Journal of Physics: Conference Series **664** (2015) 022015 doi:10.1088/1742-6596/664/2/022015

Identity federation in OpenStack - an introduction to hybrid clouds

Marek Denis, Jose Castro Leon, Emmanuel Ormancey, Paolo Tedesco
CERN, Geneva, Switzerland

E-mail: {marek.denis, jose.castro.leon, emmanuel.ormancey, paolo.tedesco}@cern.ch

Abstract. We are evaluating cloud identity federation available in the OpenStack ecosystem that allows for on premise bursting into remote clouds with use of local identities (i.e. domain accounts). Further enhancements to identity federation are a clear way to hybrid cloud architectures - virtualized infrastructures layered across independent private and public clouds.

1. The Introduction

The National Institute of Standards and Technology [1] defines cloud computing [2] as:

Definition 1 (...) *model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

Over the past few years cloud computing has been gaining huge momentum. Business model, [1] where users pay only for the resources they used, allows for more flexible and cost effective managing of computing infrastructure. Instead of building and maintaining expensive data centers, many vendors and organizations prefer to utilize a cloud provider's infrastructure and pay only for the resources they use. At the same time, organizations that maintain their own data centers often choose private cloud solutions as this helps them to provide better services to their end users. Infrastructures relying on cloud computing models usually result in better hardware utilization (multiple virtual machines can run within one bare-metal hypervisor), better resource isolation as well as faster and fully automatized service orchestration. Users can provision new virtual machines by placing a request in the cloud scheduler, instead of issuing a ticket in the ticketing system and waiting for administrators to prepare a bare-metal machine. Automatic provisioning of virtualized resources can usually be finished within minutes. What is more, requesting many virtual machines takes similar amount of time as cloud schedulers distribute requests across multiple hypervisors and the requests are processed in paralell. *OpenStack*[3] is the most popular open source platform for managing and maintaining cloud infrastructures. It is actively developed by wide community backed by developers working in both academic and commercial areas.

[1] often referenced as "pay as you go" model

1.1. Cloud computing models

There are three main deployments models outlined by NIST. These are:

Private cloud - the cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

Community cloud - the cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Public cloud - the cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Hybrid cloud - the cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

The *hybrid* model provides greater flexibility than its *private* or *public* counterparts. It can help building even more cost effective architectures and provide seamless multi-cloud experience. The model introduces another software layer that spans across multiple independent, separately governed cloud deployments. Hybrid architectures allows distinct sites to create collaborations with an easy access to the shared resources. It also allows to build deployments with different *Service Level Agreement (SLA)* requirements - resources with higher priorities are provisioned in the cloud deployments providing better latency, bandwidth and more reliable service, whereas less prioritized resources are located in clouds offering less strict requirements. OpenStack and its identity federation implementation - OS-FEDERATION - are the first attempts to provide mechanisms for deploying hybrid infrastructures.

2. OpenStack - Infrastructure as a Service

OpenStack is a free and open source cloud computing software platform. Its primary goal is to provide service in the Infrastructure-as-a-Service (IaaS) model. OpenStack consists of interrelated projects that control pools of processing, storage and networking resources through a data center. Users interact with the services via RESTful APIs which help in adding automated clients operating on text protocols. For better user experience one of the subprojects provides web access built on top of OpenStack APIs. Typical OpenStack installation consists of the following components:

Identity Service (keystone) - provisions users, groups, roles, projects, domains,

Compute (nova) - schedules Virtual Machines, manages their lifecycle,

Image Service (glance) - provides data assets (Virtual Machine images, Heat templates),

Networking Service (neutron) - provides software defined networks,

Object Storage Service (swift) - provides scalable and persistent object store,

Block Storage Service (cinder) - provides persistent block-level storage,

Orchestration Service (heat) - orchestrator with autoscaling functionalities,

Metering Service (ceilometer) - collects metering probes from other services,

Dashboard (horizon) - provides web based interface for other services.

