POUR DES COMPÉTENCES TOUJOURS À LA POINTE

KeyStone et la sécurité selon OpenStack

Yvon Kermarrec

Professeur

Institut Mines Telecom / Télécom Bretagne

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Agenda du cours

- **Contexte, définitions et architecture**
  - Identité – authentification
  - Concepts et définitions
- **Opérations sur KeyStone**
- **Politiques de sécurité**
- **Des travaux pratiques**
- **Pour aller plus loin**
- **Synthèse**

Contexte et définitions

- **Keystone - le service de gestion des identités d'OpenStack – assure le contrôle sécurisé des ressources d'un cloud**
- **Keystone fournit des fonctions vitales, afin d'authentifier les utilisateurs et de déterminer quelles ressources ces utilisateurs peuvent accéder.**
- **Service majeur d'OpenStack puisque > 95% des utilisateurs d'OpenStack indiquent avoir déployé KeySTone dans leurs infrastuctures.**

Notion d'identité

- *"Identity refers to the identification of who is trying to access cloud resources. "*
- **Pour OpenStack Keystone, une identité est associée à un utilisateur.**
- **Dans les déploiements simples d'OpenStack, l'identité d'un utilisateur peut être stockée dans la base de donnée de KeyStone**
- **Dans des déploiements plus larges (au niveau d'entreprise), on peut vouloir utiliser un gestionnaire externe d'identités (ex IBM Tivoli)**

Notion d'authentification

- **Processus permettant de vérifier l'identité du demandeur (c'est-à-dire de vérifier qu'il est celui qu'il indique être)**
- **Initialement, cette opération est effectuée par un utilisateur qui indique lors du login son identité et son mot de passe.**
- **Après cette phase initiale, on peut vouloir optimiser ce processus afin de ne pas solliciter celui qui vérifie …. Et éviter de transmettre les informations de login sur le réseau.**

Gestion des accès et autorisations

- **L'autorisation est le processus qui permet de déterminer quelles ressources un utilisateur (avec une identité vérifiée) peut accéder et quelles opérations il peut exécuter.**
- **Cet aspect est fondamental pour gérer et contrôler les accès aux ressources et fonctionnalités du cloud (ex: qui peut créer ou détruire une image, qui peut rajouter un utilisateur, etc…)**

Pourquoi avoir intégré KeyStone dans OpenStack?

- La sécurité est vitale et critique dans le cloud
- Point d'accès unique et cohérent à tous les services d'OpenStack – avec la gestion des identités, l'authentification et les contrôles d'accès.
- Une organisation interne et une gestion des domaines, rôles, projets et utilisateurs
- Une notion de catalogue de services bien utile pour accéder aux différents services et fonctionnalités de l'infrastructure du cloud.
- Des services accessibles via une API de type RESTful

Concepts de KeyStone: le ‘projet’

- ***« a Project is an abstraction used by other OpenStack services to group and isolate resources (e.g., servers, images, etc.) »***
- **Dans les versions initiales de KeyStone, on utilisait la notion de ‘*Tenant*’**
- **KeyStone enregistre les différents projets et détermine qui peut y avoir accès**
- **Les projets eux-mêmes ne possèdent pas d'utilisateurs, mais les Utilisateurs ou Groupes d'Utilisateurs ont accès à un Projet via le rôle.**
- **Via le rôle qui lui est attribué, un utilisateur a accès (ou non) à tout ou partie des ressources du projet**

Concepts de KeyStone: le ‘domaine’

- **Le domaine correspond à une limite d’administration.**
- **“Domains are high-level containers for projects, users and groups. As such, they can be used to centrally manage all keystone-based identity components.”**

Concepts de KeyStone: ‘user’ et ‘group’

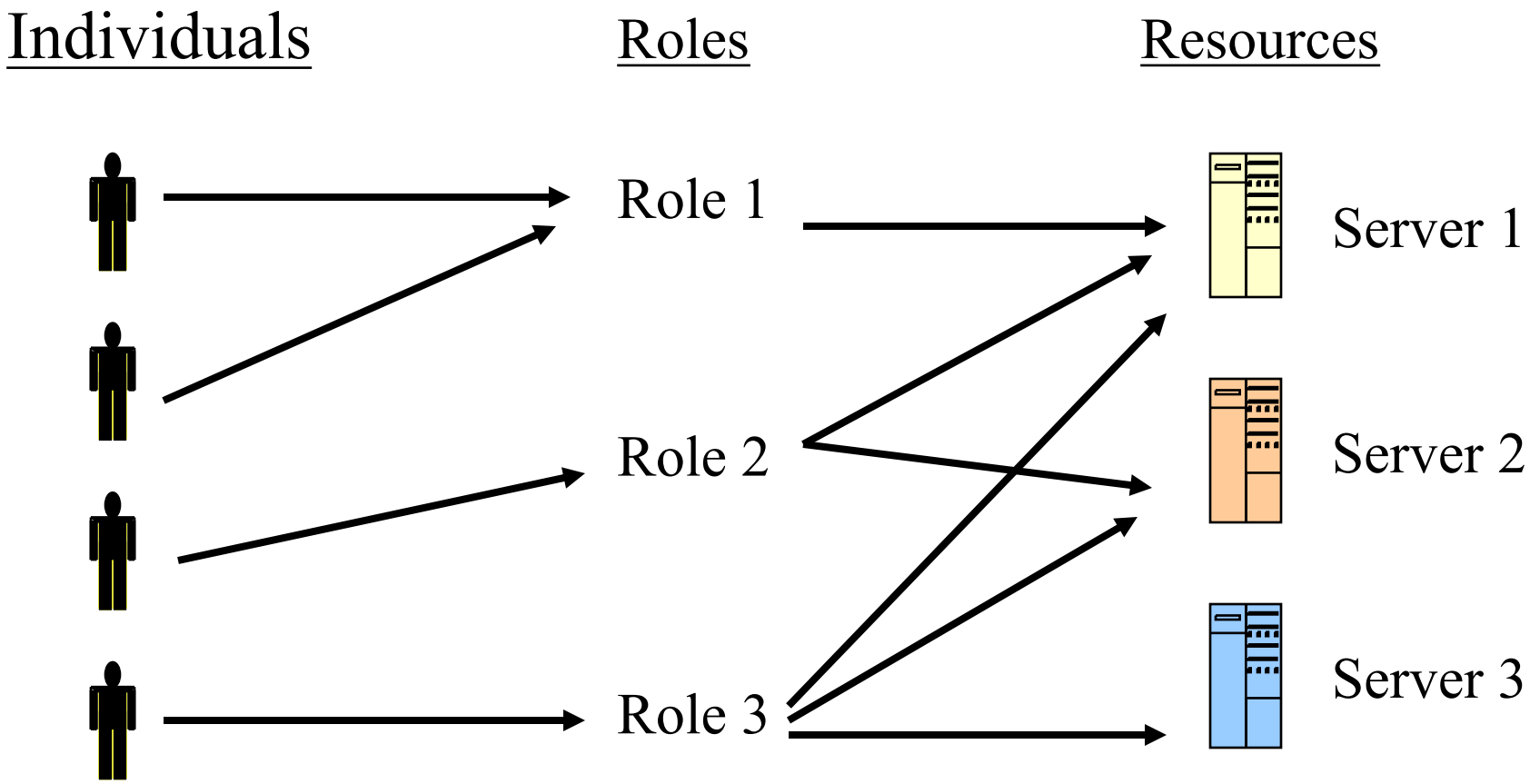
- Un ‘user’ est un individu qui utilisera le cloud (peut être un utilisateur, un système ou service) et qui dispose de « crédentials »
- Le groupe d’utilisateurs
- Un utilisateur est associé à un projet
- Les utilisateurs et groupes sont également désignés par ‘Actors’

