ÉCOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTÈMES

PROJET FEDERATEUR

Interopérabilité des méthodes de contrôle d'accès dans Openstack

---

Soutenu par :

GUEZRI AMINE ***BESRI MOHAMED AIMANE***

Encadré par : Professeur D.BOUZIDI

Examiné par : Professeur R.ROMADI

Remerciement

On tient à remercier particulièrement Dr. BOUZIDI pour la confiance qu'il nous accordés, pour son aide et ses précieux conseils apportés durant la réalisation de notre projet.

Nos remerciements également Dr. Romadi pour avoir accepté de juger notre travaille.

Résumé

Le contrôle d'accès basé sur les rôles (RBAC) est le modèle de contrôle d'accès dominant dans l'industrie depuis les années 1990. Il est largement implémenté dans de nombreuses applications, y compris les principales plates-formes cloud telles que OpenStack, AWS et Microsoft Azure.

Toutefois, en raison des limitations de RBAC, les modèles de contrôle d'accès par attributs (ABAC) évoluent pour améliorer la flexibilité en utilisant des attributs autres que les rôles et les groupes. En pratique, ce changement doit être progressif car irréaliste pour les systèmes existants. systèmes à adopter brusquement les modèles ABAC, éliminant ainsi complètement les implémentations RBAC actuelles.

Dans ce rapport, nous proposons une extension ABAC avec des attributs d'utilisateur pour le modèle OpenStack Access Control (OSAC) et démontrons son application à l'aide de la machine de politique (PM) développée par l'Institut national de la normalisation et de la technologie. Nous utilisons certains des composants de PM et une implémentation de validation de principe pour appliquer cette extension ABAC à OpenStack, tout en maintenant l'architecture RBAC actuelle d'OpenStack en place. Cela offre les avantages d'améliorer la flexibilité du contrôle d'accès grâce à la prise en charge des attributs utilisateur, tout en minimisant la charge de travail liée à la modification de la structure de contrôle d'accès OpenStack existante. Nous présentons des cas d'utilisation décrivant les avantages supplémentaires de notre modèle et montrant les résultats de l'application. Nous évaluons ensuite les performances de notre extension ABAC proposée et discutons de son applicabilité et des améliorations possibles des performances.

Mots-clés : Machine de politique; Contrôle d'accès basé sur les attributs; Openstack; Moteur d'autorisation;

Abstract

Role-Based Access Control (RBAC) has been the dominant access control model in industry since the 1990s. It is widely implemented in many applications, including major cloud platforms such as OpenStack, AWS, and Microsoft Azure.

However, due to limitations of RBAC, there is a shift towards Attribute-Based Access Control (ABAC) models to enhance flexibility by using attributes beyond roles and groups. In practice, this shift has to be gradual since it is unrealistic for existing systems to abruptly adopt ABAC models, completely eliminating current RBAC implementations.

In this paper, we propose an ABAC extension with user attributes for the OpenStack Access Control (OSAC) model and demonstrate its enforcement utilizing the Policy Machine (PM) developed by the National Institute of Standards and Technology. We utilize some of the PM's components along with a proof-of-concept implementation to enforce this ABAC extension for OpenStack, while keeping OpenStack's current RBAC architecture in place. This provides the benefits of enhancing access control flexibility with support of user attributes, while minimizing the overhead of altering the existing OpenStack access control framework. We present use cases to depict added benefits of our model and show enforcement results. We then evaluate the performance of our proposed ABAC extension, and discuss its applicability and possible performance enhancements.

Keywords : Policy Machine; Attribute-Based Access Control; OpenStack; Authorization Engine;

Listes des figures

Table des matières

Introduction générale

Le modèle de contrôle d'accès basé sur le rôle (RBAC) est le modèle de contrôle d'accès le plus utilisé dans l'industrie. De nombreuses applications et plates-formes utilisent une forme de RBAC personnalisée en fonction de leurs besoins et exigences. Les principales plates-formes de cloud computing telles que OpenStack, AWS et Microsoft Azure utilisent RBAC comme fondement des autorisations. Outre de nombreux avantages bien connus de RBAC , il comporte également certaines limitations bien connues, telles que l'explosion de rôles et l'explosion d'autorisations de rôles.

Dans RBAC, les politiques de contrôle d'accès ne peuvent être définies que sur la base de rôles limitant la flexibilité du contrôle d'accès. D'autre part, le contrôle d'accès basé sur les attributs (ABAC) offre une grande flexibilité pour exprimer des politiques de contrôle d'accès détaillées d'une manière simple et plus puissante, en fonction des attributs des utilisateurs, des sujets et des objets. Avec les progrès d'ABAC et de ses capacités, il devient inévitable de mettre en œuvre des modèles ABAC dans des applications et des systèmes du monde réel. Cependant, **il est difficile pour les systèmes existants de s'adapter** instantanément aux stratégies de contrôle d'accès basées sur des attributs car elles nécessitent un attribut bien défini et robuste pour leur mise en œuvre. Nous croyons en la transformation des politiques RBAC en ABAC **doit être progressive** mais nous verrons finalement une large adoption et implémentation des modèles ABAC dans l'industrie. En particulier combiner ABAC avec des rôles est une voie de transition prometteuse.

Ce travail est structuré en quatre chapitres, dont voici une brève présentation:

Dans le premier chapitre, nous présentons le contexte général du projet. Nous allons annoncé la problématique, définir les objectifs de notre projet et décrire la démarche que nous avons suivi pour atteindre ces objectifs.

Dans le deuxième chapitre, nous étudions quelques modèles de contrôle d'accès. Ensuite, nous présentons l'architecture globale d'Openstack .Nous concluons qu'introduire d'ABAC est une nécessité. On va étudier par la suite des solutions pour adopter ABAC comme modèle de contrôle d'accès dans notre Cloud,

voir les méthodes : «Solution Proxies », « Évaluation du risque » et enfin la «Policy Machine ».

Le troisième et dernier chapitre est dédié à la partie réalisation. Nous allons voir l'implémentation de « Policy machine » . Cela nous permet d'étendre RBAC ,existant dans OpenStack, avec des attributs afin de combiner les avantages des deux modèles.

La 'Policy Machine' permet d'exprimer et de faire respecter différents types de politiques de contrôle d'accès via ses points de configuration de politique. Il fournit un cadre unificateur pour définir, administrer et appliquer les politiques de contrôle d'accès communément connues, ainsi que des nouvelles politiques de contrôle d'accès.

Enfin, nous clôturons notre travail par une conclusion générale, et quelques perspectives.

Chapitre 1 :
Contexte général du projet

Introduction

Ce premier chapitre décrit l'environnement dans lequel le projet a été initié : le Cloud. Ensuite, il présente les problèmes de limitations des méthodes de contrôle d'accès dans Openstack (Cloud ici choisi). Enfin, il expose les objectifs pour remédier à ce problème.

Périmètre de projet

En raison de la nature ouverte des Clouds, les utilisateurs souffrent d'une insuffisance au niveau des choix dans les méthodes de contrôle d'accès. Ainsi l'une des premières questions à résoudre pour la résolution de cette problématique, serait d'assurer un maximum de flexibilité au niveau des méthodes de contrôle d'accès.
Dans ces conditions, notre projet portera sur l'implémentation d'une méthode assurant la cohabitation de différents méthodes de contrôle d'accès dans un Cloud.

Problématique

Chaque organisme adopte un mécanisme de contrôle d'accès qui ,selon lui, convient parfaitement à ses besoins. Cependant, les Clouds en général offrent des choix limités de méthodes de contrôle d'accès, mais toutefois en cherchant à élargir la gamme de clientèle. Ceci ne peut être accompli que si l'interopérabilité est atteinte.

D'autre part, la fédération de Clouds est un principe qui consiste à faire collaborer une multitude de fournisseurs de Clouds afin d'optimiser d'avantage leurs ressources et offrir à leurs clients des services diversifiés. Ceci dit, l'interopérabilité des méthodes de contrôle d'accès est primordial pour parvenir à former ce consortium.

Objectif de projet

L'objectif de notre projet est la mise en place d'une machine de politique « PM » qui joue le rôle d'un point de relais entre les utilisateurs et les ressources dans un Cloud. Ceci est également valable pour une multitude de Clouds souhaitant collaborer.

La démarche suivie

Nous avons divisé notre projet en trois grandes parties : Tout d'abord le contexte général du projet, ensuite l'étude des modèles de contrôle d'accès et de l'architecture globale d'Openstack et enfin la partie implémentation. La première partie est consacré principalement à l'analyse et la compréhension du projet. La deuxième partie consiste en l'étude et la compréhension du fonctionnement des modèles de contrôle d'accès. La dernière parie porte sur l'implémentation de la machine des politiques « PM ».

Notre temps a été répartis comme suit pour bien répartir les charges et pouvoir gérer notre temps de manière optimale.

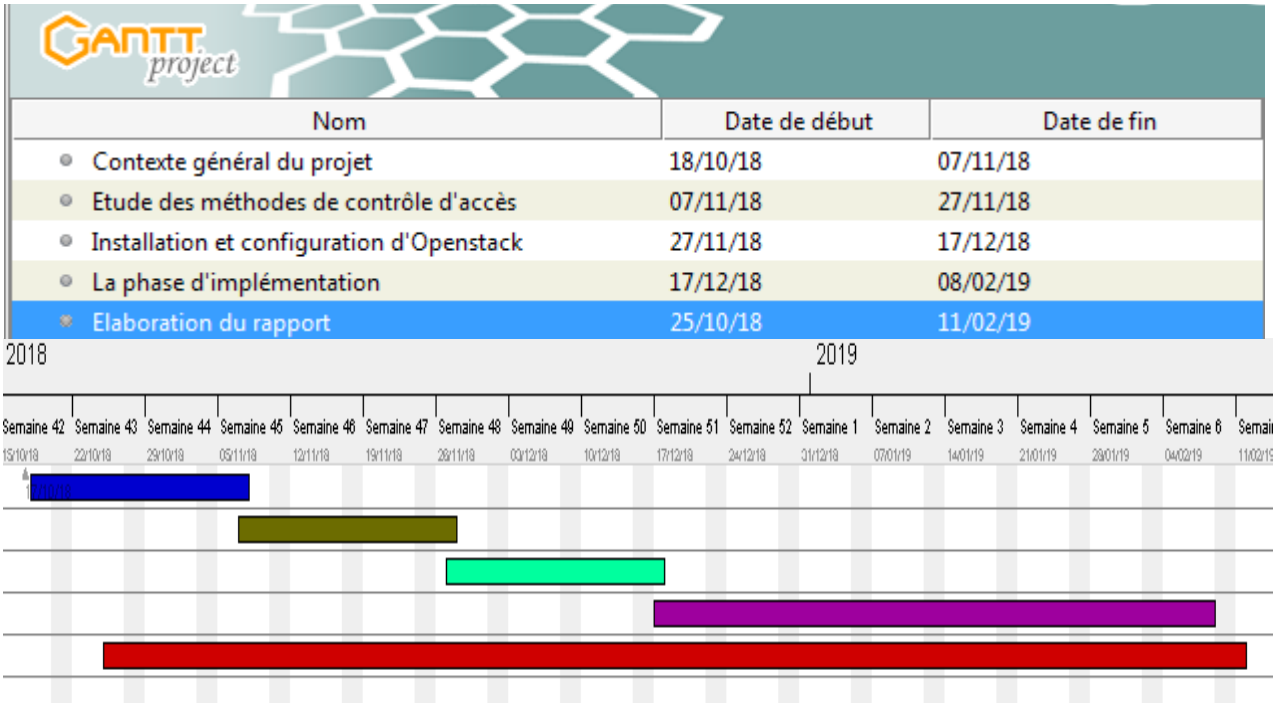

Figure 1 :Diagramme de Grantt

Conclusion

Dans ce chapitre, nous avons introduit le problème de gestion des accès, l'objectif générale de notre projet ainsi que la démarche qu'on a suivi. Dans le chapitre suivant nous allons voir de manière détaillée des méthodes de contrôle d'accès,ainsi que l'architecture globale d'Openstack (Cloud ici choisi).

Chapitre 2 : Étude des modèles de contrôle d'accès dans un Cloud

Introduction

Ce chapitre aborde les concepts fondamentaux pour comprendre les contributions de ce projet. On décrit ABAC, un concept central de ce projet. On présente également OpenStack, AWS Cloud et leurs modèles de contrôle d'accès.

1- Modèles de contrôle d'accès :

Le contrôle d'accès est un mécanisme qui détermine '*qui*' peut faire '*quoi*' dans '*quelle ressource*'. Il est composé de deux parties : L'authentication qui traite avec la partie 'qui', et L'autorisation qui identifie ce que les utilisateurs authentifiés peuvent faire avec les ressources spécifiés. Plusieurs modèles de contrôle d'accès ont été proposés et formalisés dans la littérature. Ces modèles constituent le mécanisme de sécurité fondamental dans de nombreuses applications et systèmes. Parmi ces différents modèles de contrôle d'accès, seuls quelques-uns ont été appliqués avec succès dans des applications et des systèmes du monde réel.

Les trois modèles de contrôle d'accès les plus significatifs et les mieux connus sont le contrôle d'accès discrétionnaire (DAC), le contrôle d'accès obligatoire (MAC) et le contrôle d'accès basé sur les rôles (RBAC). Bien que chacun d'entre eux ait ses avantages, ils ont aussi des faiblesses. Dans DAC, les propriétaires d'objets contrôlent l'accès des utilisateurs aux objets. Il est simple et direct, mais présente des faiblesses inhérentes, telles que *le problème de copie et les chevaux de Troie*, qui peuvent être facilement exploités pour obtenir un accès non autorisé à des informations sensibles. De même, MAC fournit un accès plus strict méthode de contrôle en attribuant des étiquettes aux utilisateurs et à l'objet. Il est conçu pour les applications militaires visant à préserver la confidentialité des informations. Considérant que RBAC est un modèle de contrôle d'accès plus souple et plus convivial sur le plan administratif. Il détermine les accès en fonction des rôles attribués aux utilisateurs et des autorisations associées à ces rôles sur des objets spécifiques. Il s'agit du modèle de contrôle d'accès le plus répandu dans l'industrie. Cependant, il présente également certaines limitations bien connues, telles *que l'explosion de rôle et l'explosion d'autorisation de rôle*.

Motivé par les limitations des modèles de contrôle d'accès traditionnels, le contrôle d'accès par attribut (ABAC) a récemment fait l'objet d'une attention significative dans la littérature. L'idée de base du contrôle d'accès basé sur les attributs est d'utiliser les attributs (caractéristiques ou propriétés) de différentes entités pour prendre des décisions de contrôle d'accès concernant l'accès d'un sujet (utilisateur, processus, etc.) à un objet (fichier, imprimante, base de données, etc.) dans un système. Les décisions de contrôle d'accès sont évaluées en fonction de politiques d'autorisation spécifiées par un administrateur à l'aide d'un langage de spécification de politique. Les politiques d'autorisation ABAC sont un ensemble de règles définies en fonction des attributs des sujets et des objets, ainsi que d'autres attributs, tels que les attributs contextuels.

Le concept d'ABAC existe depuis plus de deux décennies dans la littérature avec des modèles de contrôle d'accès par attribut conçus pour des applications spécifiques et un cryptage par attribut pour le partage sécurisé d'objets ou de données. Cependant, récemment, des universités et des organismes de normalisation tels que l'Institut national des normes et de la technologie (NIST) ont acquis un intérêt pour ABAC en le considérant comme le contrôle d'accès de nouvelle génération (NGAC).

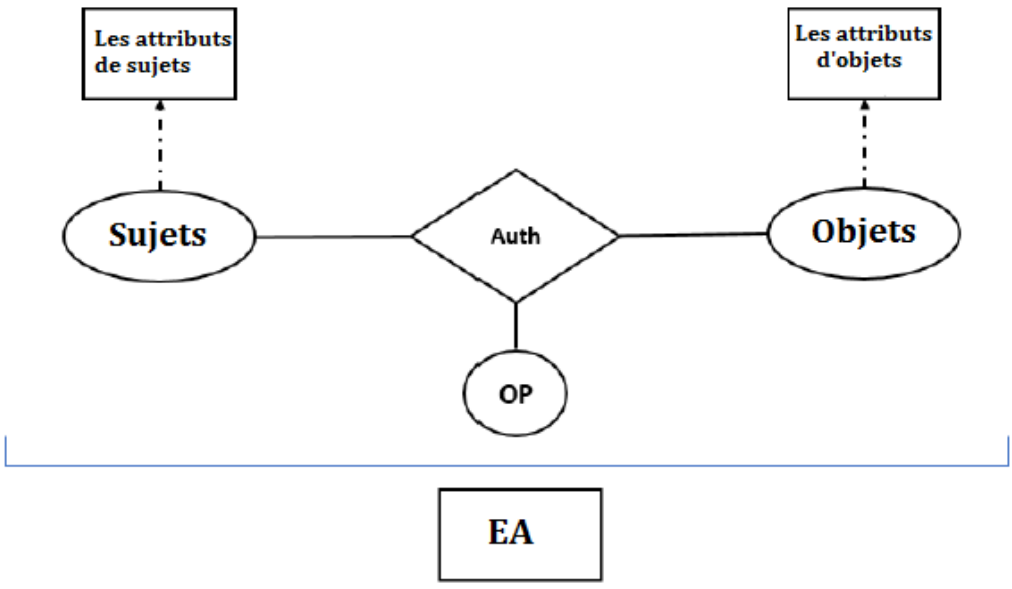

Figure 2 : Modèle conceptuel d'ABAC

La figure ci-dessus présente un modèle ABAC conceptuel simple. Les composants de base du modèle sont les sujets (S), les objets (O), les attributs de sujet (SA), les attributs d'objet (OA), les opérations (OP) et une fonction d'autorisation (Auth). Les attributs contextuels ou environnementaux (EA) existent également dans l'espace modèle et peuvent être utilisés dans la stratégie de contrôle d'accès en fonction des exigences. Par exemple, la demande d'un employé d'accéder aux ressources du bureau n'est accordée que pendant la journée; Donc, ici, daytime correspond à l'attribut d'environnement dans la politique d'autorisation.