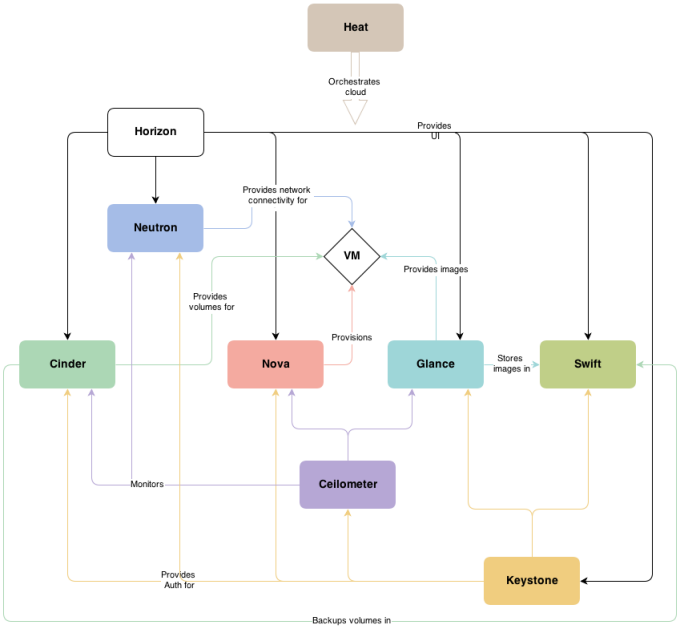

Figure 1. OpenStack Architecture

3. OpenStack keystone

Keystone is a service responsible for service discovery, provisioning users, groups, roles, projects, domains as well as user authentication and authorization. OpenStack implements Role Based Access Control (RBAC). Users and groups are granted roles (member, admin) which had also been previously assigned to projects or domains.

In a standard configuration users must be provisioned in the Identity Service (keystone). Internally, all persistent metadata is locally stored in the database. Whenever an OpenStack user is added, modified or deleted, an administrator must request a change via the Identity API. Authentication and authorization in OpenStack rely on *tokens*[2]. Whenever a user interacts with a service in the OpenStack deployment, he is required to possess the token as an instance of his identity and assigned roles. User may ask for unscoped token, however it will not be usable in a typical daily operations like creating a virtual machine or uploading a virtual machine image. When the user requests a scoped token (or scopes unscoped token) keystone checks whether user's role assignments allow for accessing requested project or domain. Tokens are structured as JSON objects and include parameters like: *user identification (id, name), authentication method, expiration date, service catalog, and project/domain to which the token is scoped.* Keystone is also a major actor in the federated authentication and authorization workflow in OpenStack.

4. Identity federation

Identity federation is a secure and easy way to split authentication and authorization. A centralized service, *identity provider*, is responsible for authenticating user, whereas multiple trusted *service providers* are providing services, authorizing (already authenticated) users to access resources, assets or request actions. Gartner [4] depicts identity federation in the following way:

[2] OpenStack Kilo release provides 3 types of tokens: *UUID*, *PKI* and *Fernet*, however the differences are beyond the scope of this paper.

Definition 2 *Federated identity management enables identity information to be developed and shared among several entities and across trust domains. Tools and standards permit identity attributes to be transferred from one trusted identifying and authenticating entity to another for authentication, authorization and other purposes, thus providing single sign-on convenience and efficiencies to identified individuals, identity providers and relying parties.*

where main actors in the architecture are:

Service Provider - a system entity providing services to other system entities.

Identity Provider - a provider that creates, maintains and manages identity information for principals and provides authentication to other service providers within a federation.

Identity federation allows for designing and implementing distributed architectures where user identities are stored centrally in well known endpoints (identity providers). Trusted services (service providers) aim to provide service without reimplementing authentication mechanisms and having to handle security related risks. With identity federation enabled, service administrators can delegate authentication responsibilities to identity providers. The gain is easily visible for both parties - end users as well as staff responsible for maintaining services. Identity federation used in cloud computing is also extremely profitable for quick, on premise bursting into remote clouds. Instead of creating, maintaining and later deleting accounts at the remote site, two peers can configure a federated trust and a fraction of users will be able to immediately access those remote resources with their local accounts.

4.1. Federation Protocols

Identity federation requires a secure and reliable transport layer used to exchange information between trusted peers. There are few established protocols defining format of the data, as well as order of the commands and interactions between the peers. The most widely used federated protocols are:

SAML - Security Assertion Markup Language[5] is a format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider. It is open-standard format, based on XML. SAML was created by OASIS. Specifications recommend using TLS for transport layer security and XML signing and encrypting for message level. SAML provides various workflows (called profiles), out of which the *Web Single-Sign-On (WebSSO)* is the most popular. Pure HTTP clients must rely on *Enhanced Client or Proxy (ECP)* [7] profile which wraps XML data format with SOAP headers.

OpenID Connect - an authentication standard[6] based on oAuth2.0 [8]. OpenID Connect uses JSON for data format and specifies a RESTful API. The protocol specification is maintained and developed by the *OpenID Foundation*.