Concepts de KeyStone: le rôle

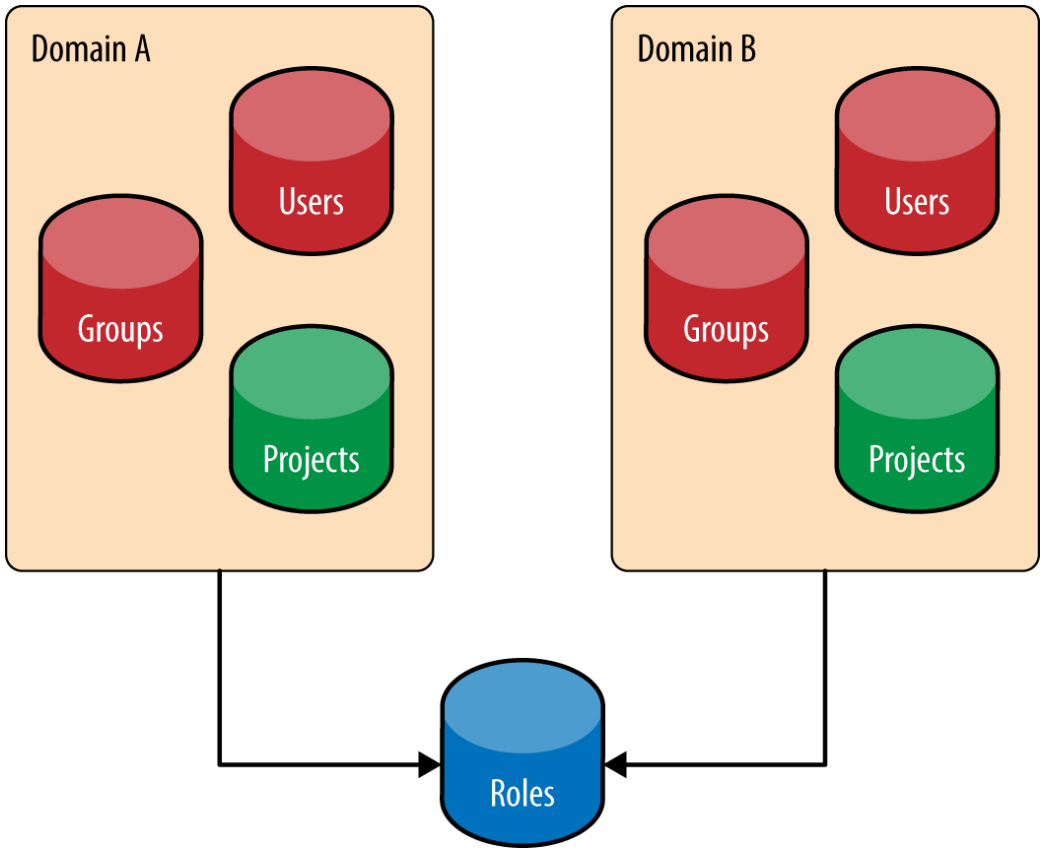
- **Élément de base pour l'expression de politiques de sécurité**
- **C'est une caractéristique (avec des privilèges associés) pour un utilisateur à un instant donné.**
- **L'utilisateur Bob prend le rôle 'rédacteur' pour le projet 'A'**
- **Un utilisateur peut avoir plusieurs rôles qui lui sont possibles (affectés) dans un projet – avec naturellement des privilèges différents.**

Concepts de KeyStone: le rôle

- **Les roles sont ‘stables’ (à la différence des utilisateurs)**



Concepts de KeyStone



Concepts de KeyStone

- **Token**
  - Un identifiant des droits associés à un utilisateur (dans le contexte ou non d'un projet)
  - Notion de *scoped token* (lié à un projet) et *unscoped token* (non lié à un projet)
- **Policy – règles**
  - Expression sous une forme de garde / condition pour l'accès à une ressource ou service
- **Endpoint**
  - Point d'entrée vers un service (avec une interface publique)

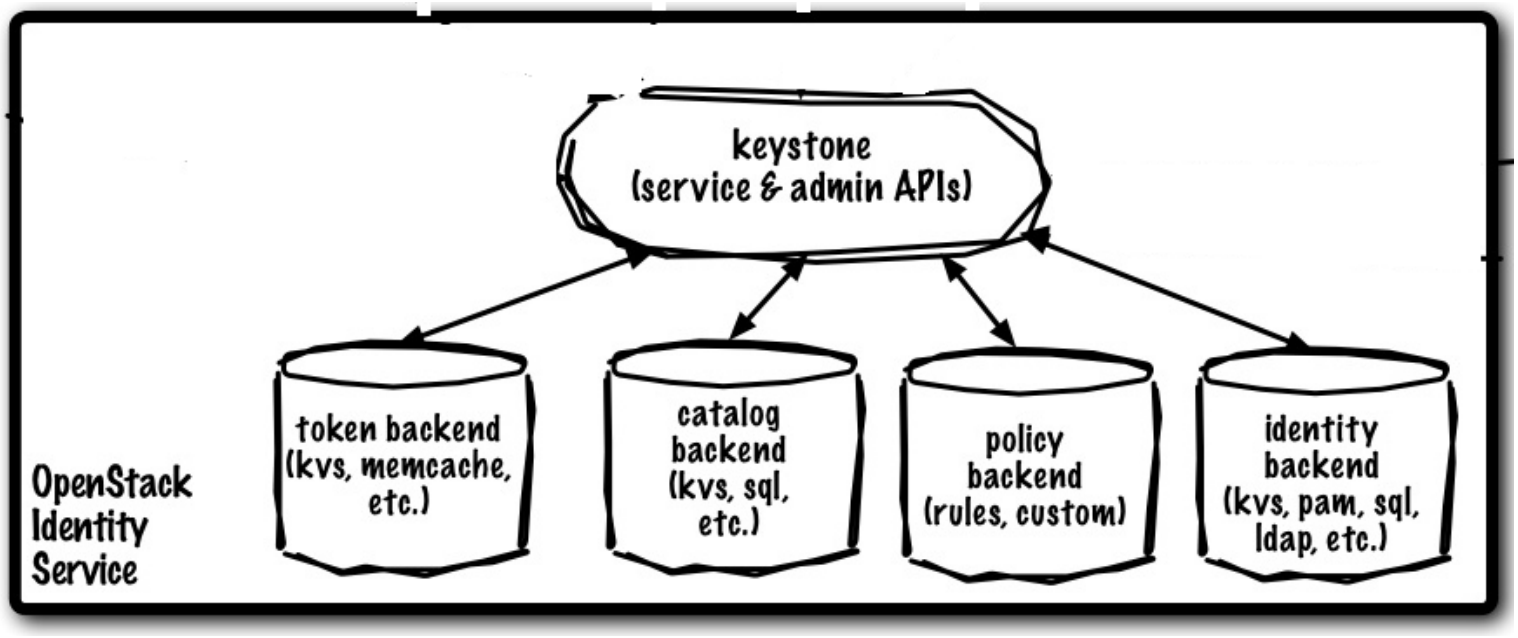
Agenda du cours

- **Contexte et définitions**
- **Opérations sur KeyStone**
  - La gestion des identités
  - Le token dans le détail
  - Commandes d'administration
- **Politiques de sécurité**
- **Des travaux pratiques**
- **Pour aller plus loin**
- **Synthèse**

