schemas.xmlsoap.org/ws/2003/07/secext"
ResponseLocation="/shibboleth-sp/wsignout.gif"/>

        <!-- Extension service that generates "approximate" metadata based on SP configuration. -->
        <Handler type="MetadataGenerator" Location="/Metadata" signing="false"/>

        <!-- Status reporting service. -->
        <Handler type="Status" Location="/Status" acl="127.0.0.1"/>

        <!-- Session diagnostic service. -->
        <Handler type="Session" Location="/Session" showAttributeValues="true"/>

        <!-- JSON feed of discovery information. -->
        <Handler type="DiscoveryFeed" Location="/DiscoFeed"/>
    </Sessions>

    <!--
    Allows overriding of error template information/filenames. You can
    also add attributes with values that can be plugged into the templates.
    -->
    <Errors supportContact="root@localhost"
        logoLocation="/shibboleth-sp/logo.jpg"
        styleSheet="/shibboleth-sp/main.css"/>

                        <!-- Uncomment and modify to tweak settings for specific IdPs or groups. -->
    <!-- <RelyingParty Name="SpecialFederation" keyName="SpecialKey"/> -->
                    <saml:Audience>https://ltartari2.cern.ch:5000/Shibboleth.sso/ADFS</saml:Audience>

    <!-- Example of remotely supplied batch of signed metadata. -->
    <!--
    <MetadataProvider type="XML" uri="http://federation.org/federation-metadata.xml"
          backingFilePath="federation-metadata.xml" reloadInterval="7200">
        <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/>
        <MetadataFilter type="Signature" certificate="fedsigner.pem"/>
    </MetadataProvider>
    -->

    <!-- Example of locally maintained metadata. -->
    <!--
    <MetadataProvider type="XML" file="partner-metadata.xml"/>
```

```
    -->

                                        <!-- Multiple IdPs MetadataProvider -->
    <MetadataProvider type="XML" uri="http://www.testshib.org/metadata/testshib-providers.xml"
                      backingFilePath="testshib-two-idp-metadata.xml" reloadInterval="180000"/>

            <MetadataProvider type="XML" uri="https://idpopen.garr.it/metadata/idp-metadata.xml"
                                                        backingFilePath="/etc/shibboleth/metadata-idpopen.xml"
reloadInterval="180000"/>

          <MetadataProvider type="XML" uri="https://login-dev.cern.ch/adfs/XML/ADFS-metadata.xml"
                      backingFilePath="/etc/shibboleth/ADFS-metadata.xml" reloadInterval="7200" />

    <!-- Map to extract attributes from SAML assertions. -->
    <AttributeExtractor type="XML" validate="true" path="attribute-map.xml"/>

    <!-- Use a SAML query if no attributes are supplied during SSO. -->
    <AttributeResolver type="Query" subjectMatch="true"/>

    <!-- Default filtering policy for recognized attributes, lets other data pass. -->
    <AttributeFilter type="XML" validate="true" path="attribute-policy.xml"/>

    <!-- Simple file-based resolver for using a single keypair. -->
    <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>

    <!--
    The default settings can be overridden by creating ApplicationOverride elements (see
    the https://spaces.internet2.edu/display/SHIB2/NativeSPApplicationOverride topic).
    Resource requests are mapped by web server commands, or the RequestMapper, to an
    applicationId setting.

    Example of a second application (for a second vhost) that has a different entityID.
    Resources on the vhost would map to an applicationId of "admin":
    -->
    <!--
    <ApplicationOverride id="admin" entityID="https://admin.example.org/shibboleth"/>
    -->
  </ApplicationDefaults>

  <!-- Policies that determine how to process and authenticate runtime messages. -->
  <SecurityPolicyProvider type="XML" validate="true" path="security-policy.xml"/>

  <!-- Low-level configuration about protocols and bindings available for use. -->
  <ProtocolProvider type="XML" validate="true" reloadChanges="false" path="protocols.xml"/>

</SPConfig>
```

4.3 Attribute mapping

In order to process the SAML assertions about the user is necessary to define how to process that information. The Service Provider translates the SAML assertions using the attribute-map.xml configuration file in which are defined the mapping rules between the SAML information and the environment variables.

To add a new mapping rule you need to define a new <Attribute> element, for example:

```
<Attribute name="EmailAddress" nameFormat="http://schemas.xmlsoap.org/claims" id="EMAIL"/>
```

where the ‘id’ attribute is the identifier for the variable you want to map.

Note:

- In the case of multiple Identity Providers at the same time you can not map attributes differently based on the Identity Provider, eventually you can separate the resource URLs but the requests will be different.
- An option is to concatenate all the mapping rules in the same attribute-map.xml configuration file:

```
<!-- Testshib attributes -->
<Attribute name="urn:oid:2.5.4.42" id="Testshib_givenName"/>
…

<!-- IDPOPEN attributes -->
<Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="IDPOPEN_mail"/>
…

<!-- ADFS attributes -->
<Attribute name="Lastname" nameFormat="http://schemas.xmlsoap.org/claims" id="ADFS_lastname"/>
...
```

4.4 Keystone Mapping

At this point is necessary to configure Keystone for the mapping with the SAML assertion received from the Identity Providers. The entities to create are:

1. Identity Provider: represents the Identity Provider responsible for the user authentication;
2. Mapping: a list of rules to map federation protocol attributes to Identity API objects;
3. Protocol: information to decide which mapping rule to use for an incoming request.

These are the steps to configure the Keystone mapping (via HTTP requests) using the Federation extension API v3 and the Identity API v3:

- Create a domain:
  - POST /v3/domains

```
{
    "description": "Federation Domain",
    "enabled": true,
    "id": "<fed_domain_id>",
    "name": "federation-domain"
}
```

- Create a shared project for all Identity Provider:
  - POST /v3/projects

```
{
```

```
        "description": "Project for federation test",
        "domain_id": "<fed_domain_id>",
        "enabled": true,
        "id": "<fed_project_id>",
        "name": "federation-project"
}
```


- Create a project for each Identity Provider:


```
{
        "description": "Project for adfs",
        "domain_id": "<fed_domain_id>",
        "enabled": true,
        "id": "<adfs_project_id>",
        "name": "adfs-project"
},
{
        "description": "Project for testshib idp",
        "domain_id": "<fed_domain_id>",
        "enabled": true,
        "id": "<testshib_project_id>",
        "name": "testshib-project"
},
{
        "description": "Project for idpopen idp",
        "domain_id": "<fed_domain_id>",
        "enabled": true,
        "id": "<idpopen_project_id>",
        "links": {
        "name": "idpopen-project"
}
```


- Create a group for each Identity Provider:
  - POST /v3/groups


```
{
        "description": "Group for idpopen idp",
        "domain_id": "<fed_domain_id>",
        "id": "<idpopen_group_id>",
        "name": "idpopen-group"
},
{
        "description": "Group for testshib idp",
        "domain_id": "<fed_domain_id>",
        "id": "<testshib_group_id>",
        "name": "testshib-group"
},
{
        "description": "Group for adfs",
        "domain_id": "<fed_domain_id>",
        "id": "<adfs_group_id>",
        "name": "adfs-group"
}
```


- Create a role for each group:
  - POST /v3/roles


```
{
        "id": "<idpopen_role_id>",
        "name": "idpopen-role"
},
{
        "id": "<testshib_role_id>",
        "name": "testshib-role"
},
```

```
{
        "id": "<adfs_role_id>",
        "name": "adfs-role"
}
```


- Grant each role to the specified domain group:
    - PUT /v3/domains/<fed_domain_id>/groups/<testshib_group_id>/roles/<testshib_role_id>
    - PUT /v3/domains/<fed_domain_id>/groups/<idpopen_group_id>/roles/<idpopen_role_id>
    - PUT /v3/domains/<fed_domain_id>/groups/<adfs_group_id>/roles/<adfs_role_id>
- Grant each role to the specified project group and to the shared project:
    - PUT /v3/projects/<fed_project_id>}/groups/<testshib_group_id>/roles/<testshib_role_id>
    - PUT /v3/projects/<fed_project_id>}/groups/<idpopen_group_id>/roles/<idpopen_role_id>
    - PUT /v3/projects/<fed_project_id>}/groups/<adfs_group_id>/roles/<adfs_role_id>
    - PUT /v3/projects/<testshib_project_id>}/groups/<testshib_group_id>/roles/<testshib_role_id>
    - PUT /v3/projects/<idpopen_project_id>}/groups/<idpopen_group_id>/roles/<idpopen_role_id>
    - PUT /v3/projects/<adfs_project_id>}/groups/<adfs_group_id>/roles/<adfs_role_id>
- Create each Identity Provider:
    - PUT /OS-FEDERATION/identity_providers/testshib
    - PUT /OS-FEDERATION/identity_providers/idpopen
    - PUT /OS-FEDERATION/identity_providers/adfs