An example of standard federated workflow (with SAML2 protocol as a transport layer) is presented in figure 2:

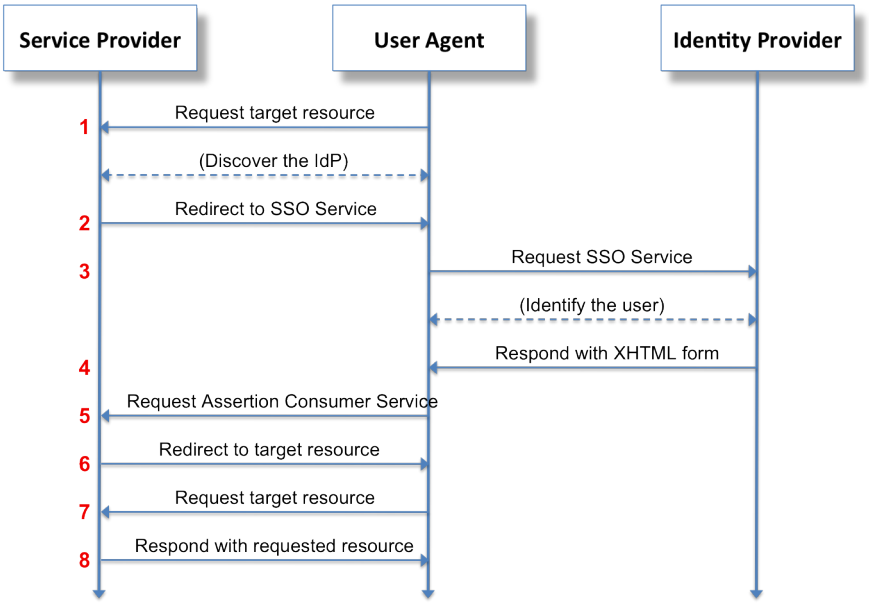

Figure 2. Federated authentication workflow

5. OS-FEDERATION in OpenStack

Identity federation[3] is supported since OpenStack Icehouse (released in April 2014). In this setup keystone acts as a service provider, issuing federated tokens, later used like standard OpenStack tokens. Federated tokens need to store information about the identity provider that authenticated user as well as federated protocol that was used in the information exchange process between trusted peers. Prior to successful federated authentication a number of configuration steps must be executed. Adding following entities via the Identity API is one of them:

- Identity provider - trusted identity provider (or identity providers),
- Protocol - federation protocol used,
- Mapping - set of rules used for translating assertions into local parameters (groups, user names)

Both *Identity Provider* and *Protocol* entities are identified by unique user provided identifiers. User who wishes to authenticate with use of OS-FEDERATION indicates identity provider and protocol of his choice by specifying them in the authentication URL - *https://identity/v3/OS-FEDERATION/identity_providers/{idp_name}/protocols/{protocol}/auth*. A simplified database diagram with relations between the entities is presented in figure 3.

[3] OpenStack implementation is named OS-FEDERATION

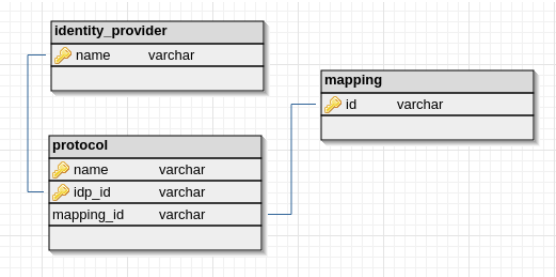

Figure 3. OS-FEDERATION simplified database schema

5.1. Mapping Engine

One of the main design decisions in identity federation in OpenStack is that users authenticating with the federated workflow are **ephemeral**. That is, the user record does not exist in keystone's database and no such record is created once the user has authenticated. With such decision, keystone must be able to assign roles dynamically and for that purpose a *Mapping Engine* was designed and implemented in keystone. Its aim is to translate assertion parameters into set of OpenStack specific parameters - local groups and user id. Each mapping rule is defined per {identity_provider, protocol} tuple. Basing on external parameters delivered in the assertion, the engine assigns user_id and local keystone groups membership. There may be multiple rules (stored in one ruleset) defined per {identity_provider, protocol} tuple. Each rule consists of two major entries: *local* and *remote* where former is responsible for evaluating input issued by the trusted identity provider and the latter for generating input for ephemeral user identity. Rules are additive - effective ephemeral identity is an union of all subsequent, positively evaluated rules. An example of mapping rules is presented in the table 1

```
{
"local": [
  {
    "user": { "name": "{0}" }
  },
  {
        "group": {"id": "0cd5e9"}
  },
  {
        "group": {
      "name": "IT-staff",
      "domain": { "name": "CERN" }
    }
  }
],
}
```

```
{
  "remote": [
    {
      "type":"DEPARTMENT",
        "any_one_of": ["IT"]
      },
      {
        "type": "GROUPS"
        "whitelist": ["IT/OIS", "MANAGEMENT"]
      }
  ]
}
```