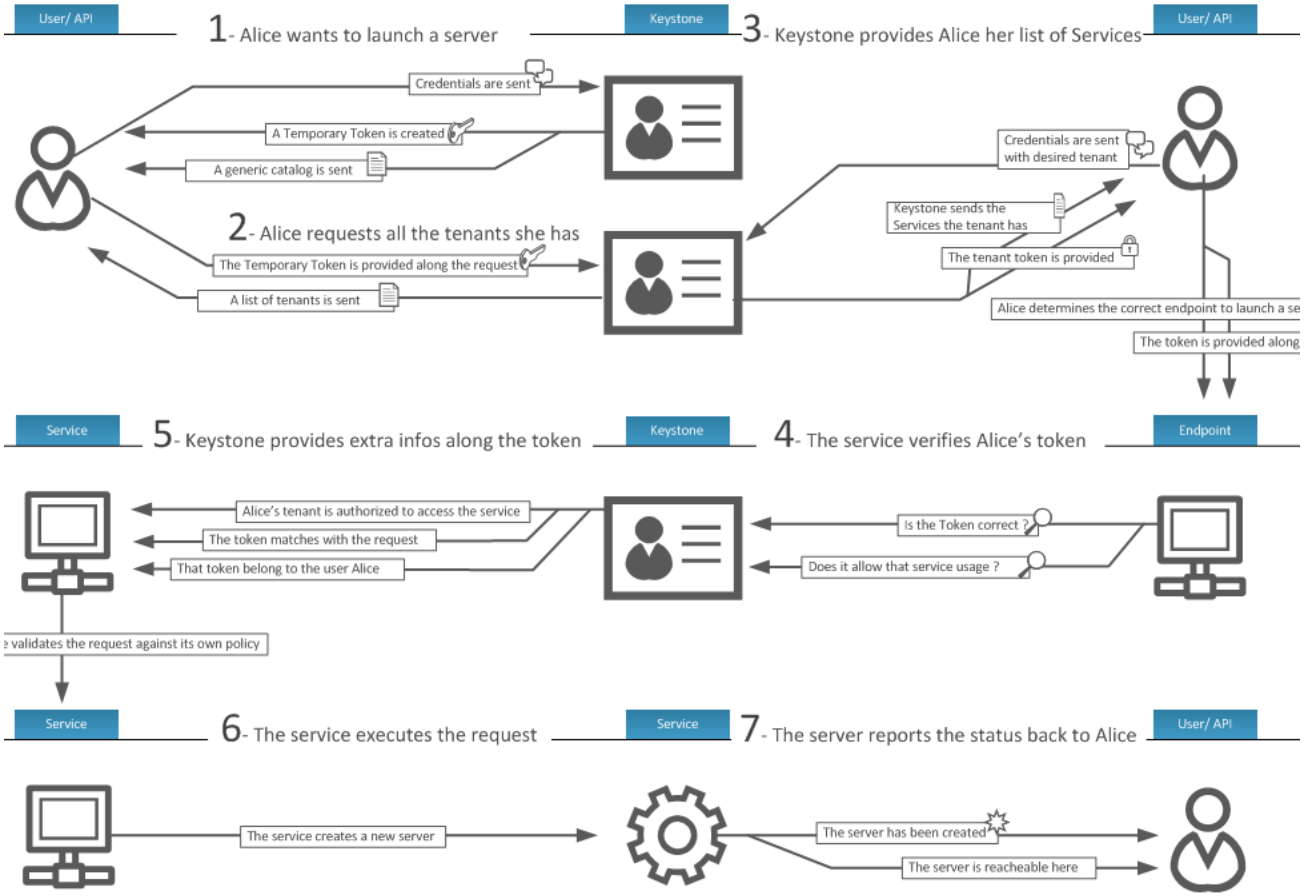
Les services internes de KeyStone

- Token (*Token back-end*) : valide et gère les jetons utilisés lors des requètes
- Catalogue (*catalog back-end*) : gère les différents points d'entrée (URL) permettant l'accès à un service
- Policy (*policy back-end*) : c'est le gestionnaire / moteur des règles et politiques de sécurité
- Identity (*identity back-end*): gère la validation des droits pour les utilisateurs, les groupes, les projets, les domaines, les rôles, …

Architecture de KeySTone

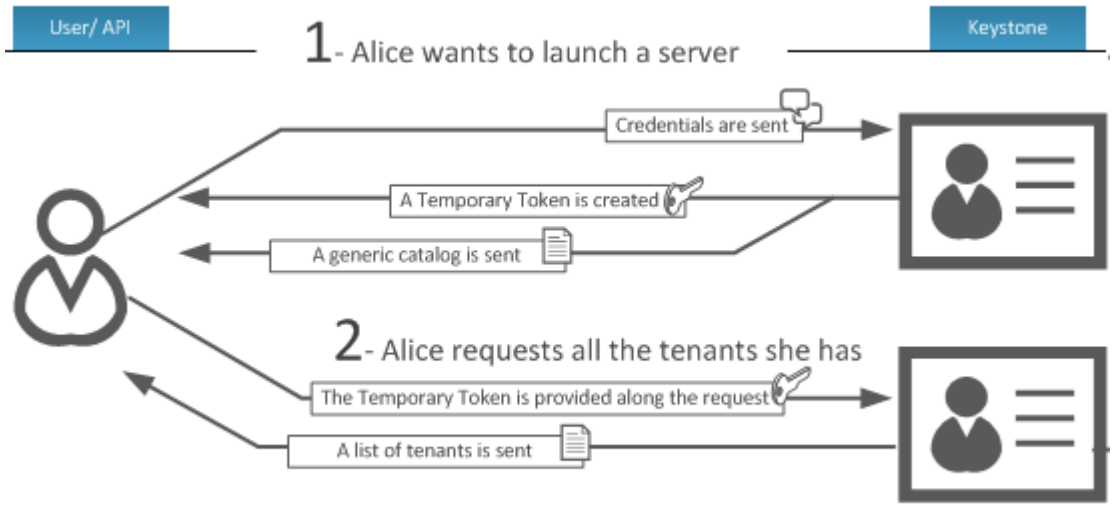


Keystone identity manager (1/6)