Les sujets représentent des utilisateurs dans un système. Ce sont des entités ou des processus créés par; et exécutés pour le compte des utilisateurs. Les objets sont des données, des informations et des ressources protégées, telles qu'une base de données, un fichier, une imprimante, etc ... Les attributs de sujet représentent les propriétés des sujets, telles que Nom, Âge, Titre, etc..., et les attributs d'objet représentent les propriétés des objets, tels que Type, Propriétaire, etc. En général, ces attributs sont des paires nom-valeur. Un attribut est une fonction avec une plage de valeurs qui prend une entité (par exemple, utilisateur, objet) en entrée et renvoie une valeur à partir de cette plage. La plage d'un attribut est un ensemble fini de valeurs atomiques.

Les attributs sont de deux types: « valeur atomique » qui renvoie une valeur unique pour un attribut et « valeur set » qui renvoie un ensemble de valeurs pour un attribut. Les opérations peuvent être simplement 'Lire' et 'Écrire'. La fonction d'autorisation évalue la stratégie d'autorisation et renvoie 'Autoriser' ou 'Refuser' pour une demande d'accès. La nature et les détails concrets de ces composants sont spécifiques à la mise en œuvre et dépendent de la discrétion de l'administrateur système.

Les modèles ABAC peuvent être classés en deux classes en fonction du type de techniques de spécification de la politique d'autorisation, à savoir Stratégie d'autorisation de formule logique (LAP) et EAP (Enumerated Authorization Policy). Dans LAP, la logique de prédicat et les opérateurs logiques (par exemple, AND, OR) sont utilisés pour définir la politique d'autorisation, y compris les attributs et leurs valeurs. Il offre la possibilité de définir des règles d'autorisation complexes de manière simple. Cependant, les stratégies dans LAP sont hétérogènes car il n'y a

aucune contrainte sur la taille et la structure de la politique. Ainsi, il difficile et fastidieux pour un administrateur système de gérer, réviser et mettre à jour les stratégies de formule logique. Cependant dans EAP, la politique d'autorisation est spécifiée comme des n-uplets énumérés impliquant les attributs de sujet et les attributs d'objet. C'est un ensemble d'un ou plusieurs n-uplets dont la structure des politiques est homogène. Par conséquent, EAP fournit une évolutivité administrative pour examiner et mettre à jour les stratégies. Les stratégies EAP peuvent être mises à jour en ajoutant ou en supprimant des n-uplets. Cependant, le PAE présente des inconvénients, tels que la taille importante de la politique en raison de l'énumération de chaque condition de la politique et la prise en charge d'opérateurs limités.
Une instance de EAP-ABAC est la Policy Machine développée par le NIST, qui a été utilisée dans le chapitre suivant pour présenter une architecture d'application unique pour appliquer les modèles ABAC.

Dans la section qui suit, on va introduire la notion d'un Cloud.

2- Le cloud :

Un cloud « nuage » est un ensemble de matériels, de raccordements réseau et de logiciels fournissant des services qu'individus et collectivités peuvent exploiter depuis n'importe où dans le monde. Un basculement de tendance fait qu'au lieu d'obtenir de la puissance de calcul par acquisition de matériel et de logiciel, le consommateur se sert de puissance mise à sa disposition par un fournisseur via l'Internet.

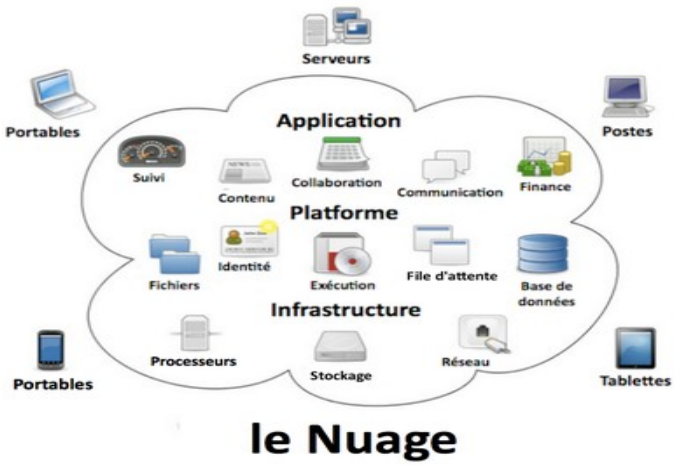

Figure 3 :Vue générale d'un Cloud

On cite parmi les cloud les plus connus : Microsoft Azure, Openstack, Amazon, Dropbox.

2-1- Openstack :

Par la suite, on va travailler avec le cloud Openstack. C'est une plate-forme de cloud computing open source largement utilisée. Il fournit une plate-forme IaaS robuste pour la construction de clouds publics, privés ou hybrides. C'est une application en évolution rapide qui est développée et maintenue par une communauté dynamique de développeurs de plus de 200 organisations de premier plan. Il est formellement défini comme suit :

Le logiciel OpenStack contrôle de vastes pools de ressources de calcul, de stockage et de réseau via un centre de données, géré via un tableau de bord ou via l'API OpenStack. Il fonctionne avec des technologies open source, le rendant idéal pour les infrastructures hétérogènes.

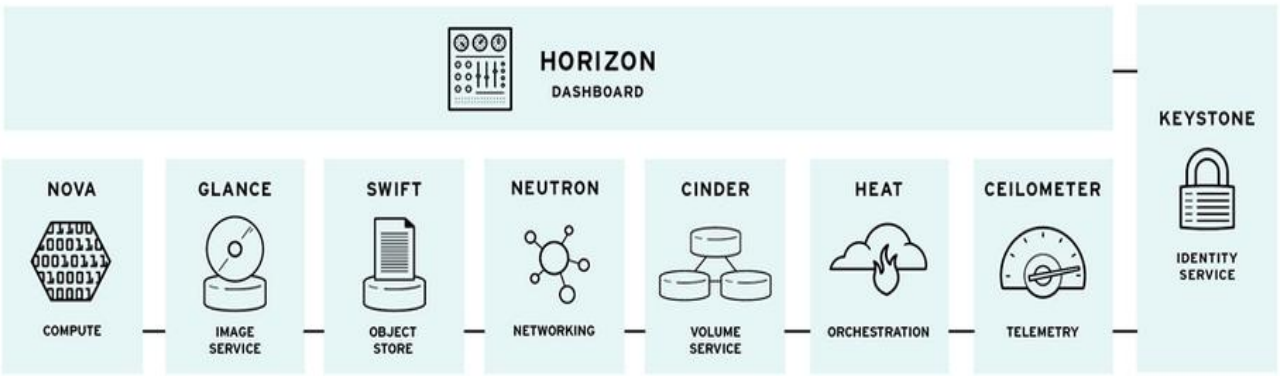

Figure 4 :Architecture modulaire d'Openstack

La figure ci-dessus décrit l'architecture OpenStack. Elle comprend divers services tels que le service de calcul (Nova), le service d'image (Glance), l'identité (Keystone), le stockage de blocs (Cinder), le tableau de bord (Horizon), le stockage d'objets (Swift) et le réseau (Neutron). Horizon est un tableau de bord Web qui permet aux utilisateurs d'accéder à tous les services via une interface graphique. Cependant, il

existe également une interface de ligne de commande permettant aux utilisateurs de se connecter à chacun des services.

Nova est un service de calcul qui permet aux utilisateurs de créer des machines virtuelles. Swift assure le stockage de données via des objets rapides. De la même manière, Cinder fournit un stockage en bloc attaché aux machines virtuelles en tant que volume de stockage. Glance fournit aux utilisateurs des images (système d'exploitation, logiciels, configurations, etc.) utilisées pour l'instanciation de machines virtuelles. Neutron fournit aux utilisateurs des services de réseau leur permettant de mettre en réseau des machines virtuelles utilisant des routeurs virtuels. Keystone est le service d'identité qui gère la sécurité globale d'OpenStack, y compris l'authentification et l'autorisation.

2-2 Openstack et RBAC:

Le modèle qu'Openstack implémente pour le contrôle d'accès est RBAC. Dans ce chapitre, nous allons étudier de manière détaillée cette approche.

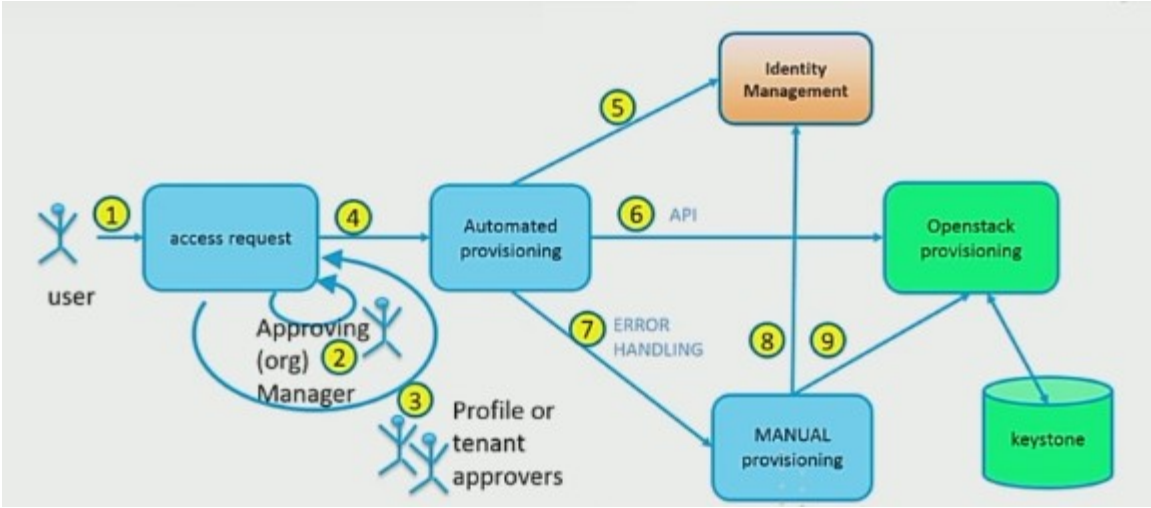

Figure 5 : Exécution du processus d'attribution de rôles dans Openstack

C'est le cas d'un utilisateur « user » qui veut accéder a une instance Openstack.

1- Il envoie une requête au système.

2/3- Cette requête peut être approuve par un superviseur, un administrateur, ou bien le

propriétaire des ressources.

4- La prochaine étape c'est le déclenchement automatique du processus de provisionnement.

5- Le provisionnement de « user » dans « Identity Management System » (Exemple :LDAP/ AD)

6- Le provisionnement de « user » dans la configuration d'Openstack.

Si le déclenchement automatique du processus de provisionnement est échoué, on exécute le processus manuellement.

- ***Remarque :***

Keystone est un module d'Openstack. Son but est centralise toutes les authentifications et autorisations nécessaires aux multiples services d'OpenStack. Il s'appuie sur un SGBDR (SQLite, MySQL ou PostgreSQL). Il utilise par défaut le port 35357.

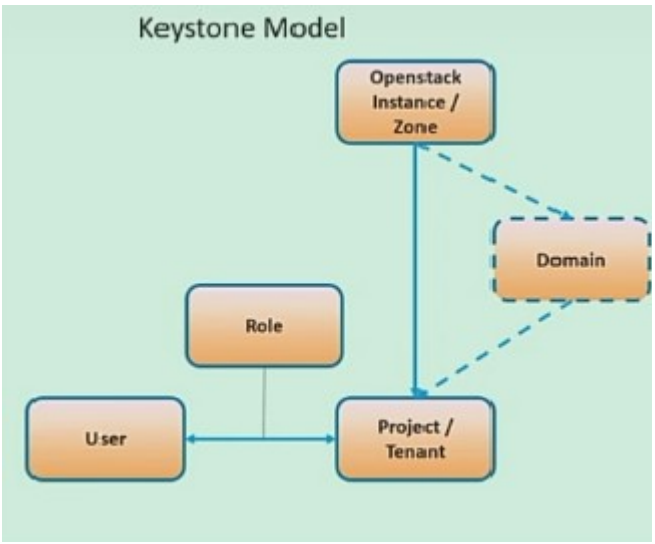

Figure 6 :Aperçu du module Keystone.

On attribue un rôle a un utilisateur ou groupe d'utilisateurs (Acteur) pour accéder a un projet. Il lui sera attribués une instance Openstack.

Afin que les Clouds OpenStack puissent prendre en charge plus efficacement plusieurs organisations d'utilisateurs simultanément, Keystone a ajouté une nouvelle

abstraction, appelée Domaine, qui pourrait permettre d'isoler la visibilité d'un ensemble de projets et d'utilisateurs (et de groupes d'utilisateurs) pour une organisation spécifique. Un domaine est formellement défini comme un ensemble d'utilisateurs, de groupes et de projets. Les domaines nous permets de diviser les ressources de notre cloud en silos destines a des organisations spécifiques.
Un domaine peut servir de division logique entre différentes parties d'une entreprise ou chaque domaine peut représenter des entreprises complètement distinctes.

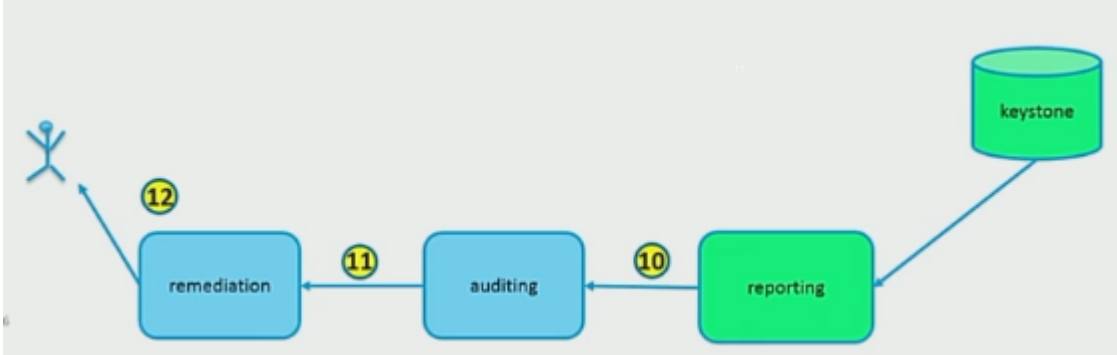

Figure 7 :Étape d'audit d'accès.

Après que le processus de provisionnement est fait, le processus de vérification des droits d'accès est déclenché.

10- Le système récupère les informations de configuration du compte courant.

11- Ceci est comparé aux enregistrements du système de gestion de compte des utilisateurs.

Le système lancera le processus de correction s'il y a une violation d'accès.

12- Le système avise la direction concernée et gère le processus d'escalade au besoin.

Sans oublier, le processus « Monitoring Employment » : Un employé qui a change de poste, ne doit pas avoir en possession ses anciens droits d'accès.

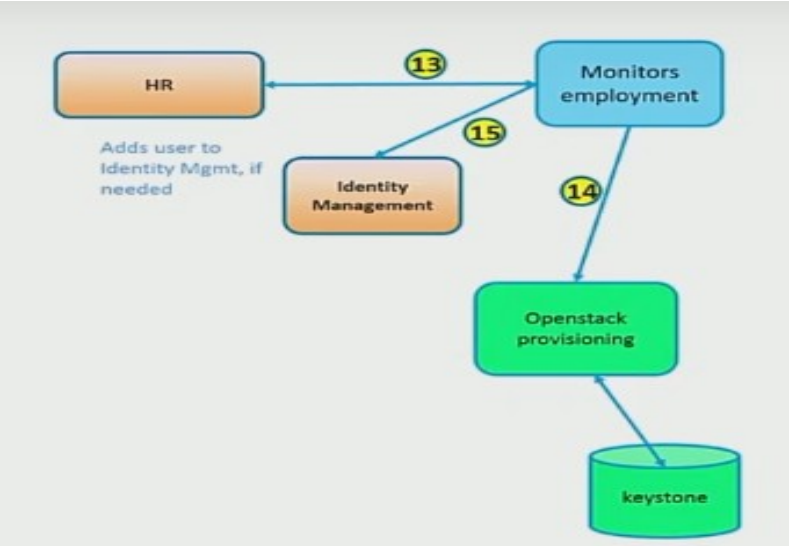

Figure 8 :processus «Monitoring Employment »

13- On reçoit une notification affirmant le changement de poste d'un employé ou la fin de contrat d'embauche…

14- Le système supprime tout les accès anciennement attribues.

15- Le système supprime l'utilisateur concerne de la LDAP/ Identity Management System.