```
{
        "description": "adfs",
        "enabled": true,
        "id": "adfs"
},
{
        "description": "idpopen @ https://idpopen.garr.it",
        "enabled": true,
        "id": "idpopen"
},
{
        "description": "testshib @ https://www.testshib.org",
        "enabled": true,
```

```
        "id": "testshib"
}
```

- Create each mapping:
  - PUT /OS-FEDERATION/mappings/testshib_mapping
  - PUT /OS-FEDERATION/mappings/idpopen_mapping
  - PUT /OS-FEDERATION/mappings/adfs_mapping

```
{
        "id": "idpopen_mapping",
        "rules": [ {
                "local": [ {
                                "user": { “name": "idpopen user" } },
                                { "group": { "id": "<idpopen_group_id>" } } ],
                "remote": [ {
                                "type": "mail",
                                "any_one_of": ["<email>”]}
                                                                                ]}
                                                ]
}
…
```

Note:

- for each mapping is necessary to specify the group identifier previously created;
- the value of the key 'type' must be an attribute of the environment variables produced from the SAML assertion by the attribute-map.xml file;
- the value of the key 'any_one_of' must be any one of the values of the attribute specified in the previous step.

- Create SAML2 protocol for each Identity Provider:
  - PUT /OS-FEDERATION/identity_providers/testshib/protocols/saml2
  - PUT /OS-FEDERATION/identity_providers/idpopen/protocols/saml2
  - PUT /OS-FEDERATION/identity_providers/adfs/protocols/saml2

```
{
    "id": "saml2",
    "mapping_id": "testshib_mapping"
}

{
    "id": "saml2",
    "mapping_id": "idpopen_mapping"
}

{
```

```
    "id": "saml2",
    "mapping_id": "adfs_mapping"
}
```

Note:

- for each protocol is necessary to specify the mapping identifier previously created.

4.5 How the Testbed works (via ECP)

The user who wants to authenticate and receive the unscoped token from Keystone Identity Service needs to specify the following HTTP request:

- GET/POST /OS-FEDERATION/identity_providers/<identity_provider_id>/protocols/<protocol_id>/auth

The received token contains information about the groups to which the federated user is mapped.

We need to distinguish two cases:

- Shibboleth Identity Provider;
- ADFS Identity Provider.

In case of Shibboleth Identity Provider to perform the login I used the ecp.py python script (https://wiki.shibboleth.net/confluence/download/attachments/4358416/ecp.py?api=v2) with the following IDP_ENDPOINTS:

```
# mapping from user friendly names or tags to IdP ECP enpoints

IDP_ENDPOINTS = {
 "testshib" : "https://idp.testshib.org/idp/profile/SAML2/SOAP/ECP",
 "idpopen" : "https://idpopen.garr.it/idp/profile/SAML2/SOAP/ECP"
}
```

The endpoints for ECP can be easily retrieved from the metadata file of each Identity Provider.

For instance, if we want to login with the IDPOPEN Identity Provider, we need to run the following command:

```
python ecp.py -d idpopen https://your.server.name:5000/v3/OS-FEDERATION/identity_providers/idpopen/protocols/saml2/auth <username>
```

After entering the password the output will be the following:

```
header: Date: Wed, 27 Aug 2014 08:58:55 GMT
header: Server: Apache/2.2.15 (Red Hat)
header: X-Subject-Token: <unscoped_token>
header: Vary: X-Auth-Token,Accept-Encoding
header: Content-Length: 338
header: Connection: close
header: Content-Type: application/json
```

```
{"token": {"issued_at": "2014-08-27T08:58:56.821301Z", "extras": {}, "methods": ["saml2"], "expires_at": "2014-08-27T09:58:56.821251Z", "user": {"OS-FEDERATION": {"identity_provider": {"id": "idpopen"}, "protocol": {"id": "saml2"}, "groups": [{"id": "<idpopen_group_id>"}]}, "id": "idpopen%20user", "name": "idpopen user"}}}
```

Where the 'X-Subject-Token' represents the unscoped token.

In case of ADFS Identity Provider to perform the login I used a specified ADFS client (https://github.com/zaccone/pyadfsclient):