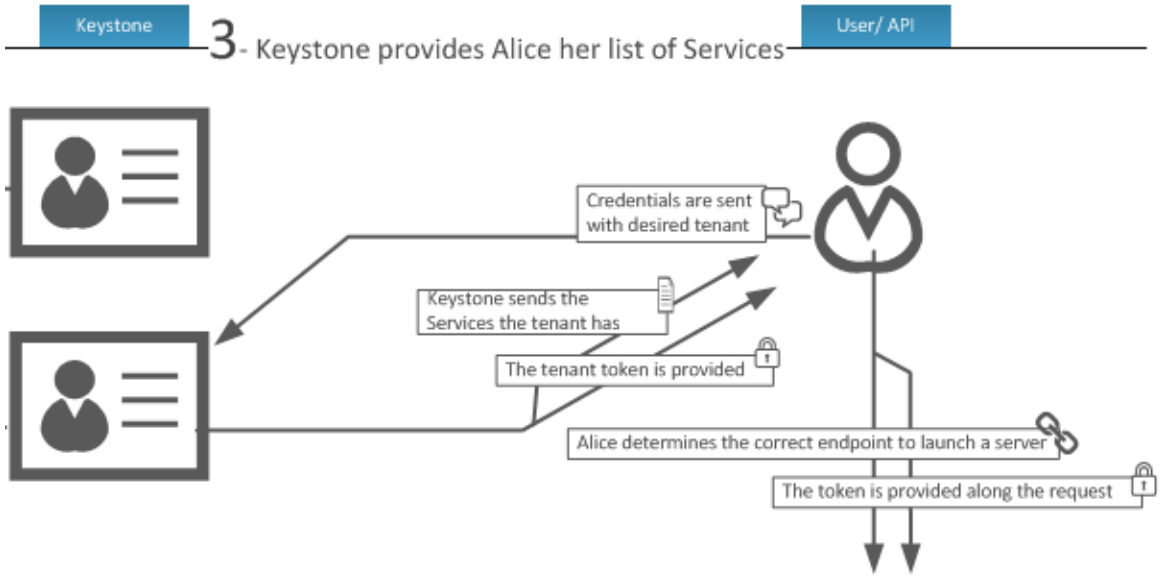
Keystone identity manager (2/6)

- Un utilisateur demande un token (unscoped) en précisant ses infos de login : id et mot de passe
- KeyStone lui renvoie un jeton
- L'utilisateur demande quels sont ses projets (« tenants »)



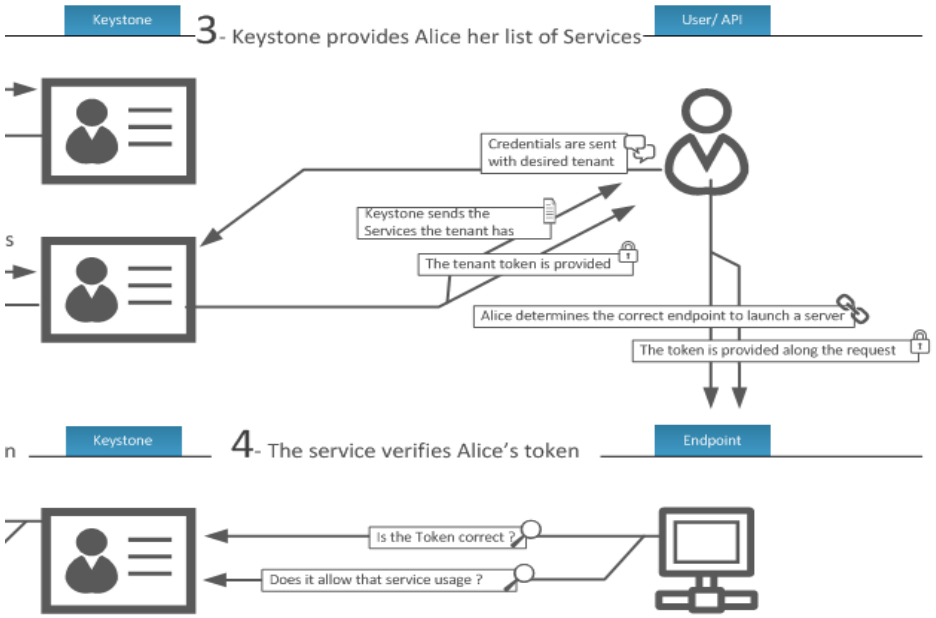
Keystone identity manager (3/6)

- **L'utilisateur précise ensuite le projet et Keystone lui renvoie un token (scoped) et une liste de points d'entrées**

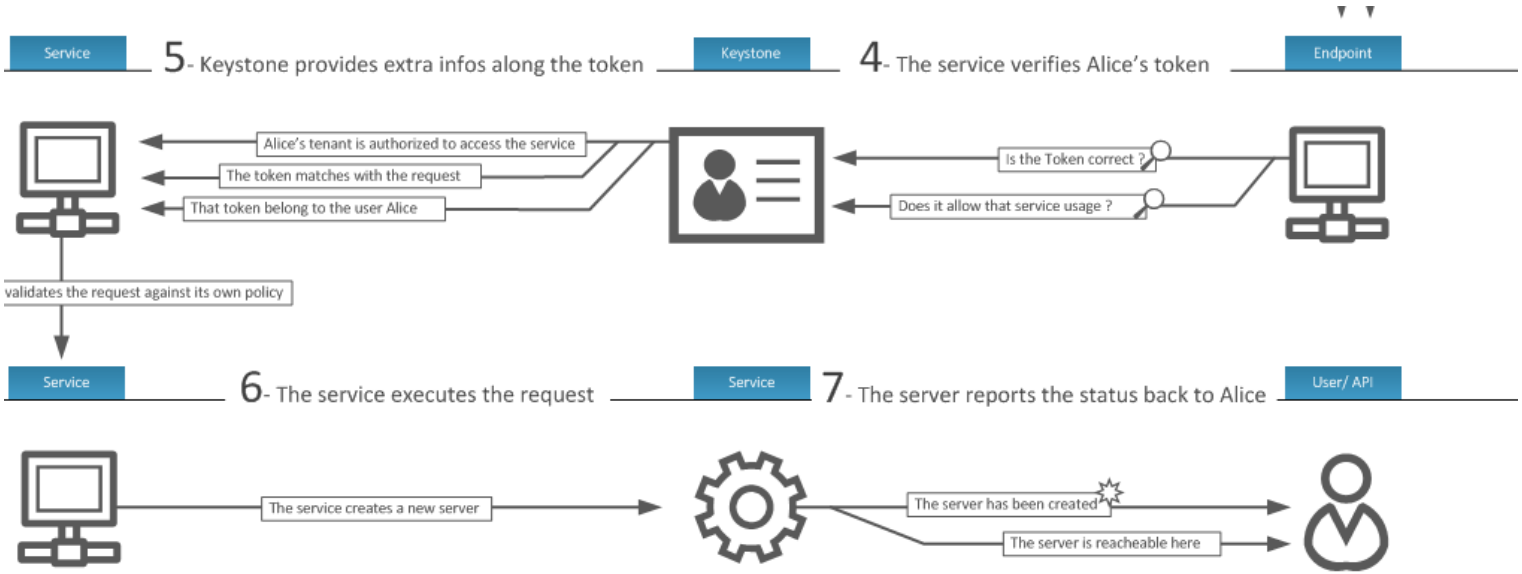


Keystone identity manager (4/6)