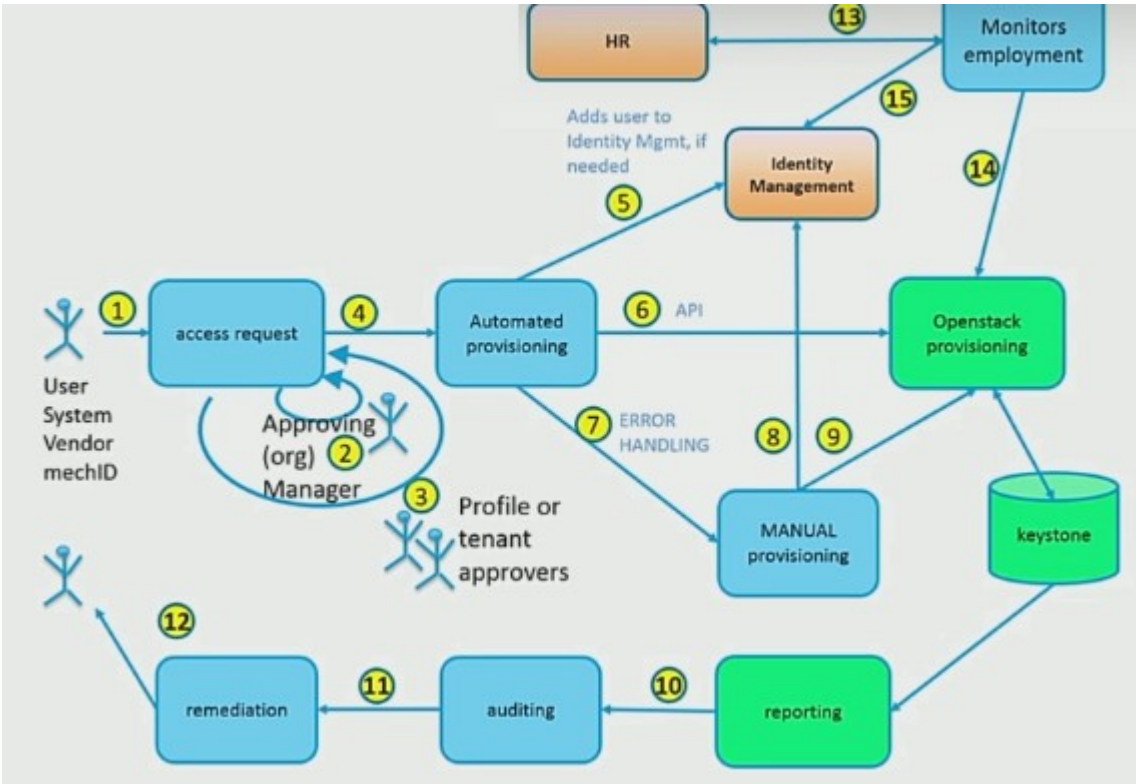

Figure 9 : implémentation de RBAC dans le cas d'OpenStack.

Openstack s'agit d'une plate-forme Cloud IaaS open source largement utilisée pour créer différents types d'architectures cloud: Cloud public, Cloud privé et Cloud hybride dans les secteurs industriel et universitaire. Avec une application aussi large d'OpenStack dans le monde réel, les clients ont déjà commencé à prendre conscience des faiblesses de RBAC, à savoir une explosion de rôles et une explosion d'autorisations de rôles.

2-3 Openstack et ABAC:

OpenStack est une plate-forme de services cloud open source de premier plan avec un cycle de développement et de publication rigoureux tous les six mois. Avec sa base de clients et ses services énormes, OpenStack sera éventuellement ou peut-être déjà confronté au problème de l'explosion des rôles ou de l'explosion des autorisations des rôles. Il est donc inévitable qu'OpenStack adopte un mécanisme de contrôle d'accès plus souple, tel que ABAC. Cependant, avec tant de versions précédentes et de développement continu, il est difficile de remplacer complètement la fondation d'autorisation RBAC par ABAC.

Dans RBAC, les politiques de contrôle d'accès ne peuvent être définies que sur la base de rôles limitant la flexibilité du contrôle d'accès. Considérant que, ABAC offre une grande flexibilité pour exprimer des politiques de contrôle d'accès détaillées d'une manière simple et plus puissante basée sur les attributs des utilisateurs, des sujets et des objets. Avec les progrès d'ABAC et de ses capacités, il est essentiel de développer des modèles ABAC pour des applications et des systèmes réels. Cependant, il est difficile pour les systèmes existants de s'adapter instantanément aux politiques de contrôle d'accès basées sur des attributs car ils nécessitent un système de gestion des attributs et du contrôle d'accès bien défini et robuste pour leur mise en œuvre. La transformation des politiques RBAC en ABAC doit être progressive, mais nous verrons finalement une adoption et une mise en œuvre à grande échelle des modèles ABAC dans l'industrie. En particulier, combiner ABAC avec des rôles est une voie de transition prometteuse.

Une deuxième motivation pour combiner plusieurs modèles de contrôles d'accès est l'augmentation du nombres de clients, et par la même occasion , augmenter les profits ; Tout cela dans le cadre de fédération de Clouds, notion que nous allons étudier par la suite.

Conclusion

Dans ce chapitre , nous avons étudié différents méthodes de contrôle d'accès ainsi que l'architecture globale d'Openstack et le fonctionnement du modèle de gestion des accès implémentés par défaut dans ce dernier. De surcroît, nous avons vu les avantages éventuelles de l'implémentation d'ABAC. De ce fait, comment pourrions-nous tirer parti de ces avantages dans le cadre d'Openstack?

Chapitre 3 : Solutions garantissant l'interopérabilité des méthodes de contrôle d'accès

Introduction

Dans cette partie, nous allons essayé de répondre a la problématique d'interopérabilité des méthodes de contrôles d'accès dans un cloud. Pour ceci , Nous allons présenter trois solutions :

1- Méthode d'évaluation de risque :

Nous allons commencer par donner une définition du terme «La fédération du cloud», ensuite nous allons introduire un modèle de contrôle d'accès dynamique base sur le risque. La fédération du cloud est une association entre différents fournisseurs de services de Cloud Computing dans le but de partager des données et des ressources afin d'accroître l'évolutivité et la disponibilité comme le montre la figure ci-dessous.

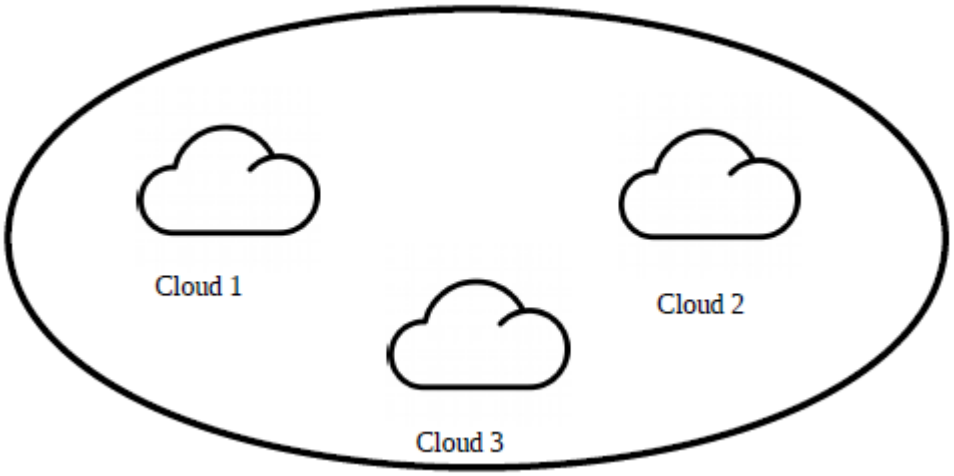

Figure 10 :Fédération de Clouds

- ***Remarque :*** Les différents Clouds peuvent adopter des modèles de contrôle d'accès différents.

Cependant, l'un des problèmes les plus importants dans l'établissement et le fonctionnement d'une fédération de clouds est la gestion des identités et des accès (IAM = Identity and access management) . Dans un seul nuage, il est possible d'utiliser des procédures IAM et des modèles d'autorisation traditionnels pour gérer

le contrôle d’accès, car tous les utilisateurs et toutes les ressources se trouvent dans le même domaine de sécurité. Toutefois, lorsque les ressources et les sujets sont redimensionnés en une fédération de nuages, il est néanmoins préoccupant que les sujets puissent provenir d'un domaine de sécurité différent de celui de la ressource à laquelle ils demandent l’accès.

Pour mettre en œuvre l'autorisation à l'aide de modèles tels que le contrôle d’accès basé sur un rôle (RBAC) ou le contrôle d’accès base sur des attributs (ABAC), le nuage doit utiliser les informations fournies par le système concernant les utilisateurs. Ces informations peuvent être, par exemple, l’identité de l'utilisateur ou des attributs de cette identité, tels que le nom, le rôle organisationnel et la date de naissance.

Pour qu'un nuage puisse faire confiance a l’identité ou aux informations d'attribut d'un utilisateur provenant d'un autre nuage, les deux nuages doivent partager un certain accord de confiance. C'est pourquoi ce processus est généralement médiatisé par une fédération d’identités. Avec la gestion des identités fédérées (FIM), chaque participant de la fédération est censé accepter que les informations reçues par un autre participant sont correctes, dans ce qu'on appelle un cercle de confiance (CoT).

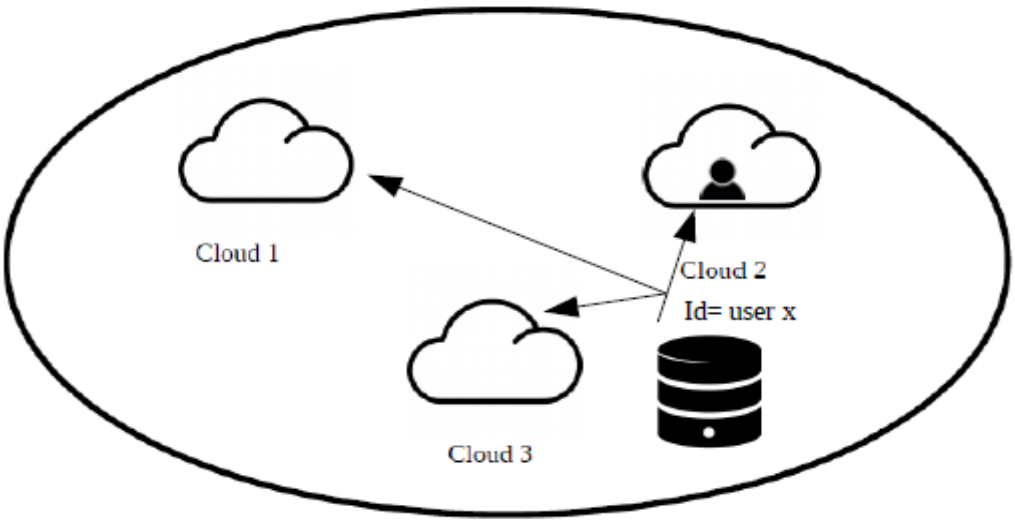

Figure 11 : Fédération d’identités

Cette solution n’est pas optimale, car les fédérations d’identités posent des problèmes tels que la nécessité d’accord d’attributs et de confiance (avoir un accord

sur les informations que les fournisseurs de cloud peuvent partager entre eux 'sur leurs utilisateurs', des accords commerciaux ,définir les responsabilités …)

On va entamer dans ce rapport, une solution alternative qu'est : **L'utilisation d'un modèle de contrôle d'accès dynamique base sur le risque sans la nécessité, mais la possibilité, d'utiliser des fédérations d'identité**. Cet idée a été prononce dans l'article *« Toward Quantified Risk-Adaptive Access Control for Multi-tenant Cloud Computing »* où les auteurs affirment que ce modèle permet de résoudre les problèmes posés par la multi organisation et qu' **un environnement dynamique nécessite un modèle de contrôle d'accès dynamique.**

L'autorisation ou le contrôle d'accès est le processus par lequel un système garantit que les demandes d'accès soit validées à l'aide de règles bien établies. Ces règles sont connues sous le nom de politiques, et le moyen de les appliquer ainsi que les mécanismes utilisés dans cette application est appelé modèle de contrôle d'accès.

Contrairement aux modèles classiques, le contrôle d'accès dynamique présente la particularité de n'utiliser que des règles prédéfinies pour calculer les décisions d'accès. Ces modèles sont basés sur des caractéristiques dynamiques, qui sont évaluées en temps réel lorsque le sujet demande l'accès à une ressource. Des caractéristiques telles que la confiance, le contexte, l'historique et le risque sont souvent utilisés pour prendre des décisions, et les caractéristiques à utiliser et à mesurer sont analysées dans plusieurs travaux

Les modèles de contrôle d'accès fondés sur les risques permettent aussi au système un meilleur traitement des demandes d'accès exceptionnelles.

On donne l'exemple d'un établissement de santé où seuls les médecins ont accès aux antécédents des patients, mais en cas d'urgence, une infirmière peut avoir besoin de ces informations pour sauver la vie d'un patient. Si ce type de situation n'était prévu dans aucune politique, l'infirmière ne sera pas en mesure de s'acquitter de sa tâche où l'infirmière peut se voir accorder un accès plus large que nécessaire.

Dans les deux cas, cela représente un risque plus grand pour le système que si un système de contrôle d'accès dynamique était utilisé et que les besoins en contrôle d'accès étaient évalués par demande.

Accorder un accès spécial dans des cas exceptionnels implique généralement une forme de surveillance par le système. Il peut s'agir: d'obligations, qui sont des conditions postérieures qu'un utilisateur doit remplir pour conserver son droit d'accès, un système de réputation qui enregistre les actions des utilisateurs et leur attribue des récompenses et des pénalités, ou un système de marche dans lequel les utilisateurs disposent d'un nombre limité de points pouvant être utilisés pour « acheter » des accès exceptionnels.

Il existe différentes approches du contrôle d'accès basées sur les risques, mais elles partagent toutes des caractéristiques communes. Un aperçu d'un modèle de contrôle d'accès basé sur les risques est présenté dans la figure ci-dessous.

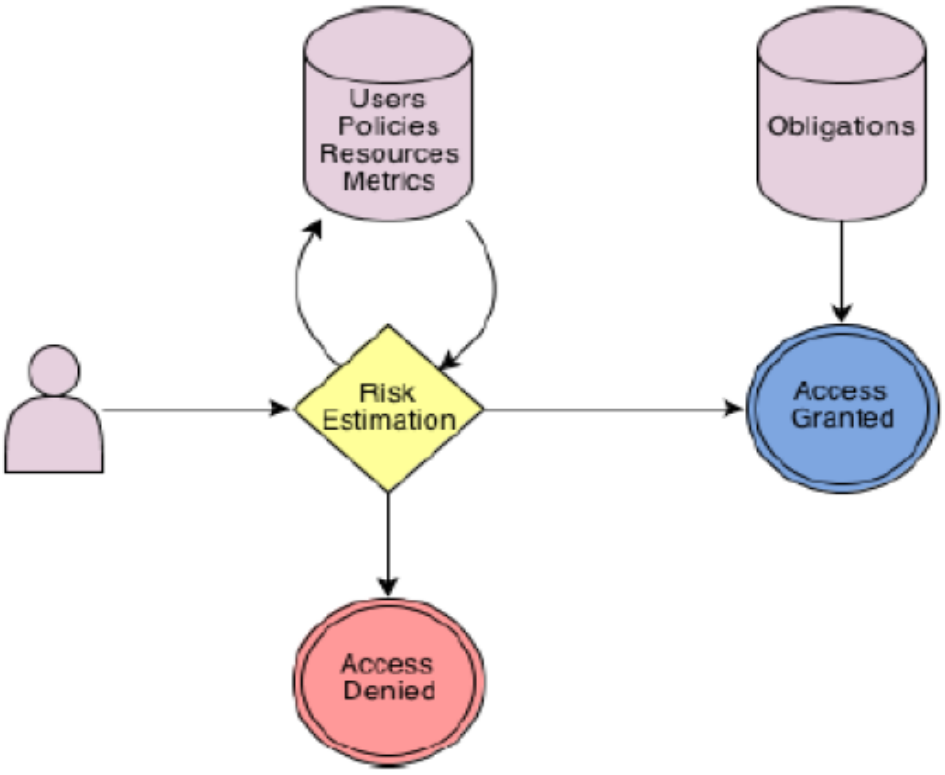

Figure 12 : Aperçu d'un modèle basé sur le risque

Le sujet tente d'accéder à une ressource en émettant une demande d'accès, qui est ensuite traitée par un moteur d'estimation du risque qui utilise toutes les informations jugées nécessaires pour prendre une décision. Il existe généralement un seuil de risque défini par les administrateurs système, et si le risque est inférieur à ce seuil, l'accès est accordé. D'autres variations mesurent le risque par rapport aux avantages d'un accès et déterminent le plus important.

La figure ci-dessous représente le mécanisme de manière plus détaillé.