```
./main.py --sp_url "https://your.server.name:5000/v3/OS-FEDERATION/identity_providers/adfs/protocols/saml2/auth" --user=<username> --password=<pasword> --no-ssl --content
```

With the following uscoped token:

```
{'content-length': '232', 'content-encoding': 'gzip', 'x-subject-token': <unscoped_token>, 'vary': 'X-Auth-Token,Accept-Encoding', 'server': 'Apache/2.2.15 (Red Hat)', 'connection': 'close', 'date': 'Wed, 27 Aug 2014 09:05:34 GMT', 'content-type': 'application/json'}
{"token": {"issued_at": "2014-08-27T09:05:35.530371Z", "extras": {}, "methods": ["saml2"], "expires_at": "2014-08-27T10:05:35.530335Z", "user": {"OS-FEDERATION": {"identity_provider": {"id": "adfs"}, "protocol": {"id": "saml2"}, "groups": [{"id": "<adfs_group_id>"}]}, "id": "adfs%20user", "name": "adfs user"}}}
(adfs-client)[root@ltartari2 pyadfsclient]# ./main.py --user=<user> --password=<password> --no-ssl --content{'content-length': '2076', 'content-encoding': 'gzip', 'vary': 'Accept-Encoding', 'server': 'Apache/2.2.15 (Red Hat)', 'connection': 'close', 'date': 'Wed, 27 Aug 2014 09:05:48 GMT', 'content-type': 'text/plain'}
```

For both Identity Providers, if we want to see the environment variables produced from the SAML assertion we need to specify the following URL:

- https://your.server.name:5000/secure

instead of the URL for the Keystone authentication and copy the wsgi script (https://gist.github.com/zaccone/021203cab26c9e4b0baf) in the /home/www/wsgi-scripts/ directory (as previously configured in the wsgi-keystone.conf file) and use the ECP client (ecp.py for Shibboleth Identity Provider or pyadfsclient for ADFS Identity Provider).

For instance, for the ADFS Identity Provider, the output will be the following:

```
ADFS_AUTHLEVEL: Normal
ADFS_BUILDING: <building>
ADFS_DEPARTMENT: <department>
ADFS_EMAIL: <email>
ADFS_FEDERATION: <federation>
ADFS_FIRSTNAME: <firstname>
ADFS_FULLNAME: <fullname>
ADFS_GROUP:  <group>
ADFS_IDENTITYCLASS: <identity_class>
ADFS_LASTNAME: <latname>
ADFS_LOGIN: <login>
ADFS_PERSONID: <person_id>
ADFS_PREFERREDLANGUAGE: EN
ADFS_ROLE: <role>
```

```
AUTH_TYPE: shibboleth
DOCUMENT_ROOT: /etc/httpd/htdocs
GATEWAY_INTERFACE: CGI/1.1
HTTPS: 1
HTTP_ACCEPT: */*
HTTP_ACCEPT_ENCODING: gzip, deflate
HTTP_COOKIE:
_shibsession_64656661756c7468747470733a2f2f6c74617274617269322e6365726e2e63683a353030302f736869626
26f6c657468=_02d0b3bbc1b033e3499623fdcd31e79f
HTTP_HOST: ltartari2.cern.ch:5000
HTTP_USER_AGENT: python-requests/2.3.0 CPython/2.6.6 Linux/2.6.32-431.20.3.el6.x86_64
PATH_INFO:
QUERY_STRING:
REMOTE_ADDR: 128.142.141.19
REMOTE_PORT: 56484
REQUEST_METHOD: GET
REQUEST_URI: /secure
SCRIPT_FILENAME: /home/www/wsgi-scripts/wsgi_app.py
SCRIPT_NAME: /secure
SERVER_ADDR: 128.142.141.19
SERVER_ADMIN: [no address given]
SERVER_NAME: ltartari2.cern.ch
SERVER_PORT: 5000
SERVER_PROTOCOL: HTTP/1.1
SERVER_SIGNATURE: <address>Apache/2.2.15 (Red Hat) Server at ltartari2.cern.ch Port 5000</address>