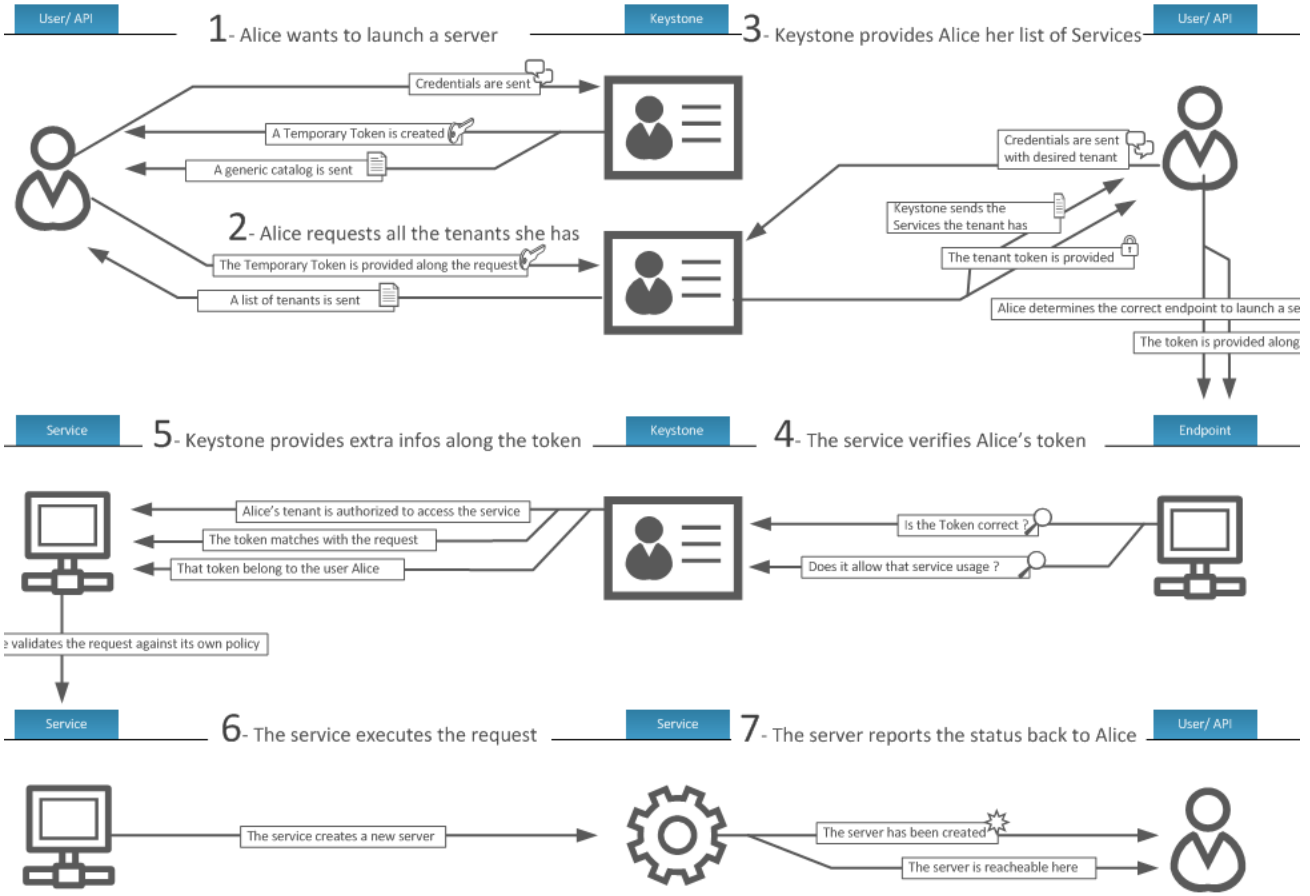
- **Appel et vérification des droits pour l'opération demandée**



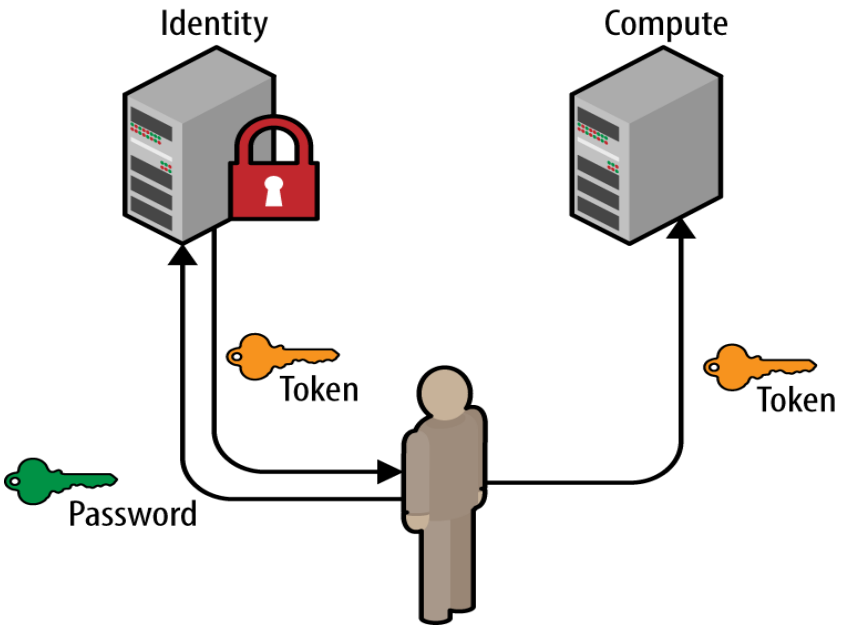
Keystone identity manager (5/6)

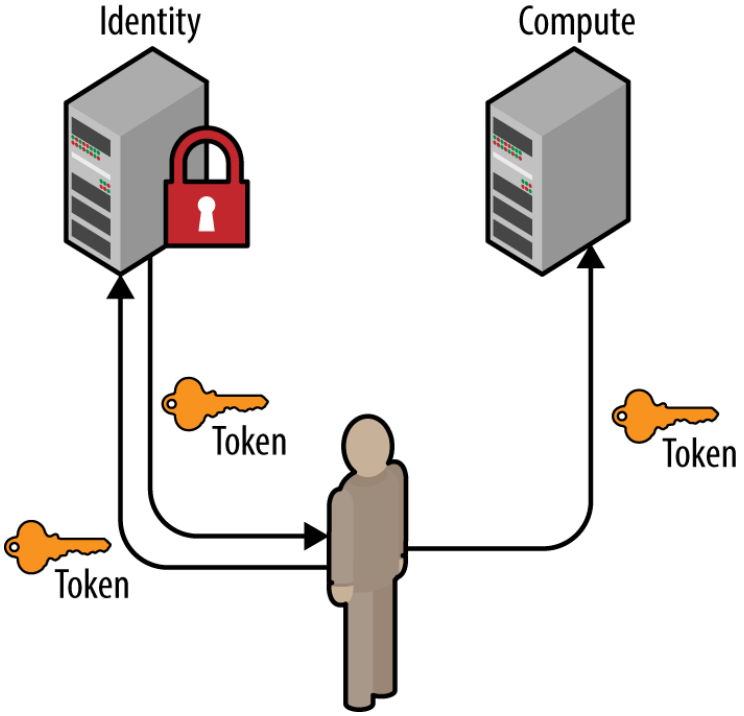


Keystone identity manager (6/6)