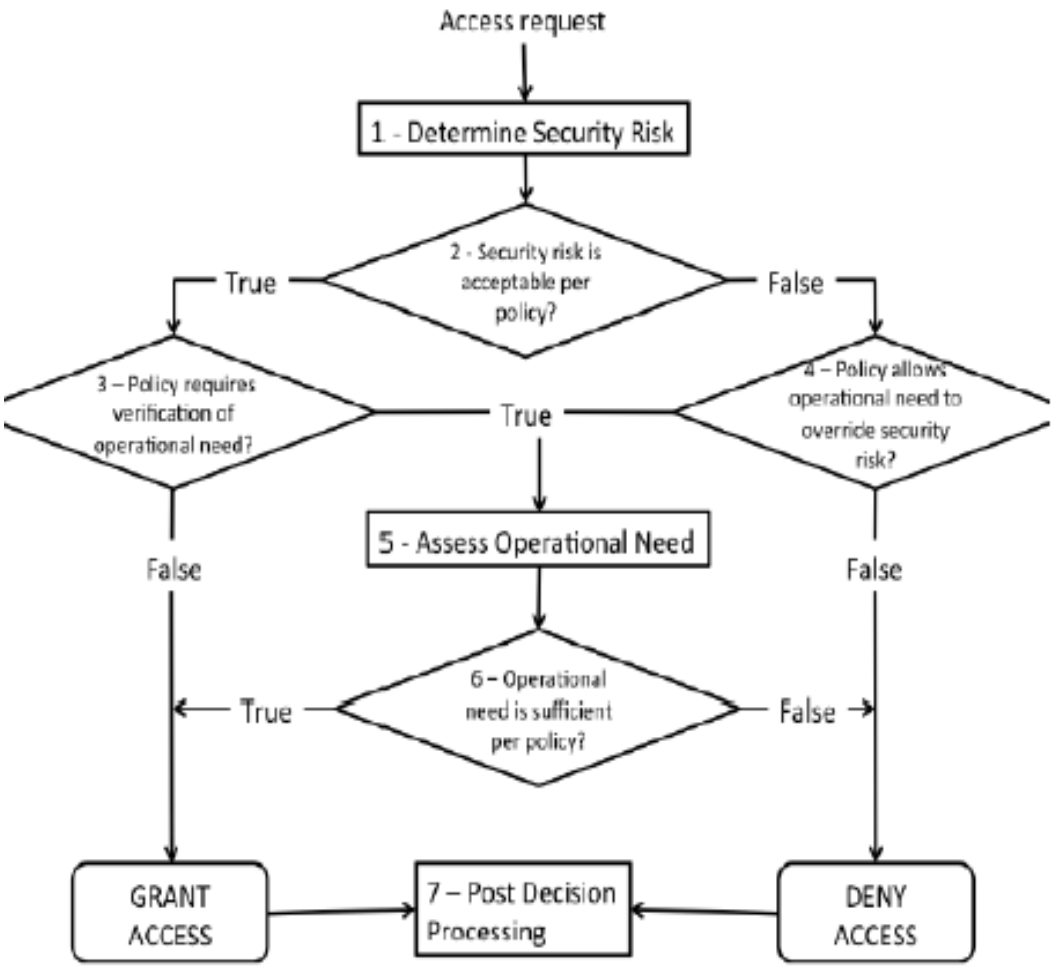

Figure 13 : Mécanisme d'accès basé sur le risque

Passons a la fédération de Clouds. Une vue d'ensemble de son architecture est représentée par la figure ci-dessous.

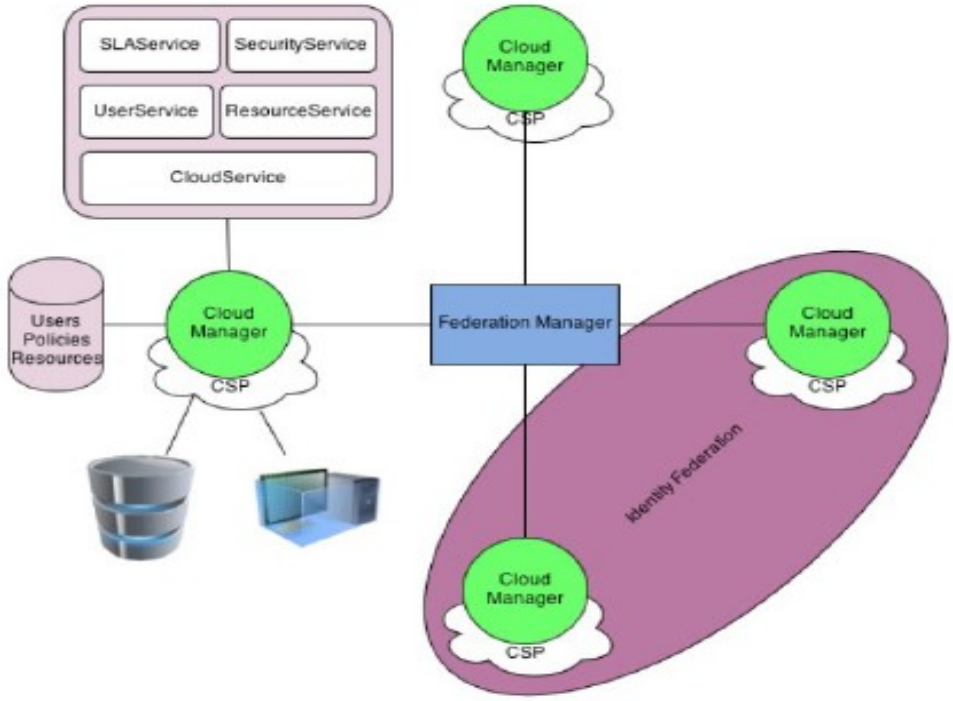

Figure 14 : Architecture de fédération de Clouds

- **CloudProvider** : (CSP) fournit l'infrastructure sur laquelle les ressources virtuelles sont allouees.
- **CloudManager** : Responsable de la connexion d'un CloudProvider a la federation.
- **FederationManager** : responsable de la coordination de la fédération. Il agit comme un service de nommage et est également responsable du passage des messages.

Dans la figure précédente, certains clouds peuvent former des fédérations d'identité entre eux. Sous le point de vue d'un utilisateur, il existe deux types de nuages dans cette architecture: **un nuage domestique** (le CSP original de l'utilisateur) et **des nuages étrangers** (les autres nuages de la fédération). Les utilisateurs peuvent déployer et accéder aux ressources dans les deux types de cloud, mais le contrôle d'accès se comporte différemment pour chaque cas.

Lorsque les utilisateurs déploient une ressource dans leur cloud domestique, ils peuvent choisir si elle sera disponible pour les utilisateurs des clouds étrangers. Dans tous les cas, l'utilisateur doit télécharger un fichier de 'policy' XACML avec la ressource.
Les utilisateurs peuvent également déployer des ressources dans un nuage étranger. Ce dernier sera automatiquement disponible pour tous les utilisateurs de la fédération. Enfin, les utilisateurs peuvent accéder aux ressources de leur cloud domestique ou aux ressources partagées de clouds étrangers.

Lorsqu'un utilisateur tente d'accéder à une ressource de son cloud d'origine, cette demande est gérée par un modèle ABAC classique. En fonction des attributs de l'utilisateur et des règles XACML, le système accorde ou refuse l'accès demandé.

Lorsqu'un utilisateur tente d'accéder à une ressource dans un cloud étranger, le système vérifie d'abord si les deux clouds se trouvent dans une fédération d'identités. Dans ce cas, l'accès sera également géré par ABAC. S'il n'y a pas de fédération d'identités entre eux, le mécanisme de *«Break the glass»* est activé et le point de décision de la politique de contrôle d'accès (PDP) basé sur le risque est appelé.

Le PDP est situé dans le cloud et traite cette demande d'accès. Les métriques et les paramètres d'estimation du risque sont définis par les administrateurs de ce cloud et les utilisateurs propriétaires des ressources.

Ces métriques sont renseignées dans un fichier XML. Ce fichier contient les définitions des métriques de risque et leur méthode de mesure et d'agrégation, ainsi qu'un niveau seuil permettant d'accorder l'accès à la ressource et les éventuelles obligations que devront respecter les utilisateurs. Ce fichier est appelé politique de risque (Risk policy).

Chaque fournisseur de cloud doit fournir un ensemble de métriques de base avec leurs règles de quantification. Ceux-ci seront utilisés pour créer une politique de risque de base pour le fournisseur. Cela garantit que les fournisseurs de cloud sont en mesure de maintenir ces exigences de sécurité minimales.

Chaque ressource a sa propre politique de risque, qui doit respecter ce qui est défini dans la politique de base, mais peut être étendue pour devenir plus ou moins restrictive à la demande de l'utilisateur. Le fichier XML de la stratégie doit être téléchargé par les utilisateurs lorsqu'ils choisissent de déployer une ressource partagée. Le système ne génère pas de stratégies de risque à la volée et toutes les stratégies de risque doivent suivre un schéma XML prédéfini, afin que différents clouds puissent communiquer.
Si un utilisateur choisit de définir une mesure de risque qui n'est pas disponible sur le serveur, il doit fournir un moyen au CSP de quantifier ce risque. Cela se fait en définissant un service Web qui sera appelé par le PDP lors de l'évaluation de la demande d'accès. Le PDP transmettra la demande d'accès au service Web, qui devra l'analyser, la traiter et renvoyer une valeur numérique représentant le risque associé à la mesure en cours d'évaluation.

Pour gérer la demande d'accès à une ressource donnée, toutes les métriques sont valorisées, en fonction des règles définies par le fournisseur de services cryptographiques et des services Web définis par l'utilisateur. Le moteur d'agrégation choisi est utilisé pour atteindre une valeur de risque finale. Cette valeur est ensuite comparée au seuil défini et, si elle est inférieure, le sujet bénéficie d'un accès spécial.

Avant d'accorder l'accès, toutefois, la stratégie est analysée à la recherche des obligations définies par l'utilisateur. Ces obligations sont stockées dans un moniteur système, qui surveille et enregistre chaque action de l'utilisateur une fois que l'accès est accordé. La figure ci-dessous montre un exemple de fichier de politique de risque. Dans cet exemple, une métrique pour le chiffrement de la couche de transport sera quantifiée, ainsi que d'autres métriques. Ils seront agrégés en fonction d'une règle de valeur maximale. Si la valeur finale est inférieure à 10, l'accès sera accordé.

```xml
<risk-ac>
    <resource id="1"/>
    <user id="2"/>
    <metric-set name="transport layer">
        <metric>
            <name>Transport Layer Encryption</name>
            <description>Quantifies the strength of the encryption scheme
            used in the access request</description>
            <quantification>https://example.com/quantify-tl-encryption
            </quantification>
        </metric>
    </metric-set>
    <aggregation-engine>maximum_value<aggregation-engine>
    <risk-threshold>10</risk-threshold>
</risk-ac>
```

Figure 15 : Fichier XML de politique de risque

La figure ci-dessous présente un déroulement pas à pas de la gestion d'une demande d'accès dans un nuage étranger.

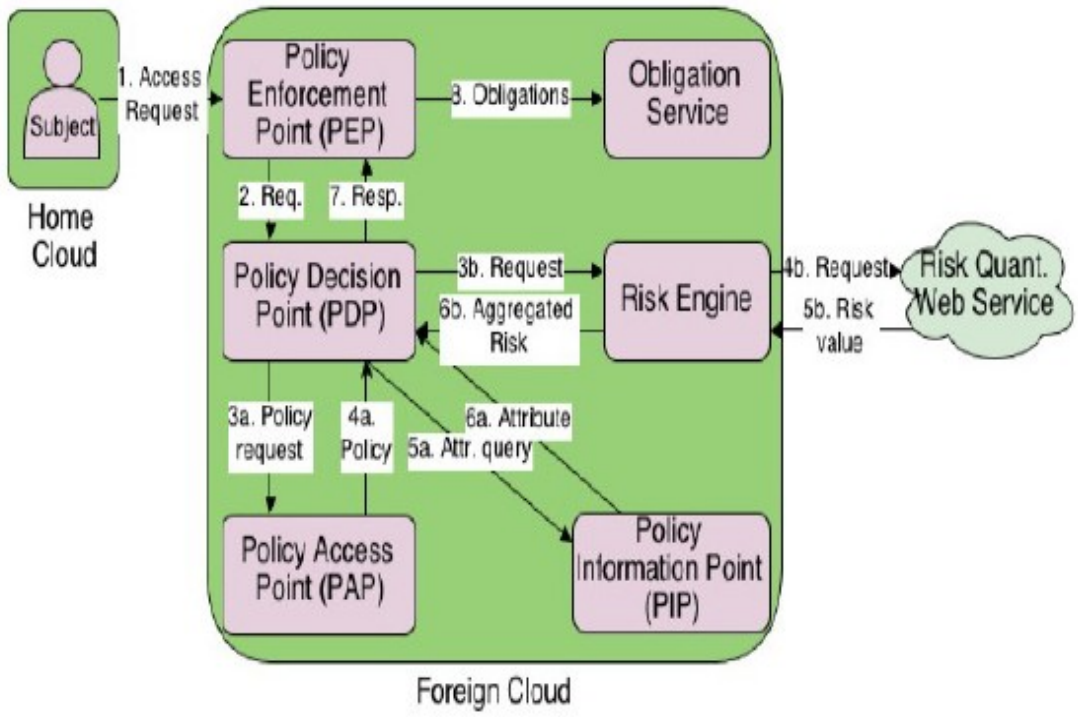

Figure 16 : Gestion d'une demande d'accès

Dans cette figure, **l'étape 1** est l'émission d'une demande d'accès d'un utilisateur à une ressource partagée étrangère (étant donné que les ressources qui ne sont pas partagées ne sont pas visibles par les utilisateurs étrangers). Le point d'application de la politique (PEP) reçoit cette demande et la transmet à un PDP (étape 2). Le PDP vérifie si le nuage d'origine de l'utilisateur et le nuage étranger participent à la même fédération d'identité. Si c'est le cas, le PDP demande les stratégies XACML applicables à la ressource (étape 3a), le point d'accès de politique (PAP) répond à cette demande (étape 4a) et le PDP extrait les attributs nécessaires du point d'information de politique (PIP) (étapes 5a et 6a).

Les étapes 3a à 6a représentent une décision de contrôle d'accès XACML classique. Toutefois, si le nuage d'origine de l'utilisateur et le nuage étranger ne font pas partie de la même fédération d'identité, le PDP transmettra la demande d'accès à un moteur de gestion des risques (**étape 3b**). Ce moteur de risque analysera ensuite le fichier de définition de risque XML associé à la ressource et quantifiera les métriques définies. Si les règles de quantification sont locales, les fonctions prédéfinies sont appelées. Si l'une des règles est définie dans un service Web, il est alors appelé, la demande d'accès étant un paramètre (**étape 4b**). Le service Web de quantification du risque joue son rôle et renvoie une valeur de risque. Une fois toutes les mesures évaluées, le moteur de risque applique une règle d'agrégation, qui est toujours locale. Le risque agrégé est ensuite renvoyé au PDP, qui utilise cette valeur pour décider de l'octroi de la demande d'accès, à nouveau en fonction de ce qui est défini dans le fichier XML. Après avoir pris une décision, le PDP le renvoie au PEP, qui applique les obligations nécessaires. La nature dynamique du contrôle d'accès est présente dans le système car la décision d'accès peut varier en fonction des informations contextuelles évaluées par les métriques.

2- Solutions Proxies :

Framework multi cloud basé sur les proxys (Faciliter la collaboration sans nécessiter d'accord préalable entre les fournisseurs de services cloud**)** permet des collaborations dynamiques à la volée et le partage des ressources entre services basés sur le cloud, abordant les problèmes de confiance, de politique et de confidentialité sans accords de collaboration préétablis ni interfaces standardisées.

- ***Exemple :*** Architecture globale de déploiement d'un Framework Cloud-Hosted Proxy.

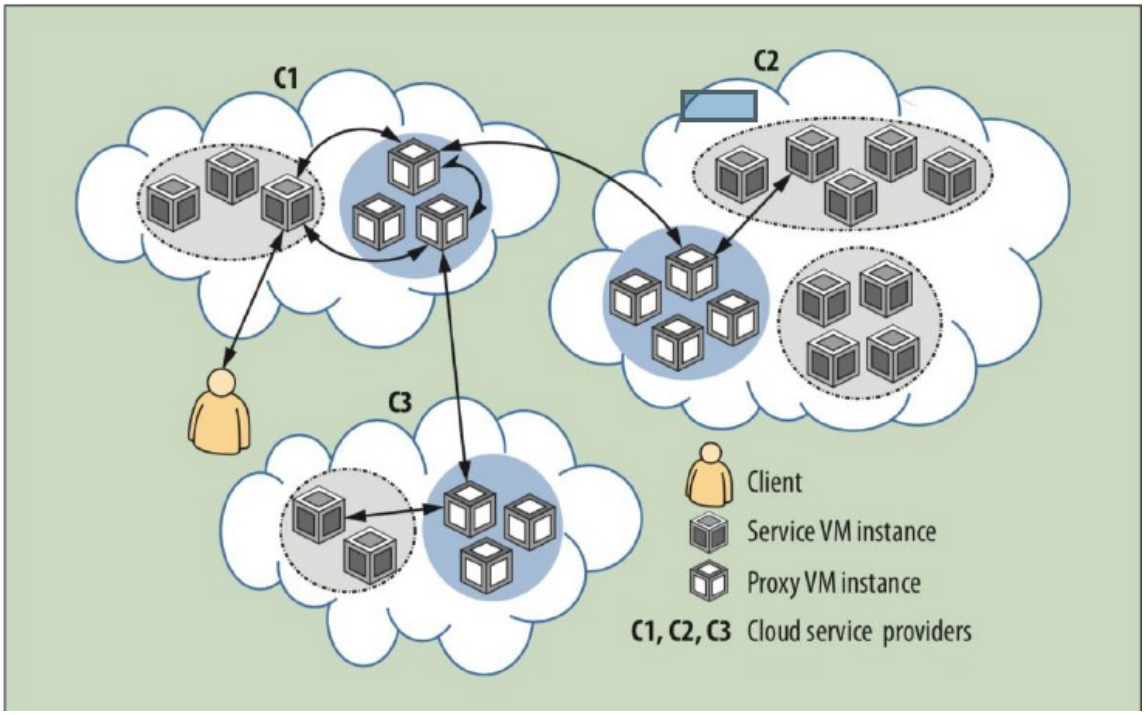

Figure 17 : Cloud-Hosted Proxy

Chaque CSP (*Cloud service provider*) peut héberger des proxys dans son infrastructure cloud, gérer tous les proxys de son domaine administratif et gérer les demandes de service venant de clients souhaitant utiliser ces proxys à des fins de collaboration. Les instances de proxy devront peut-être être spécifiques au CSP. Par exemple, sur la figure au-dessus, C1 et C2 peuvent à la fois fournir de manière mutuelle et dynamique une logique de partage et de collaboration en tant qu'instances virtuelles proxy dans leurs domaines administratifs respectifs.