SERVER_SOFTWARE: Apache/2.2.15 (Red Hat)
Shib-Application-ID: default
Shib-Authentication-Instant: 2014-08-27T09:05:50.350Z
Shib-Authentication-Method: urn:oasis:names:tc:SAML:1.0:am:password
Shib-AuthnContext-Class: urn:oasis:names:tc:SAML:1.0:am:password
Shib-Identity-Provider: https://cern.ch/login
Shib-Session-ID: _02d0b3bbc1b033e3499623fdcd31e79f
mod_wsgi.application_group:
mod_wsgi.callable_object: application
mod_wsgi.handler_script:
mod_wsgi.input_chunked: 0
mod_wsgi.listener_host:
mod_wsgi.listener_port: 5000
mod_wsgi.process_group: keystone-public
mod_wsgi.request_handler: wsgi-script
mod_wsgi.script_reloading: 1
mod_wsgi.version: (3, 2)
user: <user>
wsgi.errors: <mod_wsgi.Log object at 0x7f69aa2e3cf0>
wsgi.file_wrapper: <built-in method file_wrapper of mod_wsgi.Adapter object at 0x7f69a9d41198>
wsgi.input: <mod_wsgi.Input object at 0x7f69a9a60af0>
wsgi.multiprocess: True
wsgi.multithread: True
wsgi.run_once: False
wsgi.url_scheme: https
wsgi.version: (1, 1)
```

The environment variables are the attribute specified in attribute-map.xml file in the /etc/shibboleth/ directory.

4.6 Multiple Shibboleth IdPs via Browser (Discovery Service)

Unlike ECP, a Service Provider that use the Shibboleth Embedded Discovery Service (a set of Javascript and CSS files) allows users to select the Identity Provider from a web UI at the moment of the authentication.

For the installation and configuration of Shibboleth EDS on the host we need to perform the following steps:

- yum install shibboleth-embedded-ds
    - or download the .rpm (http://download.opensuse.org/repositories/security://shibboleth/RHEL_6/noarch/) and install it:
        - rpm -i <filename>.rpm
    - after the installation will be created a new directory: /etc/shibboleth-ds/ in which are stored the Javascript and CSS files.
- Edit the /etc/shibboleth/shibboleth2.xml file:
    - in each <MetadataProvider> element add:
        - 'legacyOrgNames=”true”

```
<MetadataProvider type="XML" uri="http://www.testshib.org/metadata/testshib-providers.xml" backingFilePath="/etc/shibboleth/metadata-testshib.xml" reloadInterval="180000"/>

<MetadataProvider type="XML" uri="https://idpopen.garr.it/metadata/idp-metadata.xml" backingFilePath="/etc/shibboleth/metadata-idpopen.xml" reloadInterval="180000"/>
```

    - in the <SSO> element add:
        - discoveryProtocol="SAMLDS"
        - discoveryURL="https://your.server.name:5000/shibboleth-ds/index.html"
    - and remove:
        - ECP=”true”

```
<SSO discoveryProtocol="SAMLDS" discoveryURL="https://your.server.name:5000/shibboleth-ds/index.html">SAML2
SAML1</SSO>
```

Now you can open a web browser and specify the URL:

- https://your.server.name:5000/v3/OS-FEDERATION/identity_providers/<identity_provider_id>/protocols/saml2/auth

depending on which Identity Provider you want to use. The web UI is shown in Figure 11 below:

Figure 11. Shibboleth EDS UI

After confirming the Identity Provider choice you are redirected the the Identity Provider's login page:

Figure 12. Testshib login page

Once logged in the user will see the unscoped token:


```
▼<token xmlns="http://docs.openstack.org/identity/api/v3" issued_at="2014-08-27T12:26:49.278876Z" expires_at="2014-08-27T13:26:49.278826Z">
  <extras/>
  ▼<methods>
    <method>saml2</method>
  </methods>
  ▼<user id="testhib%20user" name="testhib user">
    ▼<OS-FEDERATION>
      <identity_provider id="testshib"/>
      <protocol id="saml2"/>
      ▼<groups>
        <group id="5e845737a3d646e6b32c173ef5a3cbe4"/>
      </groups>
    </OS-FEDERATION>
  </user>
</token>
```


Figure 13. Unscoped token via browser

Note:

- in this case Shibboleth Embedded Discovery Service is working only with Shibboleth Identity Providers (not ADFS)

5 Keystone-client wrapper

A federated user that has received an uscoped token can have access to a resource (project or domain):

- List the projects accessible to federated user:
  - GET /OS-FEDERATION/projects
- List the domains accessible to federate user:
  - GET /OS-FEDERATION/domains

For instance, the IDPOPEN user can have access to the following projects:

```
{
        "description": "Project for idpopen",
        "domain_id": "<fed_domain_id>",
        "enabled": true,
        "id": "<idpopen_project_id>",
},
{
        "description": "Project for federation test",
        "domain_id": "<fed_domain_id>",
        "enabled": true,
        "id": "<fed_project_id>",
        "name": "federation-project"
}
```

In order to scope the unscoped token received from Keystone Identity Service I used a keystone-client wrapper (https://gist.github.com/zaccone/509136cfa1c4efca6926).

Using the <fed_domain_id> as a parameter in the python script:

```
scopeTokenplugin = saml2.Saml2ScopedToken(
   SERVICE_PROVIDER_URL, token.auth_token, domain_id=VALID_DOMAIN_ID)
```

we are able to scope the unscoped token:

```
{
   "auth_token": <scoped_token>,
   "catalog": [
      {
         "endpoints": [
            {
               "id": "5b99593a5a5b435eaf550a3c14d64796",
               "interface": "internal",
               "region": "RegionOne",
```

```
                "url": "http://128.142.141.19:8776/v1/None"
            },
            ...
        ],
        "id": "fc0daffc9c034ebd841d31a696ba70c1",
        "type": "ec2"
    }
  ],
  "domain": {
    "id": "default",
    "name": "Default"
  },
  "expires_at": "2014-08-27T14:00:47.270712Z",
  "extras": {},
  "issued_at": "2014-08-27T13:00:47.270750Z",
  "methods": [
    "saml2"
  ],
  "roles": [
    {
      "id": "<idpopen_role_id>",
      "name": "idpopen-role"
    }
  ],
  "user": {
    "id": "idpopen%20user",
    "name": "idpopen user"
  },
  "version": "v3"
}
```


Issue:

- can not scope the token using the project identifier.

6 Federate Ipsilon Identity Provider on FreeIPA

In this chapter I will describe the main configurations steps for the installation of Ipsilon Identity Provider (https://fedorahosted.org/ipsilon/) on FreeIPA (both from Red Hat). Ipsilon implements an Identity Provider that exposes SAML while FreeIPA is a project which aims to provide an easily managed Identity, Policy and Audit (IPA).

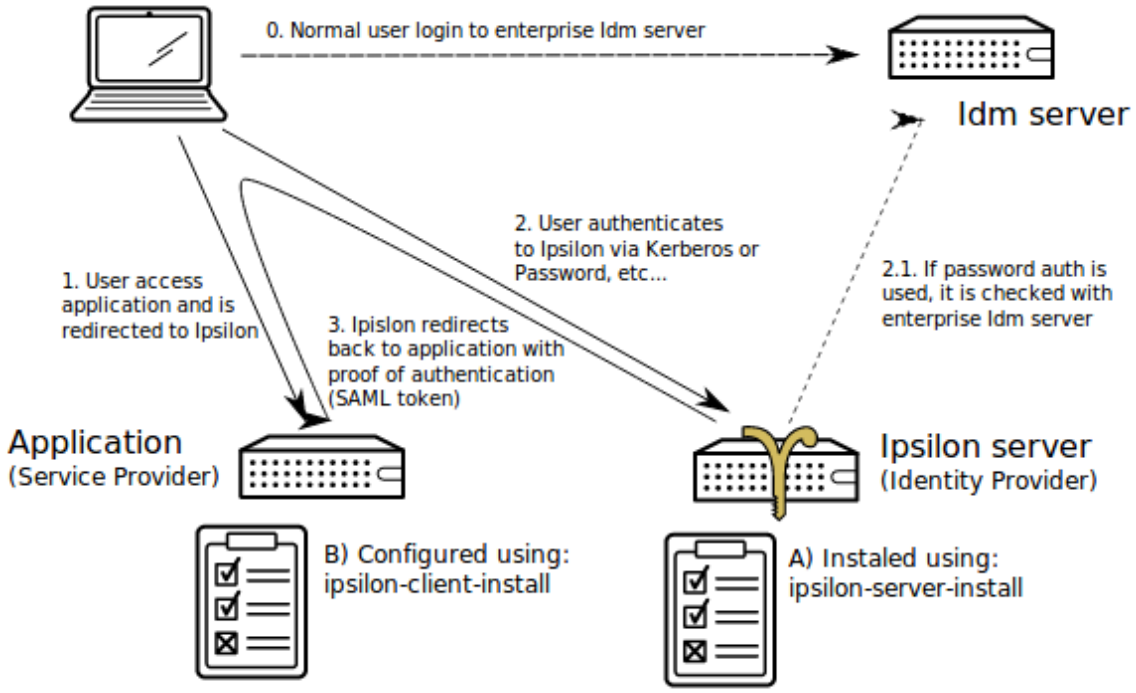

Figure 14. Ipsilon architecture

In the VM in which Keystone (Service Provider) is running we are going to install and configure:

- Ipsilon-client;

in a new VM we are going to install and configure:

- FreeIPA,
- Ipsilon-server.

6.1 Installing FreeIPA and Ipsilon-server

The main configuration steps are:

- yum install ipa-server

- ipa-server-install
- kinit admin
- git clone https://git.fedorahosted.org/cgit/ipsilon.git/
- install lasso:
  - git clone https://repos.entrouvert.org/lasso.git
  - cd lasso
  - edit: lasso/lasso.c
    - line 178: if (xmlSecCryptoDLLoadLibrary(BAD_CAST "openssl") < 0) {
  - edit: lasso/doc/reference/lasso/Makefile.am
    - line 98: EXTRA_DIST = lasso-sections.txt lasso-docs.sgml version.xml.in lasso.types.in style.css