Table 1. Mapping rules

5.1.1. local - an entity that defines local attributes - user_id of the ephemeral user and list of effective local keystone groups that the user will become a member of. Groups must be added to keystone configuration prior to federated authentication. Being a member of a group implies inheriting role assignments tied to a group. Rules allow for direct transfer of assertions' parameters into local entities (i.e. an e-mail address from the assertion may be used as a user_id),

transferring black/whitelisted parameters and evaluating parameters with a boolean result (if assertion's *parameter_1* equals "A" and *parameter_2* doesn't equal "B" then assign group "G")

How ephemeral users are identified depends on actual rule. It is recommended to make an effort to ensure proper user identification. Federated tokens always carry identifier of issuing identity provider, as well as federated protocol , so user identifiers can be grouped by {identity provider, protocol} tuple. However, many services depend only on user_id attribute so it is advised to build rules making them unique within a system deployment.

5.1.2. remote - an entity defining rules that must pass assertion evaluation. All the subsequent requirements must be positively validated and only then ephemeral user will be entitled for the corresponding *local* rule.

Following keywords iare available for a *remote* rules specification:

- **any_one_of** - validated parameter must match one of the specified values. Result is either true or false,
- **not_any_of** - validated parameter must not match any value specified in the rule. Result is either true or false,
- **whitelist** - used for positive filtering parameter where rules can be used in direct mappings,
- **blacklist** - used for negative parameter filtering where parameters can be used in direct mappings

Regular expressions can be used for parameter evaluation, however a *regex=True* directive must be added in the rule's section.

6. Deployment and configuration

In order to setup OpenStack deployment capable of federated authentication deployers must install keystone server on top of the Apache HTTPD webserver and add modules for proper federated protocol handling. The keystone team has tested and recommend mod_shib module for SAML deployments as well as mod_auth_oidc for OpenID Connect.

Since the OpenStack Icehouse release, OS-FEDERATION is supported with use of SAML protocol enhanced with ECP. WebSSO support has been added in OpenStack Kilo release (April 2015), both in keystone and horizon.

7. OS-FEDERATION auditing

Traceability and auditing are important aspects of running every service. Whether it is for the billing purposes or post-mortem analysis, administrators have to be able to trace all actions of all their users. For that purpose OpenStack utilizes *Cloud Auditing Data Federation (CADF)* [9] - a standard that enables cross-vendor information sharing via its data format and interface definitions. CADF has been developed by Distributed Management Task Force (DMTF) and fully adopted by OpenStack keystone [10] [11]. Each action requested by a user triggers a broadcast of a *CADF event* which is later injected into the message bus. Such events can be consumed, parsed and stored by other services, for instance *OpenStack ceilometer*. CADF events are also broadcasted for ephemeral users' actions, making them fully traceable and accountable.

8. Conclusions

Cloud identity federation is a suitable way to share identities between many, independent cloud deployments. It can be used for further optimizations of resource usage (by using remote cloud resources when local are fully saturated) as well as quick configuration of shared infrastructures used for collaborative work.

Using open and well tested standards in OpenStack ensures that only the best and most secure practices are bought along. Cloud identity federation is also a first step towards truly hybrid clouds.

References

[1] - http://www.nist.gov/
[2] - http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[3] - http://www.openstack.org
[4] - http://www.gartner.com/it-glossary/federated-identity-management
[5] - http://saml.xml.org/saml-specifications
[6] - http://openid.net/connect/
[7] - https://wiki.oasis-open.org/security/SAML2EnhancedClientProfile
[8] - http://oauth.net/2/
[9] - http://www.dmtf.org/standards/cadf
[10] - https://wiki.openstack.org/wiki/ReleaseNotes/Icehouse#Key_New_Features_5
[11] - https://wiki.openstack.org/wiki/ReleaseNotes/Juno#Key_New_Features_4