3- Solution « Policy Machine »

Policy Machine (PM) est un cadre général de contrôle d'accès basé sur des attributs développé par l'Institut national de la normalisation et de la technologie (NIST). C'est une application open source.

3-1 Machine de politique

La machine de politique est un mécanisme de contrôle d'accès qui comprend: (1) des données de contrôle d'accès utilisées pour exprimer des politiques de contrôle d'accès et fournir des capacités de services de données pour effectuer des opérations sur des objets; (2) un ensemble d'opérations administratives pour configurer le contrôle d'accès; et (3) un ensemble de fonctions pour appliquer la politique sur les demandes d'exécution d'opérations sur des objets et pour calculer les décisions d'accès afin de prendre en compte ou de rejeter ces demandes en fonction de l'état actuel des données de contrôle d'accès.

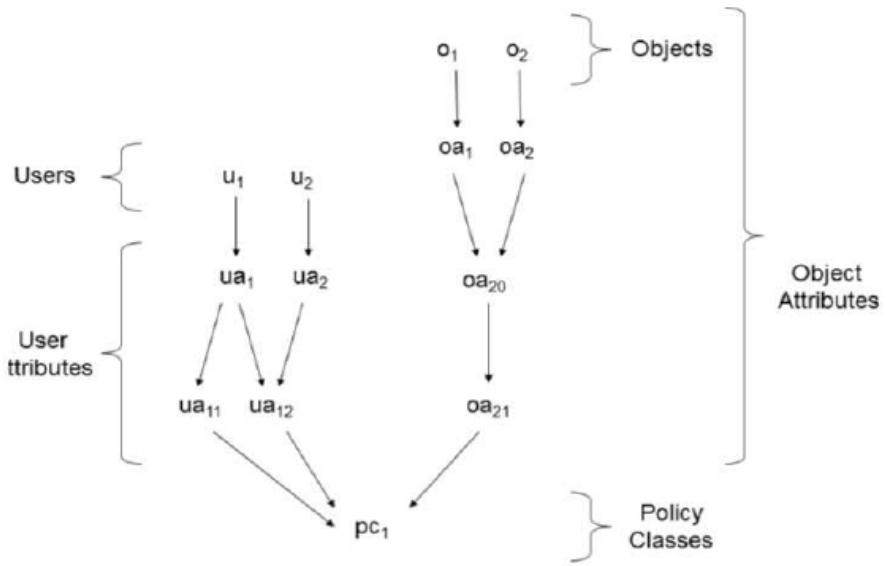

Figure 18: diagramme simplifié d'éléments de politiques

On a utilisé la version 1.5 d'Harmonia et les concepts et architectures de gestion de la performance sont pertinents pour cette version. La PM peut exprimer et appliquer des politiques de contrôle d'accès arbitraires, spécifiques à l'organisation et basées

sur des attributs. Il s'agit d'un mécanisme permettant de définir les politiques de contrôle d'accès en termes d'un ensemble générique et normalisé de relations et de fonctions pouvant être réutilisées. L'objectif principal de la gestion de projet est de fournir un cadre unificateur permettant de prendre en charge un large éventail des politiques basées sur des attributs.

La PM comporte huit éléments ou entités de base: utilisateurs, objets, attributs d'utilisateur, attributs d'objet, opérations, processus, droits d'accès et classes de règles. Les classes de stratégie, les attributs d'utilisateur et les attributs d'objet sont des conteneurs pour les stratégies, les utilisateurs et les objets, respectivement. Il a quatre types de relations: assignation, association, interdiction et obligation et deux ensembles de fonctions: décisions de contrôle d'accès et application des politiques.

La relation d'affectation est utilisée pour définir la relation entre les utilisateurs, les attributs d'utilisateur, les objets et les attributs d'objet, et la relation d'association est utilisée pour établir l'association entre les attributs d'utilisateur et les attributs d'objet ou des objets via certaines opérations. Ces associations définissent des stratégies de contrôle d'accès. Relations d'interdiction et d'obligation sont utilisés pour appliquer des contraintes et des restrictions sur les capacités de l'utilisateur dans le contrôle d'accès spécifique politiques.

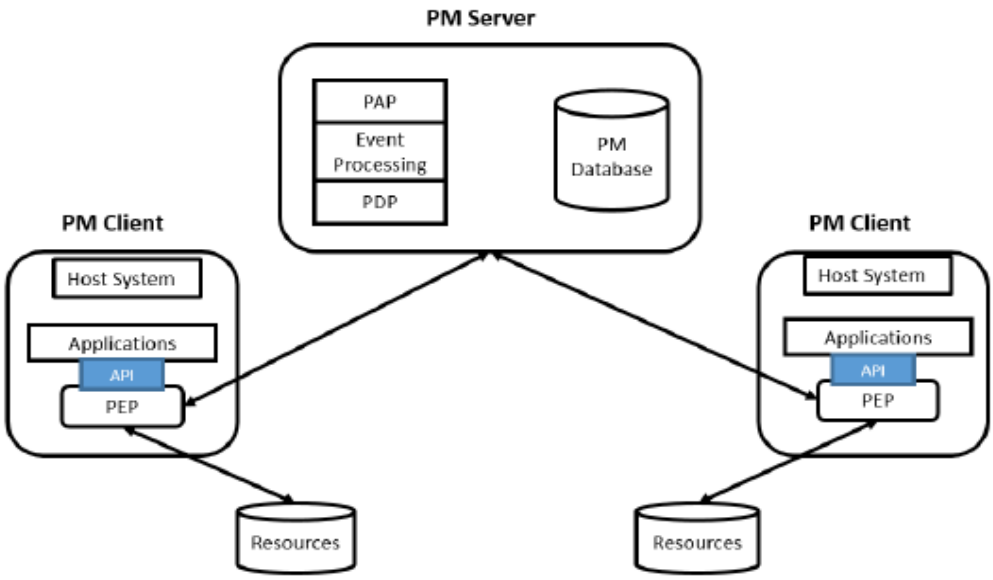

Figure 19: Architecture PM

Grâce à la relation d'affectation dans le PM, il existe une propriétée de confinement parmi les attributs PM. La propriété de confinement implique que s'il existe deux éléments x et y tels que x soit assigné à (contenu dans) y par une ou plusieurs relations d'affectation, alors x acquiert ou obtient toutes les propriétés et capacités de y en plus des siennes biens conférés. PM prend en charge les relations hiérarchiques via la propriété de confinement. Sur la base de l'ensemble existant d'éléments et de relations PM, différents types de stratégies de contrôle d'accès (par exemple, DAC, MAC, RBAC) peuvent être spécifiés et appliqués à l'aide du PM.

Dans PM, un diagramme d'éléments de politique représente un graphe de politique dirigé comprenant des éléments de stratégie de base et des affectations entre eux. La figure 18 illustre un diagramme simplifié d'éléments de stratégie. C'est un graphique inversé avec des flèches pointant vers le bas. Il montre les affectations entre différents types d'éléments de politique. Dans la figure 18, les utilisateurs et les objets sont affectés, respectivement aux attributs d'utilisateur et aux attributs d'objet. Les attributs d'utilisateur et d'objet sont attribués, respectivement à d'autres attributs d'utilisateur et d'objet. Enfin, tous les éléments sont affectés à une classe de politique.

L'architecture générale de PM est illustrée à la figure 19. Elle comprend un serveur PM, des clients PM et des dépôt de ressources. Le serveur PM comprend une base de données PM **(Active Directory)**, un point de décision de stratégie (PDP), un point d'administration de stratégie (PAP) et un module de traitement des événements.

Les clients PM sont des systèmes hôtes dans lesquels les politiques sont appliquées. Ils encapsulent les interfaces de programmation d'application (API) et les applications prenant en charge PM. Dans l'implémentation actuelle de PM, le point d'application (Policy Enforcement Point, PEP) est implémenté en tant que simulateur de noyau. Le client PM ou l'environnement utilisateur correspond au contexte dans lequel les processus PM de l'utilisateur s'exécutent. Ces processus sont similaires aux sujets. Un client PM peut être un système d'exploitation, une application (par exemple, un système de gestion de base de données), un service dans une architecture orientée service ou un environnement virtualisé. Les ressources sont les référentiels pour

différents types d'objets tels que des fichiers, des enregistrements, des répertoires, etc...

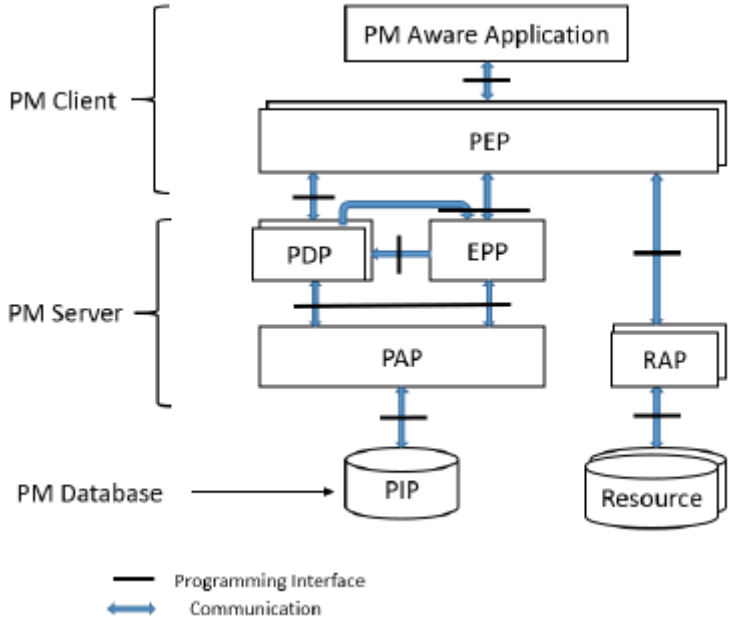

Figure 20:composants architecturaux de PM

La figure ci-dessus illustre les composants architecturaux du PM, sous un angle différent. Pour qu'une application générale soit conforme à la norme PM, les applications doivent être modifiées pour pouvoir intégrer et pouvoir communiquer avec le PEP situé dans le PM, qui communique avec le PDP pour prendre les décisions de contrôle d'accès. Un PDP détermine si une demande d'accès faite par PEP doit être accordée ou refusée conformément à la politique définie dans PAP. Toutes les informations concernant les données de contrôle d'accès et les relations sont stockées dans une base de données PM, qui est un point d'information de politique (PIP).

La Policy Machine prend en charge un ensemble complet de fonctionnalités permettant de définir des stratégies de contrôle d'accès personnalisées. Il permet de définir des relations de refus ou d'interdiction et d'appliquer des contraintes, ainsi que de combiner différentes stratégies de contrôle d'accès spécifiées dans PM à l'aide de l'outil d'administration. L'outil PM Admin est un outil basé sur une interface graphique, utilisé pour définir et administrer les stratégies de contrôle d'accès dans

PM en créant des classes de stratégies, des utilisateurs, des attributs d'utilisateur, des objets, des attributs d'objet et en définissant des ensembles d'opérations et des autorisations entre les attributs d'utilisateur et les attributs et objets. Toutes ces données, informations et relations sont stockées dans Active Directory (la base de données PM) comme dans PM version 1.5.

Les utilisateurs demandent l'accès à des objets via des clients PM qui à leur tour communiquent avec le serveur PM pour prendre des décisions en matière de contrôle d'accès et appliquer ces décisions sur les systèmes hôtes. PM suit un mécanisme de politique d'autorisation énumérée (EAP) pour spécifier des politiques de contrôle d'accès basées sur des attributs.

Le XACML *(eXtensible Access Control Markup Language)* est un des autres frameworks connus basés sur des attributs pour appliquer les stratégies ABAC. Il s'agit d'un standard OASIS qui a été utilisé pour définir les politiques ABAC dans les milieux universitaires et industriels. C'est un langage de politique de contrôle d'accès basé sur des attributs pour la gestion des accès autorisés aux ressources. Il fournit une architecture avec un ensemble standard de composants, tels que point d'administration de politique (PAP), point de décision de politique (PDP), politique point d'information (PIP), point d'application des règles (PEP), etc. pour évaluer les politiques de contrôle d'accès. Il suit une formule logique ABAC système de stratégie où la logique de prédicat est utilisée pour spécifier la stratégie règles.

On a utilisé la PM pour développer une architecture de mise en application pour les modèles ABAC. La capacité de PM et la flexibilité de spécifier et de combiner différents types de stratégies de contrôle d'accès en font un choix approprié pour son utilisation en tant qu'architecture d'application. En revanche, les applications qui l'utilisent doivent connaître sa structure et ses éléments, ce qui ajoute de la complexité. Pour simplifier le processus et faciliter l'utilisation du PM avec des applications telles que OpenStack, un moteur d'autorisation (AE) est développée dans ce projet afin de faciliter l'interaction entre le PM et les applications qui l'utilisent.

OpenStack est une plate-forme cloud open source en constante évolution qui fournit une architecture permettant d'utiliser ou d'améliorer ses services conformément aux exigences des utilisateurs ou des clients. OpenStack et PM, tous les deux étant Open source, offrent la possibilité d'intégrer une implémentation de composant d'autorisation personnalisée dans la structure d'autorisation OpenStack.

Un moteur d'autorisation personnalisé est nécessaire pour appliquer le modèle ABAC sur la plate-forme OpenStack à l'aide du PM. Ainsi, un moteur d'autorisation (AE), qui est un service RESTful, a été mis en œuvre servant d'interface entre OpenStack et PM . Dans le cadre d'application, la version OpenStack Rocky avec Identity API version 3 et PM version 1.5 (Harmonia 1.5) a été utilisée. PM est un cadre de contrôle d'accès flexible basé sur des attributs qui permet de spécifier, d'appliquer et de combiner différents types de stratégies de contrôle d'accès.

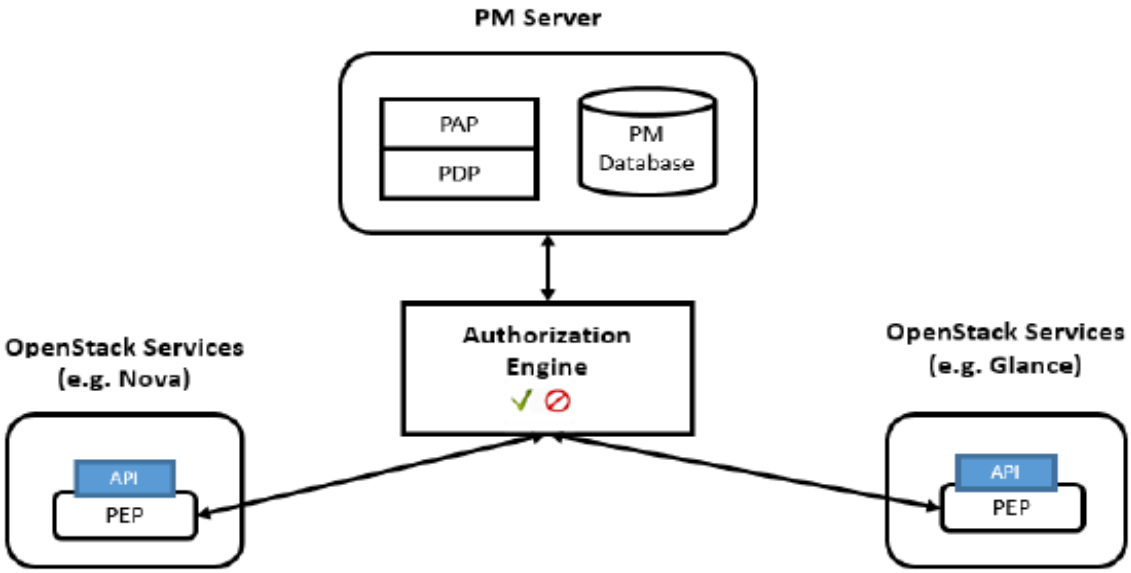

Figure 21 : Architecture centralisée de la PM

Dans ce projet, nous implémentons une architecture d'application pour appliquer les stratégies ABAC dans les plates-formes Cloud IaaS à l'aide de Policy Machine (PM), un outil de spécification de stratégie et d'application basé sur les attributs développé par NIST, ainsi qu'un moteur d'autorisation (AE). La figure ci-dessus illustre l'architecture d'exécution. Dans cette architecture, le serveur PM agit comme un point d'administration de stratégie centralisé qui renvoie un ensemble

d’autorisations d’utilisateur sur les objets en fonction des définitions de stratégie. Il se connecte à un Active Directory (AD), une base de données principale pour PM qui stocke tous les utilisateurs et leurs attributs d'utilisateur associés. Pour des raisons de simplicité, OpenStack est supposé utiliser le même AD que son serveur principal d’identité d’identité pour stocker toutes les informations relatives aux utilisateurs, y compris les attributs.