Le Token : principe d'obtention : avec login ou un autre token





Le jeton

- **Pour qu'un utilisateur puisse appeler une API OpenStack, il doit**
  - prouver qui il est, et
  - qu'il peut invoquer l'API en question.
- **Pour cela, l'utilisateur passe un jeton OpenStack lors de l'appel de l'API.**
- **Un utilisateur reçoit ce jeton lors d'une authentification réussie avec Keystone.**
- **Le jeton contient l'autorisation qu'un utilisateur a sur le cloud.**

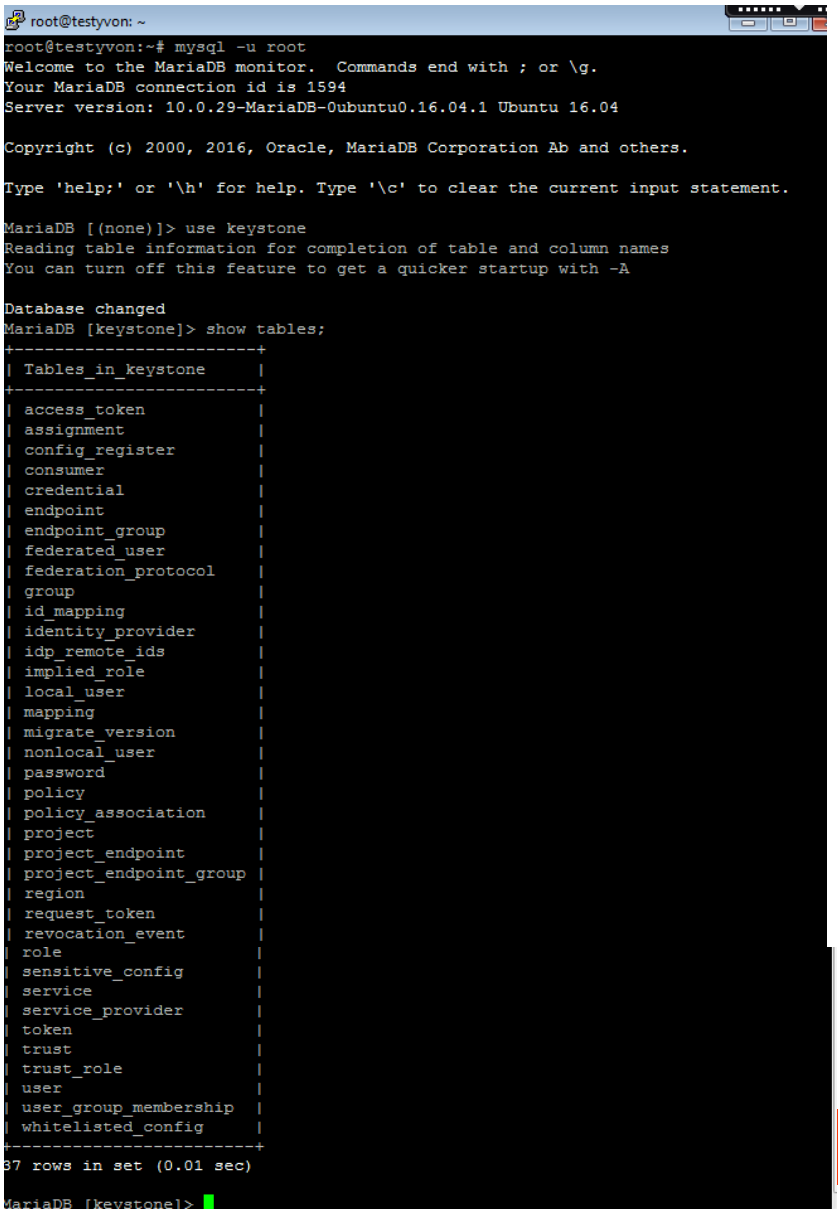
Le jeton

- **Un jeton possède à la fois un ID et une charge utile. L'ID d'un jeton est garantie unique par nuage.**
- **Le jeton peut être de plusieurs types:**
  - UUID – 32 octets
  - Fernet (qui se termine)
  - PKI (taille > 1 kO)
  - PKIZ

Le contenu (payload) d'un token


```
{
    "token": {
        "issued_at": "201406-10T20:55:16.806027Z",
        "expires_at": "2014-06-10T2:55:16.806001Z",
        "roles": [{
                "id": "c703057be878458588961ce9a0ce686b",
                "name": "admin"}
        ],
        "project": {
            "domain": { "id": "default",
                        "name": "Default" },
            "id": "8538a3f13f9541b28c2620eb19065e45",
            "name": "admin"
        },
        "user": {
            "domain": { "id": "default",
                        "name": "Default" },
            "id": "3ec3164f750146be97f21559ee4d9c51",
            "name": "admin"
        },
        "catalog": [
            {
                "endpoints": [...],
                "type": "identity",
                "id": "bd73972c0e14fb69bae8ff76e112a90",
                "name": "keystone"
            }
        ]
    }
}
```

KeyStone database : mysql inside



Catalogue de service

```
"catalog": [
    {
        "name": "Keystone",
        "type": "identity",
        "endpoints": [
            {
                "interface": "public",
                "url": "https://identity.example.com:35357/"
            }
        ]
    }
]
```

Comment appeler un service ?

- **Connaitre son endpoint!**
- **Utiliser les commandes préfixées par `openstack`**
  - https://docs.openstack.org/user-guide/cli-cheat-sheet.html
  - Pensez à initialiser les variables d'environnement (ex nom du domaine, du projet, user id, ….)
- **Appeler le endpoint en direct avec une commande de type curl**
  - c'est pas très pratique….mais très explicite

Quelques commandes utiles (vues en TP)

- **Liste des utilisateurs :** `openstack user list`
- **Création d'un token :** `openstack token issue`
- **Révocation d'un token:** `openstack revoke token id`
- **Liste des projets:** `openstack project list`
- **Liste des groupes :** `openstack group list`
- **Liste des rôles :** `openstack role list`

Authentification par mot de passe

POST https://api.keystone.cloud/v3/auth/tokens

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "daneel",
                    "domain": {
                        "id": "default"
                    },
                    "password": "r0b0t!!"
                }
            }
        }
    }
}
```

Authentification par jeton

POST https://api.keystone.cloud/v3/auth/tokens