  - ./autogen.sh --prefix=/usr/lib/python2.6/site-packages/
  - make install
- easy_install cherrypy (version 3.2.2)
- easy_install Jinja2
- for SSL certificate edit: /usr/lib/python2.6/site-packages/requests/api.py
  - line 55: kwargs.setdefault('verify', False)
- for python2.6 edit: /usr/lib/python2.6/site-packages/ipsilon/admin/login.py
  - line48: plugins_by_name = dict((p.name, p) for p in self._site[FACILITY]['enabled'])
- build ipsilon-server:
  - cd ipsilon
  - python setup.py build
  - python setup.py install
- ipsilon-server-install –ipa=yes

6.2 Installing Ipsilon-client

- Install lasso (see Chapter 6.1)

- ipsilon-client-install –saml-idp-metadata https://your.idp.name/idp/saml2/metadata

After this command the ipsilon-client will generate:

- idp-metadata.xml: Identity Provider's metadata file;
- metadata.xml: Service Provider's metadata file.

Then it is necessary to upload the metadata.xml file to the Identity Provider (Ipsilon-server).

Issue:

- can not login as administrator to Ipsilon-server.

7 Conclusions

In this report is described the work that has been done during the two-months stay as an Openlab Summer Student at CERN.

At the beginning, in the early chapters, the background concepts are explained to get a better understanding for the following sections in which is described in detail and step by step the development of the project.

The main goal was to create a testbed for performing federation tests with different Identity Providers at the same time. Testshib IdP, IDPOPEN and ADFS have been installed successfully and tested with both web browser and CLI. Unfortunately Ipsilon IdP has not been tested because of some issues with authentication in the Ipsilon-server.

In order to have different Identity Providers at the same time via ECP we focused primarily on the Shibboleth configuration files to achieve the goal. Several emails were exchange with the open-source mailing lists to figure out and configure the various products and tools used during the project.

The testbed has been developed on OpenStack Virtual Machines in which was installed the Service Provider with Keystone, while the Identity Providers are: Testshib (an Identity Provider provided by Shibboleth to perform tests), IDPOPEN (an Identity Provider used by the INFN of the University of Catania) and ADFS (used by CERN).

Subsequently was used the Shibboleth Embedded Discovery Service in order to handle multiple Identity Providers via browser enabling the Discovery Service. This solution is only valid for the Identity Providers provided by Shibboleth.

At the end we tried to scope the unscoped token received from Keystone Identity Service using a wrapper for keystone-client and use the scoped token to perform some OpenStack commands.