Figure 22: Fenêtre d’administration

L'architecture d'application basée sur PM utilise une architecture client-serveur où le serveur PM et AE a une relation serveur-client, et de même, AE, lui-même, agit en tant que serveur pour différents services d'OpenStack. Les services OpenStack communiquent via une API RESTful avec AE, qui à son tour communique avec le serveur PM pour prendre des décisions d'autorisation (par exemple, Autoriser / Refuser).

En raison de la nature dynamique des objets de cloud, les commandes d'OpenStack sont modélisées en tant qu'objets dans la stratégie d'autorisation définie dans PM. Ces commandes sont spécifiques aux services dans OpenStack. Par exemple, Nova a ses propres commandes, mais également d’autres services tels que Glance, Cinder,

etc. Les définitions de stratégie d'autorisation requises, généralement répertoriées dans le fichier de stratégie OpenStack (document JSON), été défini dans PM Admin Tool dans une classe de règles nommée OpenStack.

Une instance de la classe de règles OpenStack, issue de la vue «Objets / Attributs avec ACE» dans le PM outil, est illustré à la figure 19. La figure montre un exemple de stratégie de contrôle d'accès défini pour Nova. Commandes de paires de clés dans OpenStack. Ces commandes génèrent des clés ssh pour un utilisateur, qui sont utilisées à leur tour lors de la création de machines virtuelles dans Nova. Dans la figure 3.4, Admin et Manager sont des rôles (décrits en tant que conteneurs / attributs utilisateur dans PM), et compute_extension-keypair-index, compute_extension-keypaircreate, compute_extension-keypair-delete et compute_extension-keypair-show sont des commandes Nova (représentées sous la forme d'objets). en PM). Ces attributs et objets utilisateur sont associés à un ensemble d'opérations spécifique nommé aléatoirement. Par exemple, «06932852» est un ensemble d'opérations doté d'autorisations de «lecture» uniquement entre les attributs d'utilisateur (Admin et Manager) et l'objet compute_extension-keypair-index. Il signifie que les attributs PM, Admin et Manager, ont des autorisations de lecture sur compute_extension_keypair-index, de sorte qu'un utilisateur possédant l'un de ces rôles est autorisé à effectuer cette opération dans OpenStack.

3-2 Moteur d'autorisation AE

Cette section présente le moteur d'autorisation (AE), une implémentation de validation de concept permettant d'appliquer le modèle UAE-OSAC dans OpenStack à l'aide du PM. Parmi les nombreux avantages du PM, il s'agit d'un outil open source qui a conduit au développement du moteur d'autorisation (AE). AE agit comme un composant d'autorisation facilitant la communication entre les règles OpenStack, moteur et le PM. C'est un service RESTful qui fournit une interface entre OpenStack et PM en obtenant des informations sur les attributs et la liste des autorisations auprès du serveur PM et en évaluant les décisions d'autorisation. La décision d'autorisation, Autoriser ou Refuser, est ensuite renvoyée au service OpenStack spécifique où la stratégie est appliquée. Il est écrit en Java à l'aide d'une API REST et agit en tant que serveur RESTful pour les services OpenStack. Pour toute opération

effectuée par un utilisateur sous OpenStack, AE vérifie initialement le projet dans le dossier cible et le jeton. Ensuite, il se connecte au serveur PM et l'interroge via différentes commandes PM afin de prendre des décisions de contrôle d'accès basées sur le modèle UAE-OSAC pour OpenStack.

AE remplace le moteur de stratégie existant dans OpenStack et est chargé d'évaluer la stratégie définie dans PM et de renvoyer les décisions d'accès aux services OpenStack. Les services OpenStack (S) sont les points d'application des règles (PEP) qui appliquent les décisions d'accès renvoyées par AE et répondent aux utilisateurs avec les résultats appropriés, tels que fournir les informations demandées si l'accès est autorisé ou renvoyer une erreur si l'accès est refusé.

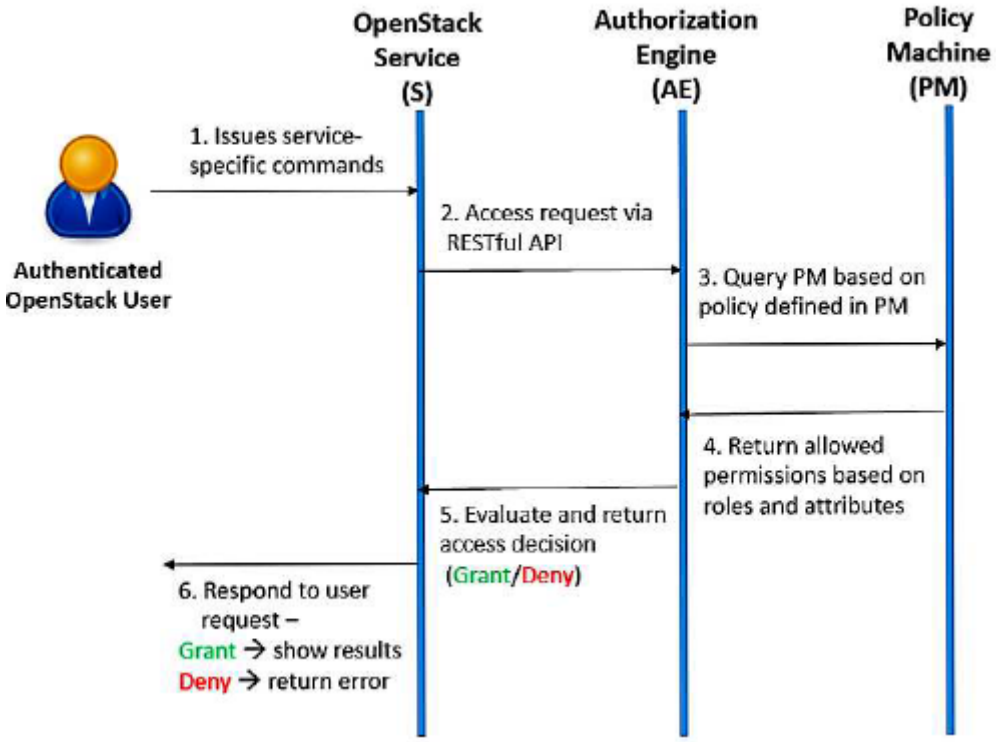

Figure 23:Diagramme de séquence de gestion des autorisations

La figure ci-dessus illustre un diagramme de séquence présentant les autorisations dans OpenStack utilisant AE et PM. Il montre la séquence d'actions impliquées dans le processus d'autorisation. Comme AE est une implémentation de validation de concept et est principalement conçu pour illustrer l'applicabilité et la faisabilité du

modèle UAE-OSAC proposé sur la plate-forme OpenStack Cloud IaaS, il n'a pas été optimisé en termes de performances. AE peut être conçu pour être un composant plus général et indépendant pouvant être utilisé avec n'importe quel outil de configuration de stratégie, tel que PM, et pouvant être appliqué à d'autres plates-formes en nuage en plus d'OpenStack.

Conclusion

Ce chapitre présente trois solutions que nous avons choisis pouvant concrétiser le concept d'interopérabilité des méthodes de contrôle d'accès dans un Cloud en général , et dans Openstack en particulier.

Dans le chapitre suivant, Nous allons passer à l'implémentation de la machine de politique *« PM ».*

Chapitre 4 : Implémentation de la solution « Machine de Politiques »

Introduction

Dans cette partie, nous allons voir les technologies que nous avons utilisées ainsi que le déploiement de la machine de politiques.

1-Technologies utilisées

Voici les technologies que nous avons utilisées pour le déploiement de « PM » :

- *Ubuntu Server* est un système d'exploitation GNU/Linux basé sur la distribution Linux Debian.

- *DevStack* développé en **Python**, fournit et maintient les outils utilisés pour l'installation des services centraux OpenStack à partir de la source.

- *Windows Server* est un système d'exploitation orienté serveur développé par Microsoft

- *Harmonia 1.5* - Première version d'une implémentation d'une machine de politiques(utilisant Active Directory en tant que magasin de règles et d'attributs)

- *Oracle VM VirtualBox* est un logiciel libre de virtualisation publié par Oracle . (Nous l'avons utilisé pour créer des machines virtuelles WinServer, qui implémente la machine de politique, et une instance UbuntuServer pour déployer Openstack)

2-Implémentation

Ce chapitre décrit la phase de l'implémentation de la solution « Machine de politique ». Tout d'abord, nous allons décrire l'outil Harmonia 1.5 . Ensuite, nous allons réaliser une simulation réelle utilisant l'environnement Openstack et la « Machine de Politique ». Cette simulation est une preuve de la possibilité d'intégrer le modèle de contrôle d'accès à base d'attribut « *ABAC »* au sein du Cloud Openstack.

2-1 Harmonia 1.5 :

Harmonia 1.5 est un mécanisme de contrôle d'accès basé sur des attributs dont les fonctionnalités principales sont la possibilité de configurer et d'appliquer des politiques de contrôle d'accès arbitraires et la capacité de protéger des ressources dans plusieurs instances de ces politiques.

L'implémentation d'Harmonia est une application à trois niveaux s'exécutant sur des serveurs *MS Windows 2008* et ultérieurs :

- **La couche logique** de l'application comprend un serveur de politique s'exécutant aussi sur un serveur *MS Windows 2008* et ultérieur dédié.

- **La couche de données** comprend un répertoire de données et les mécanismes d'accès associés. PM utilise *MS Active Directory* en tant que référentiel de données et son serveur LDAP en tant que mécanisme permettant de **stocker/extraire** des données PM **dans/à partir** du référentiel.

- **La couche de présentation** comprend des instances de l'outil d'administration, qui fournit des interfaces graphiques pour la gestion des données de gestion de la performance, et des simulateurs d'environnement utilisateur, qui gèrent les sessions / processus utilisateur et fournissent des interfaces graphiques pour l'interaction de l'utilisateur avec le système de gestion de la performance.

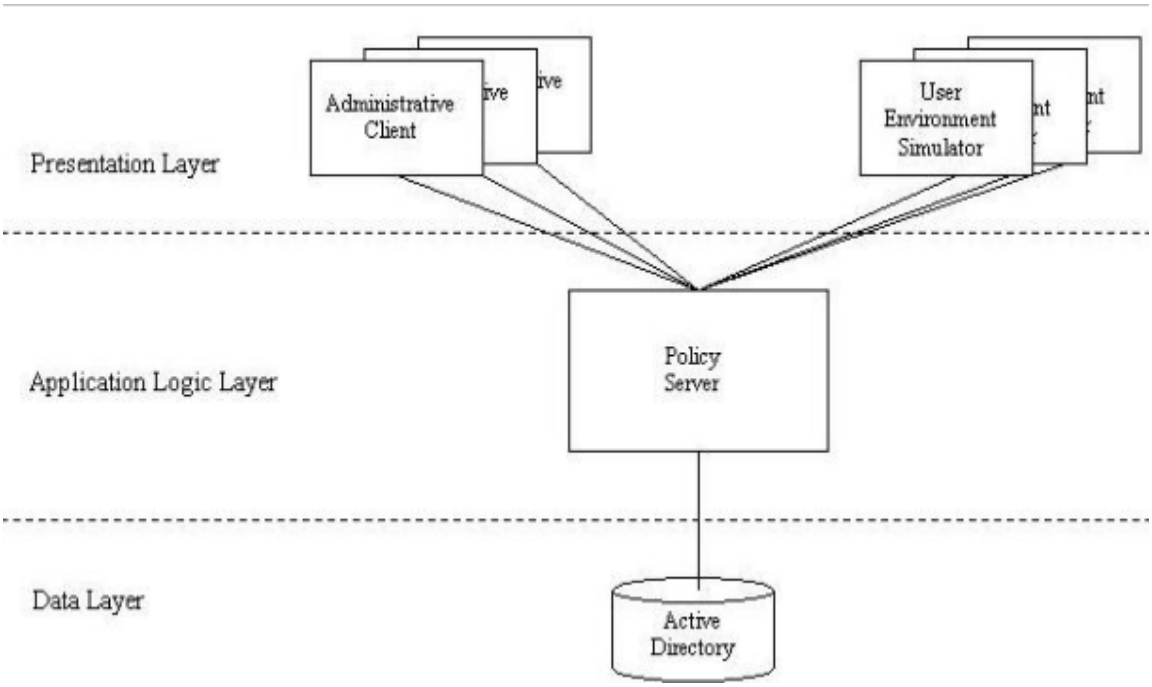

Figure 24:Architecture de la machine de politique

Par la suite, nous allons étudier de manière plus détaillé les trois couches, voir la couche logique, de données, de présentation ainsi que leurs composantes respectives serveur de politique, répertoire de données et l'outil d'administration.

2-1-1 Serveur de politiques :

Le serveur / moteur de politiques est le cœur de la machine de politique *« PM »* et gère toutes les données PM (utilisateurs, attributs d'utilisateur, classes de politiques, etc.) et relations (diverses relations "d'affectation", etc.). Les données et relations PM se trouvent dans le *MS Active Directory (AD)* et le moteur utilise le serveur LDAP afin de stocker / récupérer ses données dans / à partir de l'AD.

Le moteur agit comme un serveur pour les composants de la couche de présentation, en ce sens qu'il fournit des services via un ensemble de commandes du moteur. Un client administratif peut demander des services administratifs, tels que "créer un nouvel utilisateur", "créer un nouvel attribut d'utilisateur", "attribuer un attribut d'utilisateur à un autre attribut d'utilisateur".

Dans la figure 24, le serveur de politique, composant de la couche logique, est lancé lors de l'exécution d'un fichier .bat dont voilà un bref aperçu 'Figure 25'.

Figure 25: Server.bat

Le serveur se trouvant au chemin spécifié par la variable PM_ROOT est lancé sur le port 8080.

« Keystore » contient des clés privées et des certificats utilisés par les serveurs SSL pour s'authentifier auprès des clients SSL.

« Truststore » est le fichier contenant les certificats des serveurs SSL approuvés par des CA de confiance pour identifier les serveurs.

On a recours aux certificats numériques pour éviter toute usurpation d'identité ou toute authentification non autorisée.

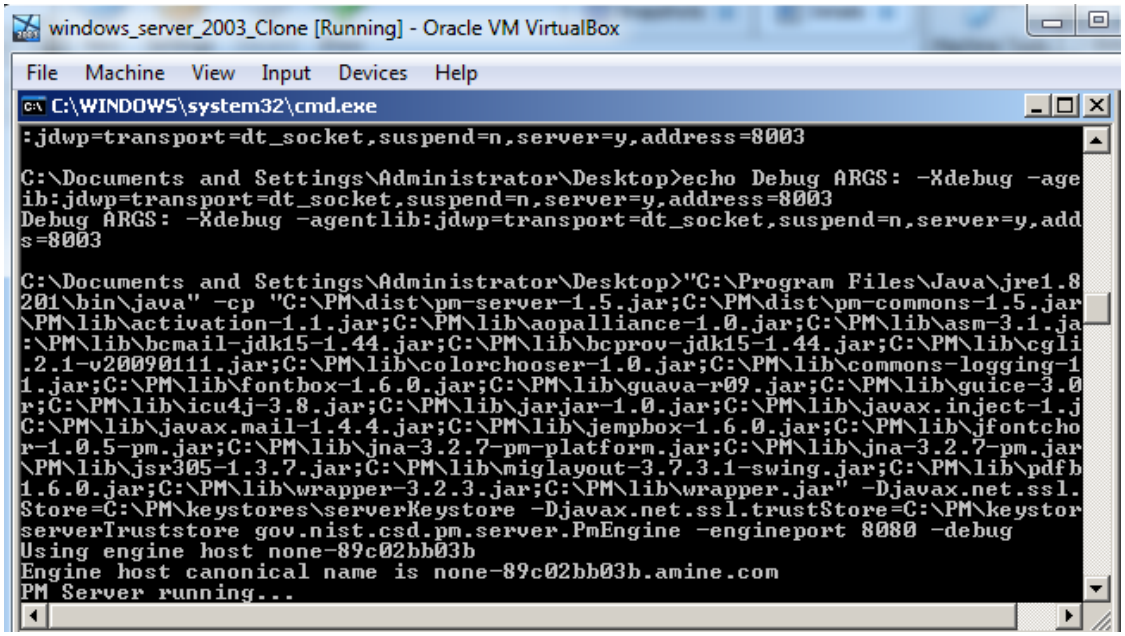

Figure 26: Démarrage du Serveur

2-1-2 Répertoire de données :

Le répertoire de données de la machine de politique stocke les données PM. Elle utilise *MS Active Directory* en tant que support de référentiel réel et le serveur LDAP en tant que mécanisme d'accès.