```
{
    "auth": {
        "identity": {
            "methods": [
                "token"
            ],
            "token": {
                "id": "110e8400-e29b-11d4-a716-446655440000"
            }
        }
    }
}
```

Demande de jeton avec scope

POST https://api.keystone.cloud/v3/auth/tokens

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "id": "ee4dfb6e5540447cb3741905149d9b6e",
                    "password": "r0b0t!!"
                }
            }
        },
        "scope": {
            "project": {
                "id": "a6944d763bf64ee6a275f1263fae0352"
            }
        }
    }
}
```

Réponse à la demande de jeton


```
{
    "token": {
        "expires_at": "2016-10-17T21:29:55.118796Z",
        "issued_at": "2016-10-17T20:29:55.118829Z",
        "methods": [ "password" ],
        "project": {
            "domain": { "id": "default", "name": "Default" },
            "id": "8ddce4acebcc44b8b4f4cc3142ddafaa",
            "name": "vapor"
        },
        "roles": [
            { "id": "9250b2448e2942499734c1c7029f2f18",
              "name": "manager" }
        ],
        "user": {
            "domain": { "id": "default", "name": "Default" },
            "id": "ec80ca1e609245b9bc082e422661a967",
            "name": "daneel"
        }
    }
}
```

Agenda du cours

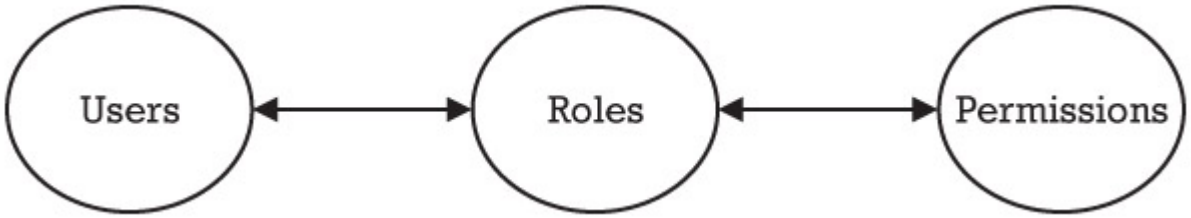
- **Contexte, définitions et architecture**
- **Opérations sur KeyStone**
- **Politiques de sécurité**
  - RBAC : Role-Based Access Control
  - Pour KeyStone
- **Des travaux pratiques**
- **Pour aller plus loin**
- **Synthèse**

RBAC

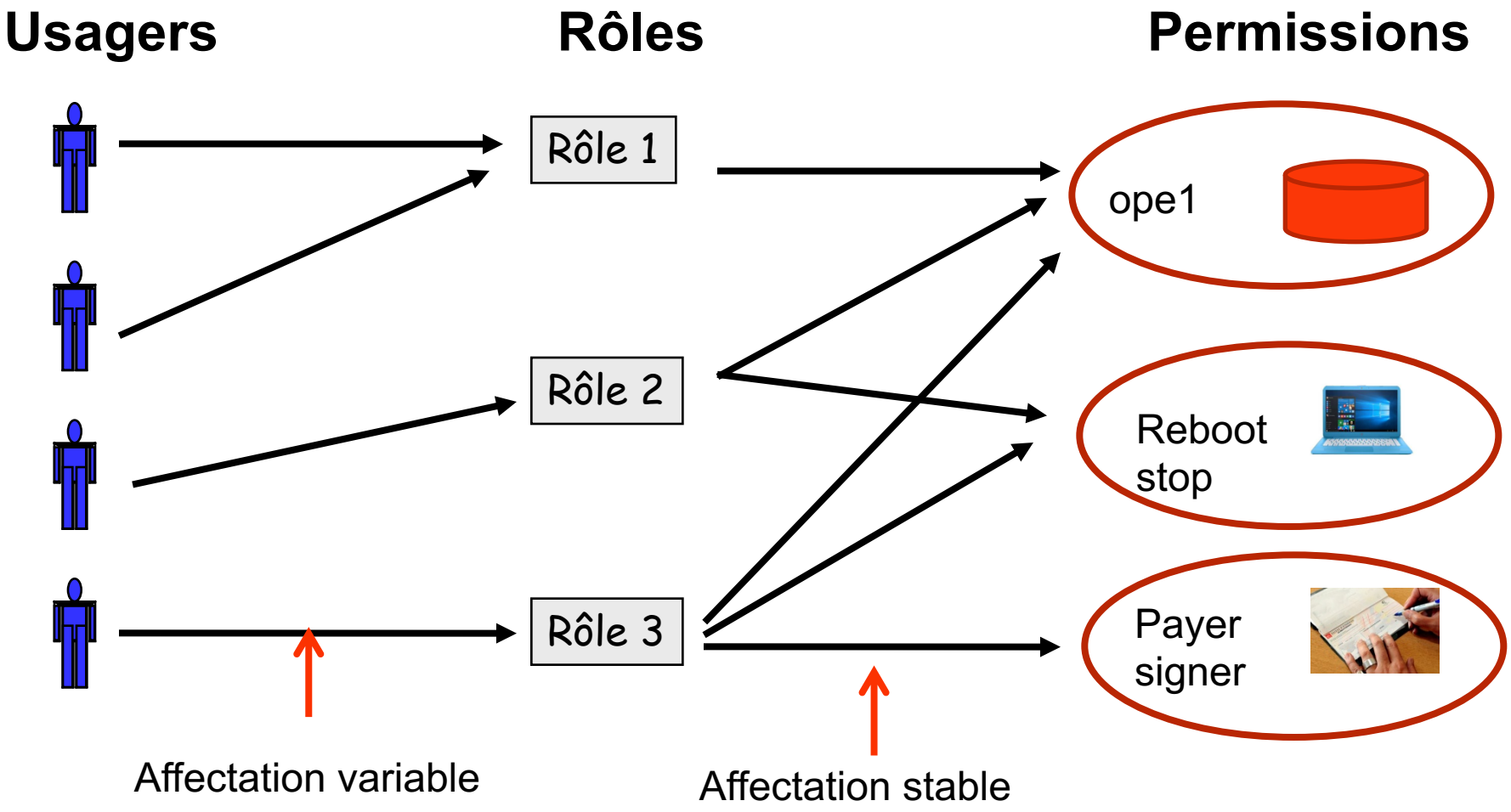
- **RBAC est basé sur deux constats:**
  - dans les organisations, les employés sont classés par rôles ou métiers
    - Chef, secrétaire général, directeur, analyse, programmeur
    - Les rôles sont organisés en **hiérarchies**
  - chaque employé, pour exécuter son rôle, a besoin de certaines ‘permissions’ ou ‘autorisations’

RBAC

- **RBAC profite de cette notion d'organisation (et de hiérarchie) de rôle pour associer des permissions de sécurité aux différents rôles**
- **Le rôle devient alors un mécanisme pour associer des permissions aux utilisateurs**