Figure 26:Active Directory

2-1-3 Outil d'administration PM :

Plusieurs instances de l'outil d'administration peuvent être exécutées sur le même ordinateur ou sur des ordinateurs différents du système PM. L'outil d'administration possède une interface graphique qui permet à l'administrateur de gérer les données et les relations entre les membres, par exemple en créant un nouvel attribut d'utilisateur ou en affectant un attribut d'objet à un autre attribut d'objet. Les actions de l'administrateur sur l'interface graphique du client sont traduites en une séquence de commandes du moteur PM envoyées au moteur PM. Le moteur interprète les commandes et renvoie éventuellement une réponse au client d'administration.

```
set PM_VERSION=1.5

set PM_ROOT=C:\PM

set MY_KEYSTORE=superKeystore

set MY_TRUSTSTORE=clientTruststore

set ENGINE_HOST=%computername%
set ENGINE_PORT=8080

set JAVA_JRE=%JAVA_HOME%
```

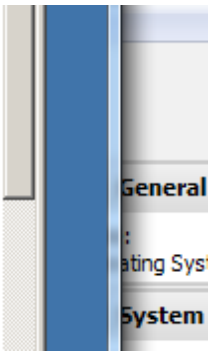

Figure 27:Super.bat

Dans la figure 27, l'outil d'administration (lancé en tant que l'utilisateur SUPER), composant de la couche présentation, est lancé lors de l'exécution d'un fichier .bat dont voilà un bref aperçu 'Figure 28'.

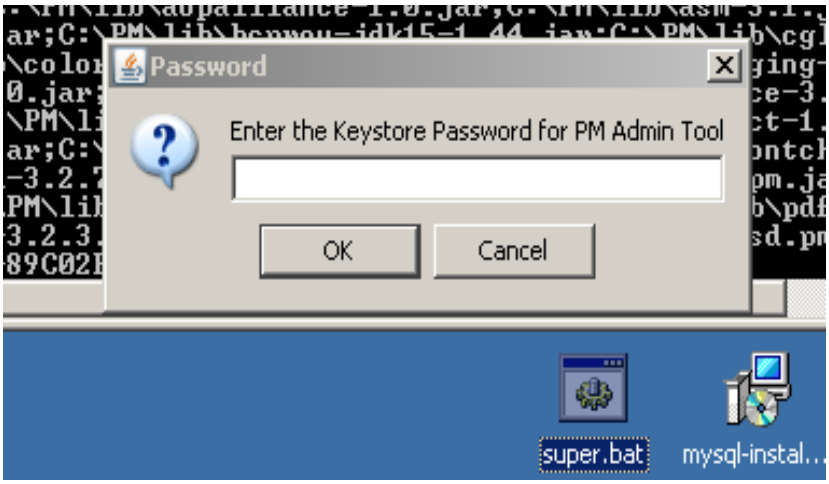

Figure 28: Saisie du mot de passe de l'Admin

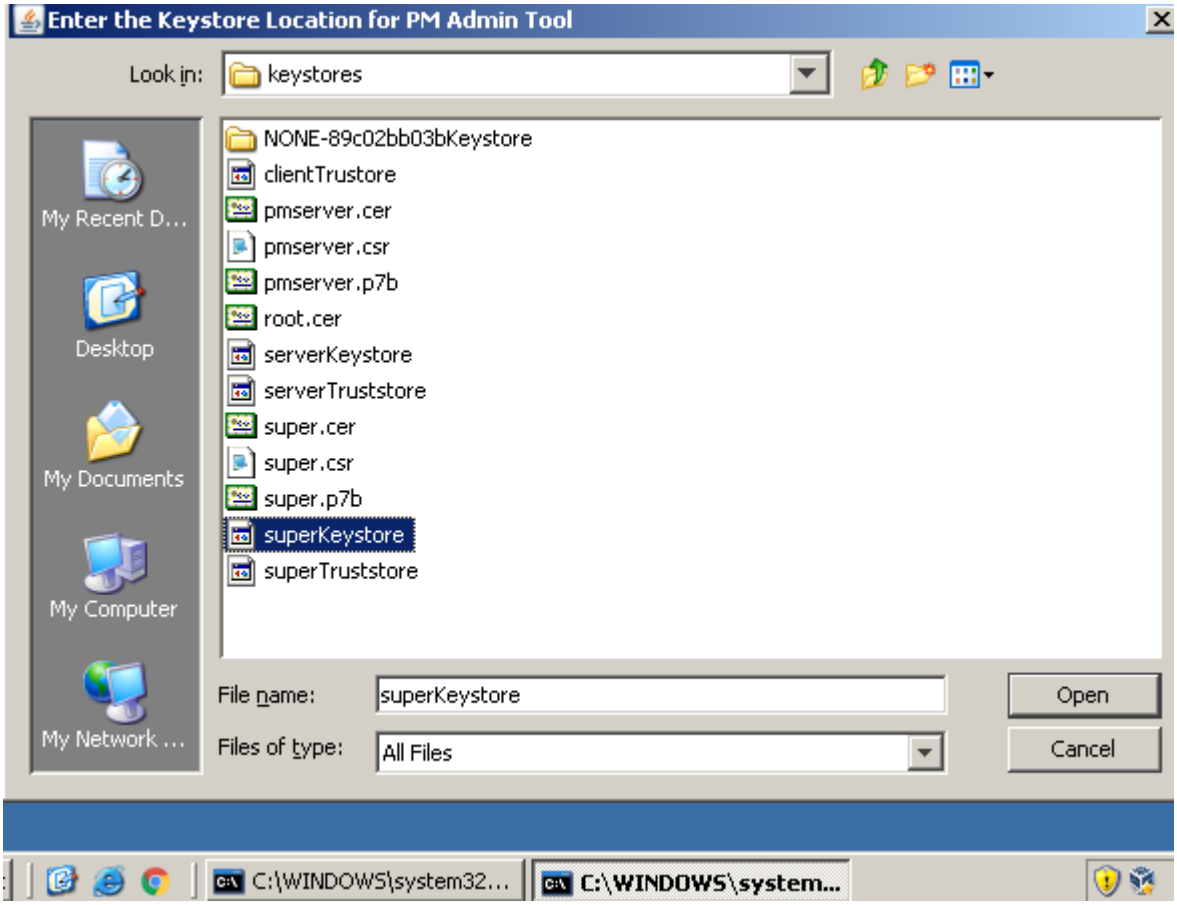

Figure 29: Sélection des certificats de l'admin

L'interface graphique par défaut de l'outil d'administration est limité aux utilisateurs, aux attributs d'utilisateur, aux classes de stratégie et au connecteur en tant que nœuds et à leurs attributions en tant qu'arcs. L'administrateur peut parcourir le graphique en cliquant sur l'un de ses nœuds. Le nœud sélectionné sera ensuite affiché avec quelques «niveaux» de nœuds vers le haut et vers le bas. Ce n'est qu'une des quatre vues possibles du graphe de la machine de stratégie. Les trois autres options pouvant être sélectionnées par l'administrateur sont: la vue d'objet, qui affiche les objets, les attributs d'objet, les classes de règles et le connecteur sous forme de nœuds et leurs affectations sous forme d'arcs; la vue des capacités, qui affiche les attributs de l'utilisateur et les capacités qui lui sont attribuées (opérations sur les attributs d'objet); et la vue des entrées de contrôle d'accès, qui affiche les attributs de l'objet et les entrées de contrôle d'accès qui lui sont attribuées (attributs d'utilisateur avec opérations).

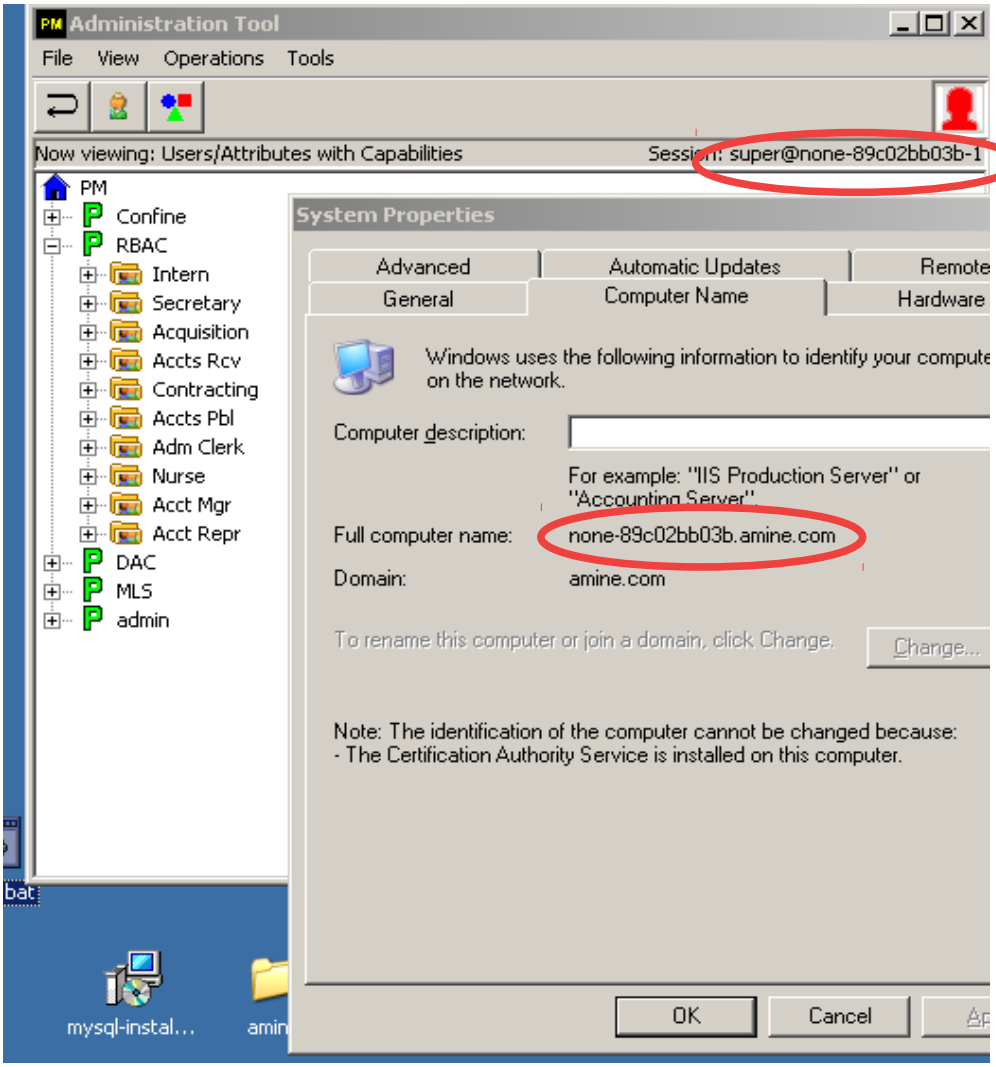

Figure 30: Outil d'administration par défaut

2-2 Harmonia et Openstack

Avant d'entamer cette partie qui traite la liaison entre la machine de politiques et Openstack, nous voudrons mettre le point sur le fait que l'outil Harmonia 1.5 **n'a pas été conçu** en premier lieu pour répondre à la problématique de l'interopérabilité des méthodes de contrôles d'accès dans **un Cloud**.

Cette section présente des cas d'utilisation avec deux types de politiques de contrôle d'accès, tout d'abord *avec les rôles d'utilisateur uniquement*, et ensuite *avec les rôles d'utilisateur et les attributs d'utilisateur*. Ces cas d'utilisation illustrent comment le **RBAC** déjà existant et le modèle **ABAC** proposé fonctionneraient dans un environnement spécifique, dans ce cas, **OpenStack**.

Les deux modèles de contrôles d'accès peuvent être appliqués dans OpenStack à l'aide de **PM** et **AE**.

Nous allons simuler le modèle de contrôle d'accès basé sur les rôles adopté par Openstack à l'aide de notre outil PM. Ensuite, nous allons essayer d'introduire le modèle de contrôle d'accès basé sur les attributs en se basant sur le même outil. Si succès, nous avons alors démontrer la possibilité de l'interopérabilité des modèles de contrôles d'accès d'Opentack ; et par cela nous avons résolus la problématique annoncé dans le premier chapitre.

2-2-1 RBAC dans Openstack

Le premier cas d'utilisation présente une politique RBAC, équivalente à la celle dans OpenStack.

- **Avec deux rôles**: *Admin* et *Manager*,
- **Quatre commandes Nova**: compute_extension-keypair-index,

  compute_extension-keypair-create,

  compute_extension-keypair-delete

  et compute_extensionkeypair-show

De même, d'autres commandes dans différents services d'OpenStack peuvent être incorporées dans les politiques d'autorisation. Les autorisations utilisateur sont déterminées par les rôles attribués à un utilisateur dans un projet spécifique. Les commandes **Nova keypair** permettent de **générer des clés ssh** pour un utilisateur. Ces clés sont utilisées lors de la création de machines virtuelles. Un utilisateur peut créer, supprimer, répertorier et afficher les détails des paires de clés à l'aide de ces commandes. Les règles d'autorisation pour chaque commande, pour un utilisateur **U**, sont données ci-dessous.

Pour exécuter, les commandes *« create et delete »*, il faut avoir le rôle **Admin**. Pour les commandes *« index et show »*, il suffit d'avoir le rôle **Manager.**

Pour appliquer cette politique d'autorisation, une politiques d'autorisation équivalente est spécifiée dans PM.

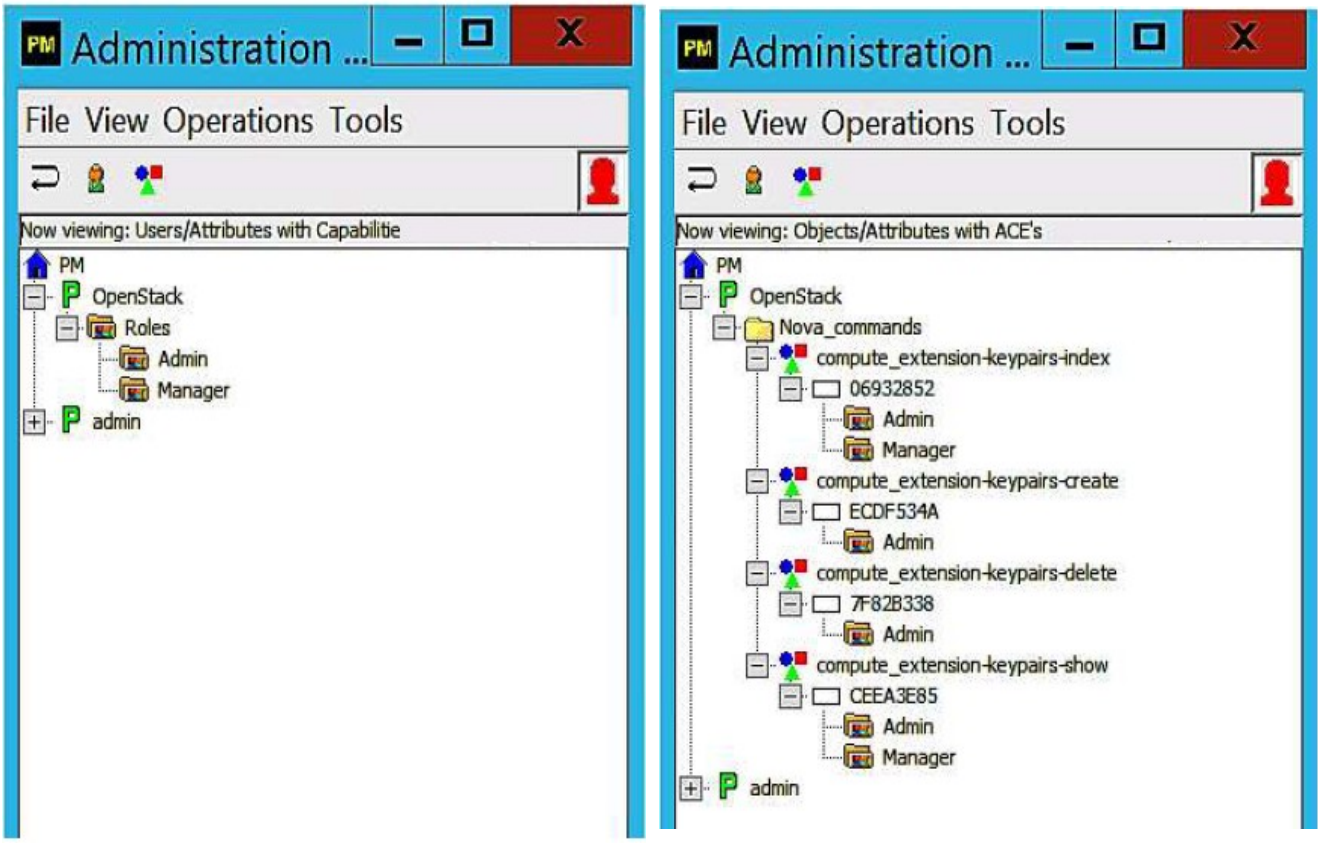

Figure 31: RBAC dans Openstack

Soit deux utilisateurs User1 et User2 dans Openstack. User1 est un Admin, tandis que User2 est Manager. Testons la politique illustré par la figure 31, pour s'assurer du bon fonctionnement de notre PM.

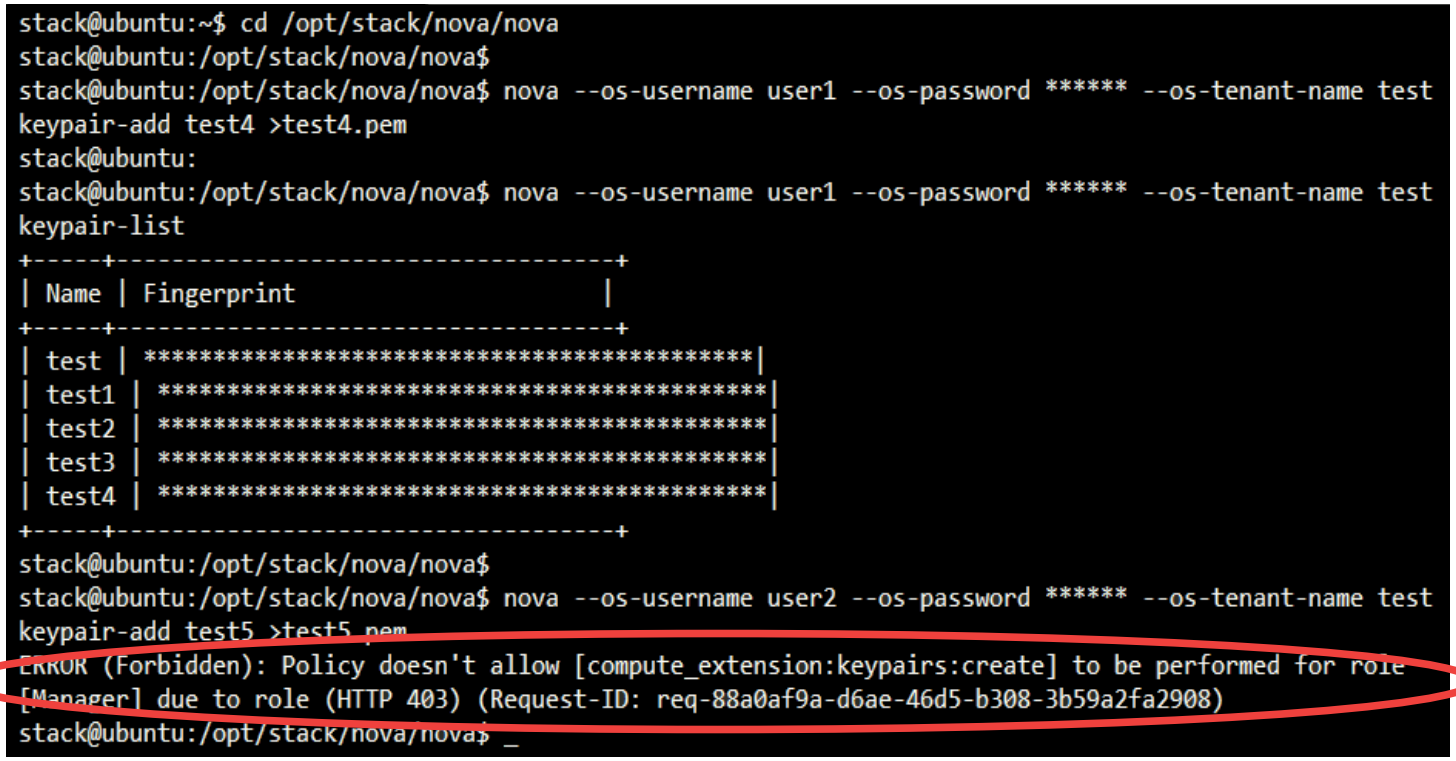

Figure 32: Simulation RBAC / Openstack

Comme dit auparavant, User2 possède le rôle *Manager*; il n'a pas alors le droit d'exécuter la commande *« keypair-add »*. Un message d'erreur apparaît : **Policy doesn't allow**.

On remarque qu'aucun message d'erreur n'apparaît dans le cas de User1: C'est un *Admin* : Il a tout les privilèges.

2-2-2 ABAC dans Openstack

Dans cette section, nous allons intégrer le modèle de contrôle d'accès dans l'environnement Openstack à l'aide de l'outil PM.

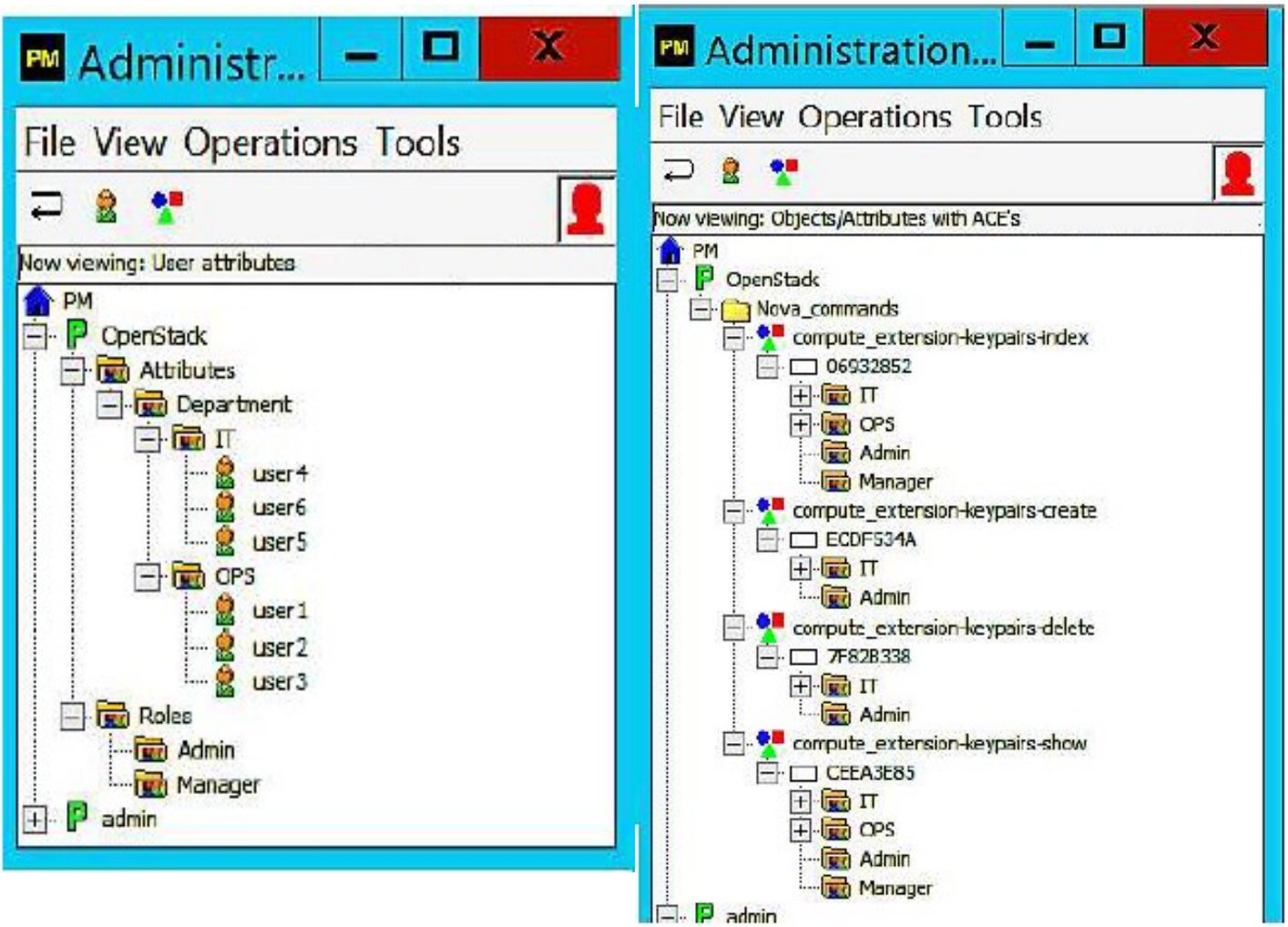

Figure 33: ABAC dans Openstack

La politique d'autorisation est définie en fonction des rôles et des attributs de l'utilisateur et est indiquée ci-dessous.

- *« compute_extension_keypair_create »* → (Role(u)=Admin **ET** Dep(u) =IT)
- *« compute_extension_keypair-delete »* → (Role(u)=Admin **ET** Dep(u) =IT)

- *« compute_extension_keypair-index »* → (Role(u)=Admin **OU** Role(u)=Manager) **ET** (Dep(u) = IT **OU** Dep(u) = OPS))

- *« compute_extension_keypair-show »* → (Role(u)=Admin **OU** Role(u)=Manager) **ET** (Dep(u) = IT **OU** Dep(u) = OPS))

Pour la première règle, un utilisateur U est autorisé à utiliser la commande *« compute_extension-keypair-create »* uniquement si l'utilisateur possède le rôle *Admin* **et** se trouve dans *le département IT*. Les deux conditions doivent être vraies. Sinon, l'accès sera refusé à un utilisateur sans le rôle spécifié ni la valeur de l'attribut utilisateur.

Dans la troisième règle, un utilisateur doit avoir un rôle *Admin* ou *Manager* avec une valeur d'attribut d'utilisateur IT ou OPS. Il s'agit d'une politique centrée sur les rôles, ce qui signifie que le rôle est d'abord vérifié, puis que la valeur de l'attribut utilisateur est vérifiée. Si la vérification d'un rôle échoue, l'accès est refusé et la valeur de l'attribut utilisateur n'est pas vérifiée.

La figure 33 illustre la spécification de cette politique dans l'ouil PM. Contrairement au cas d'utilisation précédent, il existe un attribut supplémentaire qui contient l'attribut d'utilisateur *Département* avec deux valeurs IT et OPS. Les associations apparaissent à droite et incluent à la fois l'attribut utilisateur et les rôles.

Du côté OpenStack, peu d'utilisateurs avec des rôles et des valeurs d'attribut d'utilisateur différents sont définis et leurs accès sont testés par rapport aux résultats souhaités. Par exemple, un utilisateur User1 est défini avec le rôle *Admin* et l'attribut *OPS*, et un autre utilisateur User4 est défini avec le rôle *Admin* et l'attribut *IT*. Dans ce cas, User4 peut exécuter la commande *« compute_extension-keypair-create »*, alors que User1 n'a pas le droit. La figure ci-dessous présente une capture d'écran des résultats sous OpenStack.

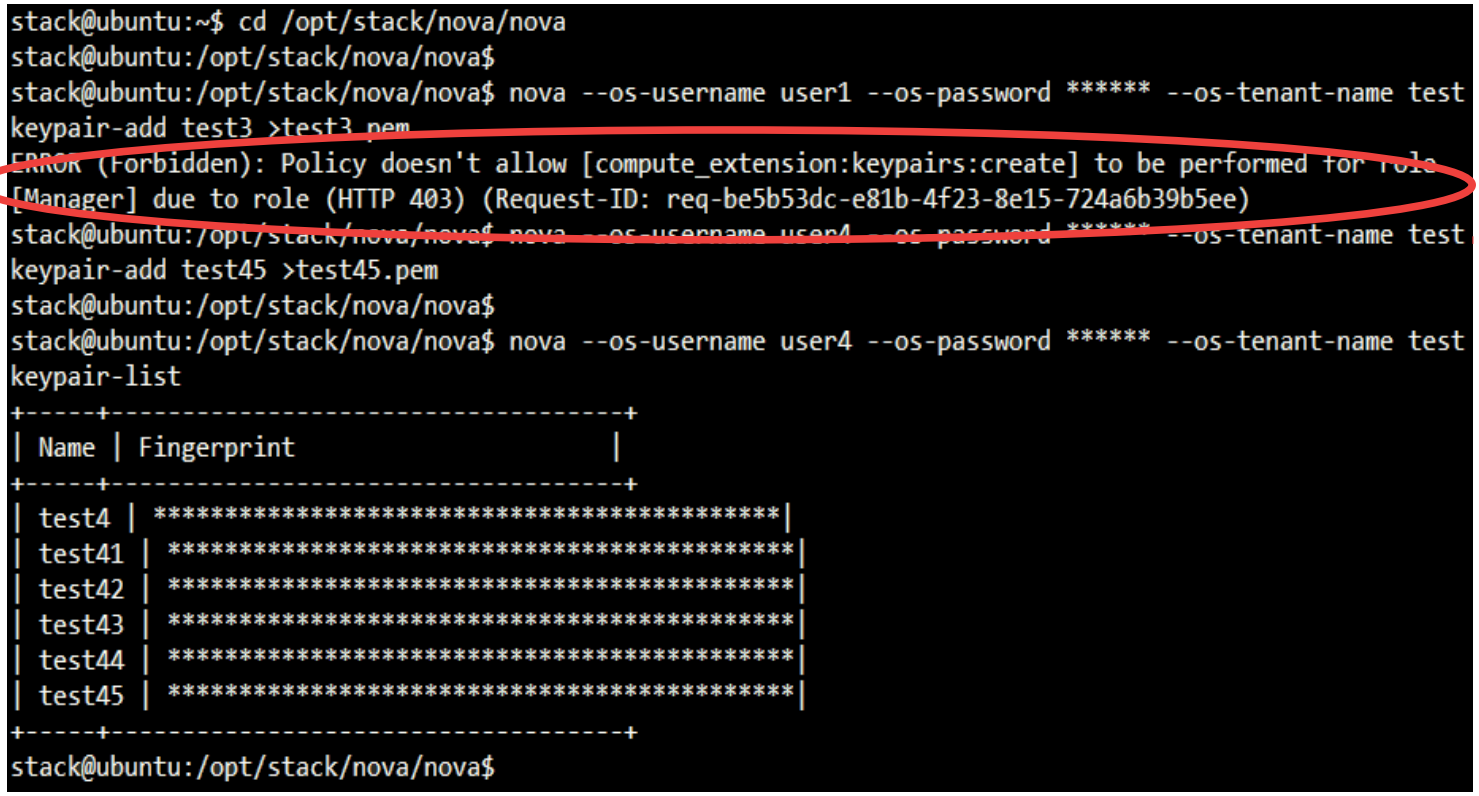

Figure 34: Simulation ABAC / Openstack

Conclusion

Lors de ce chapitre, nous avons vu le déploiement de la machine de politiques qui nous permettra de gérer l'accès aux ressources dans une instance Openstack en passant par un moteur d'évaluation.

Conclusion et perspectives

En raison de la nature ouverte des ***Clouds***, les utilisateurs souffrent d'une insuffisance au niveau des choix dans les méthodes de contrôle d'accès. Ainsi, la résolution de cette problématique était d'assurer un **maximum de flexibilité** au niveau des méthodes de contrôle d'accès.

Chaque organisme adopte un mécanisme de contrôle d'accès qui ,selon lui, convient parfaitement à ses besoins. Cependant, les Clouds en général offrent des choix limités de méthodes de contrôle d'accès, mais toutefois en cherchant à élargir la gamme de clientèle. Ceci ne peut être accompli que si **l'interopérabilité** est atteinte.

D'autre part, **la fédération de Clouds** est un principe qui consiste à faire collaborer une multitude de fournisseurs de Clouds afin d'optimiser d'avantage leurs ressources et offrir à leurs clients des services diversifiés. Ceci dit, l'interopérabilité des méthodes de contrôle d'accès est primordial pour parvenir à former ce consortium.

Durant ce rapport, nous avons réussi à résoudre la problématique **d'interopérabilité** en ayant recours à l'outil *« PM » Harmonia 1.5* qui nous a servi de point de relais entre les utilisateurs et les ressources. C'est un conteneur de politiques de contrôle d'accès qui pilote les demandes des utilisateurs , que nous avons adapté à nos besoins, voire faire **cohabiter** plusieurs **méthodes de contrôle d'accès** notamment **ABAC** et **RBAC**.

Comme perspectives, nous citons :

- Gestion des demandes d'utilisateurs au sein d'un environnement dynamique , ouvert et illimité ***(IOT)*** dans un Cloud.

- L’intégration d’un mécanisme de calcul de risque qui peut s’avérer plus efficace que le moteur d’autorisations de « PM » dans certaines situations où une flexibilité est primordial .

Bibliographie

[1] : Méthodes de contrôle d'accès : https://www.brighthub.com/computing/smbsecurity/articles/22839.aspx

[2] : Documentation Openstack : https://wiki.openstack.org/wiki/Documentation

[3] : A framework and risk assessment approaches for risk-based access control in the cloud. [Daniel Ricardodos Santos.] *[2016]*

[4] : Collaboration in Multicloud Computing Environments: Framework and Security Issues [*Mukesh Singhal and Santosh Chandrasekhar, University of California , Merced.] [2017]*

[5] : L'outil Harmonia 1.5 : https://github.com/PM-Master/Harmonia-1.5

[6] : L'outil Harmonia 1.5 : https://github.com/PM-Master/Harmonia-1.6

[7] : Guide Ubuntu Server : https://help.ubuntu.com/lts/serverguide/

[8] : Guide Windows server **:** http://econ.tu.ac.th/masteringwindows2003server.pdf

[9] : Keystone identity API v 2.0 : https://www.oreilly.com/library/view/identity-authentication-and/9781491941249/ch01.html

[10] : Emergence de l'IOT dans le cloud **:** https://www.networkworld.com/article/3229667/cloud-computing/how-will-the-cloud-be-able-to-handle-the-emergence-of-iot.html

[11] : Access control policies enforcement in a cloud environment: Openstack. [Meryeme Ayache,MohammedErradi, Bernd Freisleben] *[2015]*