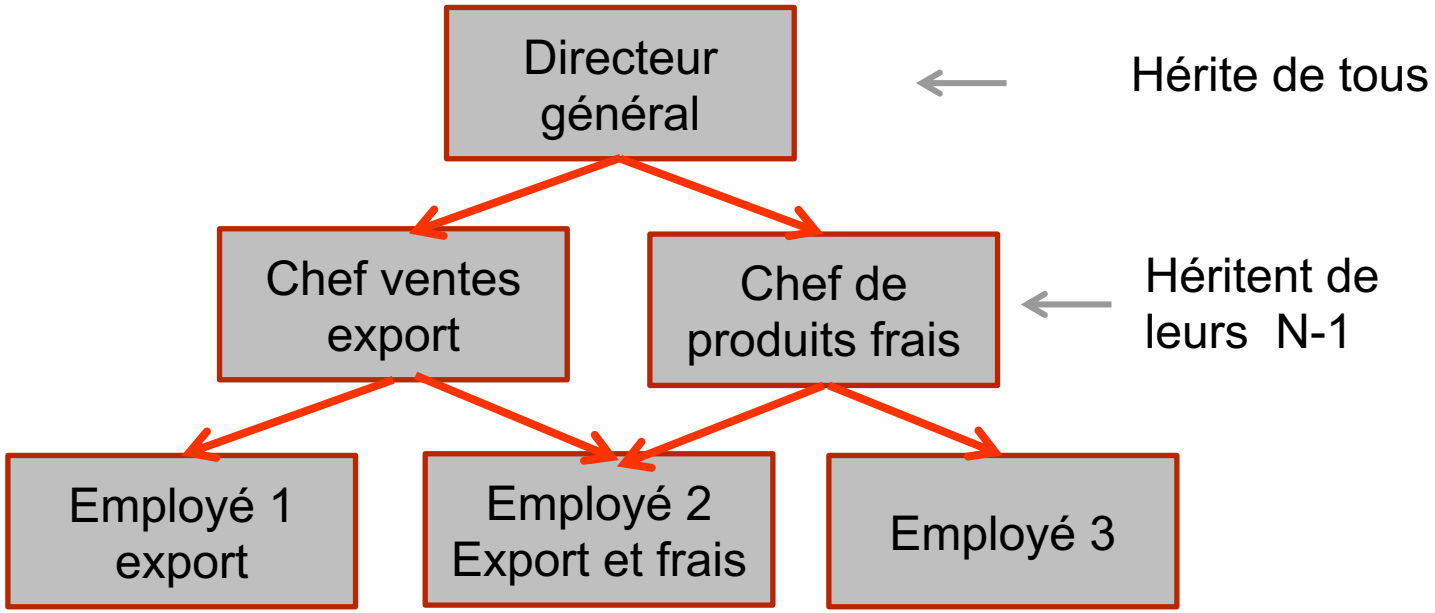
RBAC: modèle conceptuel



RBAC et cohérence

- La politique de sécurité est critique car une erreur peut avoir des conséquences majeures
- Les règles doivent dont être exprimées avec soin
- La modification du fichier policy.json implique une mise à jour immédiate
- Par défaut, toute opération sur une API est rejetée

RBAC et hiérarchie de rôles



Exprimer une politique de sécurité (1/5)

- Tous les services d'OpenStack peuvent être protégés par une stratégie (*policy*) de sécurité.
- Elle est exprimée dans un formalise ad'hoc et stockée dans un fichier json (Javascript Object Notation).
- Chaque API Keystone possède une ligne dans le fichier de politique qui dicte le niveau de protection qui lui est appliqué
- Il y a donc un langage d'expression de règles (https://docs.openstack.org/kilo/config-reference/content/policy-json-file.html) mais la syntaxe / sémantique me semblent encore non finalisées.

Exprimer une politique de sécurité (2/5)

- **Le fichier est de type texte et est accédé lors de chaque appel d'une API de service**
- **A manier avec beacoup de soin !!!**
- **Chaque ligne exprime une règle de sécurité sous la forme : `<target> : <rule>`**
  - Target: appelée aussi « action » et associée à un appel d'une API de service – ex: "start an instance";"attach a volume"
  - Target est classiquement un nim qualifié : cad nom du service ":" nom de l'API
  - Rule : exprime la condition / circonstance pour que l'appel de l'API soit validé
  - "The mapping between API calls and actions is not generally documented."

Exprimer une politique de sécurité (3/5)

- **`"compute:get_all" : " "`**
  - La section rule de cette règle est vide et la condition est donc vraie. L'appel à cet API est toujours possible.
- **`"compute:shelve": "! "`**
  - La section rule de cette règle est « ! » et la condition est donc fausse. L'appel à cet API est toujours impossible.
- **`"identity:create_user" : "role:admin "`**
  - Pour appeler l'API 'create-user' du service identity il faut que le rôle de l'appelant soit admin
- **`"stacks:create": "not role:heat_stack_user"`**
  - On peut mettre des expressions booléennes…. Et donc combiner des conditions dans la partie 'rule'

Exprimer une politique de sécurité (4/5)

- **`"compute:start" : "user_id:%(user_id)s"`**
  - Seul le propriétaire d'une instance peut l'activer. Le 'user-id' avant le ':' designe l'identité de celui qui appelle l'API. « %(user_id)s » désigne l'identité de l'objet sur lequel l'API est demandé – ici le propriétaire de l'instance.
- **`"identity:delete_user": "role:admin and domain_id:%(target.user.domain_id) "`**
  - Seul un administrateur du domaine peut retirer un utilisateur
- **`"admin_required": "role:admin or is_admin:True "`**
  - Ici on définit un alias qui peut donc apparaitre dans la suite dans une partie 'rule'

Exprimer une politique de sécurité (5/5)

- **Un exemple qui utilise des alias et qui permet de modifier le mot de passe d'une entité uniquement si l'appelant est 'admin' ou s'il est le propriétaire de cet objet**

```
"admin_required": "role:admin or is_admin:1",
"owner" : "user_id:%(user_id)s",
"admin_or_owner": "rule:admin_required or rule:owner",
"identity:change_password": "rule:admin_or_owner"
```

Agenda du cours

- **Contexte, définitions et architecture**
- **Opérations sur KeyStone**
- **Politiques de sécurité**
  - RBAC : Role-Based Access Control
  - Pour KeyStone
- **Des travaux pratiques**
- **Pour aller plus loin**
- **Synthèse**

Documents techniques

- **La documentation OpenStack sur les policies**
  - https://docs.openstack.org/kilo/config-reference/content/policy-json-file.html
- **Le wiki associé**
  - https://docs.openstack.org/developer/keystone/configuration.html#api-protection-with-role-based-access-control-rbac
- **Un livre sur le sujet**
  - « Identity, Authentication, and Access Management in OpenStack » par *Steve Martinelli, Henry Nash, and Brad Topol, publié par O'Reilly*

Agenda du cours

- **Contexte, définitions et architecture**
- **Opérations sur KeyStone**
- **Politiques de sécurité**
  - RBAC : Role-Based Access Control
  - Pour KeyStone
- **Des travaux pratiques**
- **Pour aller plus loin**
- **Synthèse**

Synthèse

- KeyStone est comme son nom l'indique la clé de voute d'une architecture OpenStack
- KeyStone gère les identités et fournit des mécanismes d'authentification de confiance
- Des politiques de sécurité à base du modèle RBAC permet de préciser finement les règles d'accès aux API des services