Travaux pratiques avec KeyStone d'OpenStack

Yvon Kermarrec

Institut Mines Télécom / Télécom Bretagne

Objectifs du TP:

- utiliser l'interface graphique pour se connecter via Horizon
- exécuter des commandes de base avec KeyStone (le gestionnaire d'identité d'OpenStack).
- dérouler un cas d'étude complet avec création de rôles et de politiques de sécurité.

00 - Précisions sur mon environnement en cours

Il convient naturellement d'adapter à votre contexte. Voici celui que j'utilise pour exécuter ce TP et extraire les copies d'écran.

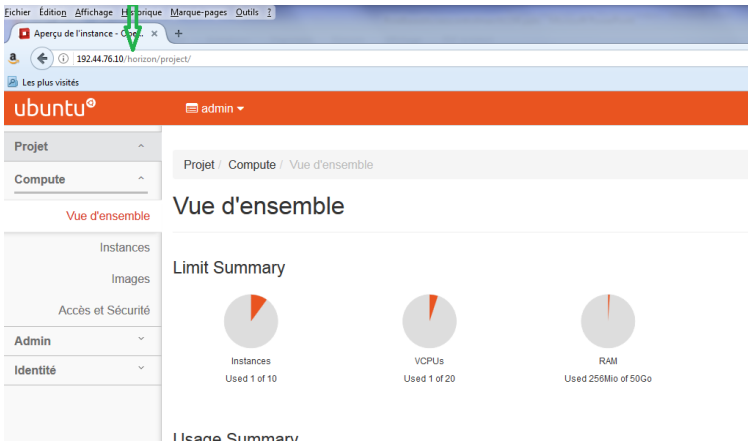
- numéro IP: 192.44.76.10
- un endpoint de gestion via Horizon: http://192.44.76.10/horizon/project/

0 - Initialisation de l'environnement

L'appel à OpenStack est assez complexe lors de sa première utilisation: il faut configurer des variables d'environnement, connaitre le nom du domaine ou du projet que l'on veut utiliser.... Bref, c'est pas simple si on ne part de rien.

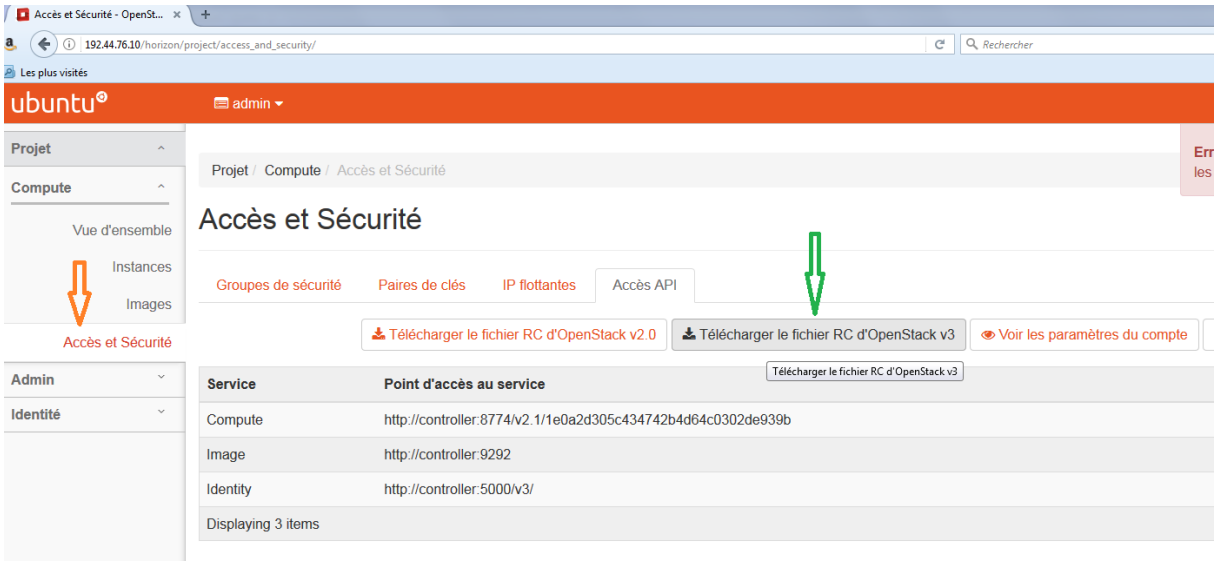
Heureusement, l'interface graphique propose une solution pour générer le fichier de configuration. Un fichier de type RC ("Run Command") directement accessible depuis l'interface Horizon.

Connectez vous sur l'interface de gestion avec l'URL ad'hoc



Puis dans le menu Compute/"accès et sécurité", sélectionnez le téléchargement du fichier RC d'OpenStack en version 3 (flèche verte).

Sauvegardez ce fichier dans votre 'home' et exécutez le avec la commande Unix `source`



1 - commandes de base

1.1 - Initialiser et vérifier que tout est bien configuré

c'est important et pour cela nous allons générer un jeton - voir ci-après.

1.2 - Obtenir un jeton avec données du login

Pour cela, nous utilisons la commande `openstack token issue`

```
ubuntu@testyvon:~$
ubuntu@testyvon:~$ source admin-rc
ubuntu@testyvon:~$ openstack token issue
+------------+----------------------------------+
| Field      | Value                            |
+------------+----------------------------------+
| expires    | 2017-02-23 13:13:03+00:00        |
| id         | 075e7977da7344e09c222ee68aed8052 |
| project_id | 1e0a2d305c434742b4d64c0302de939b |
| user_id    | 03f2ba67d72443c2942626286a6a8664 |
+------------+----------------------------------+
ubuntu@testyvon:~$ more c*2
```

1.3 - Obtenir un jeton à partir d'un login et de curl

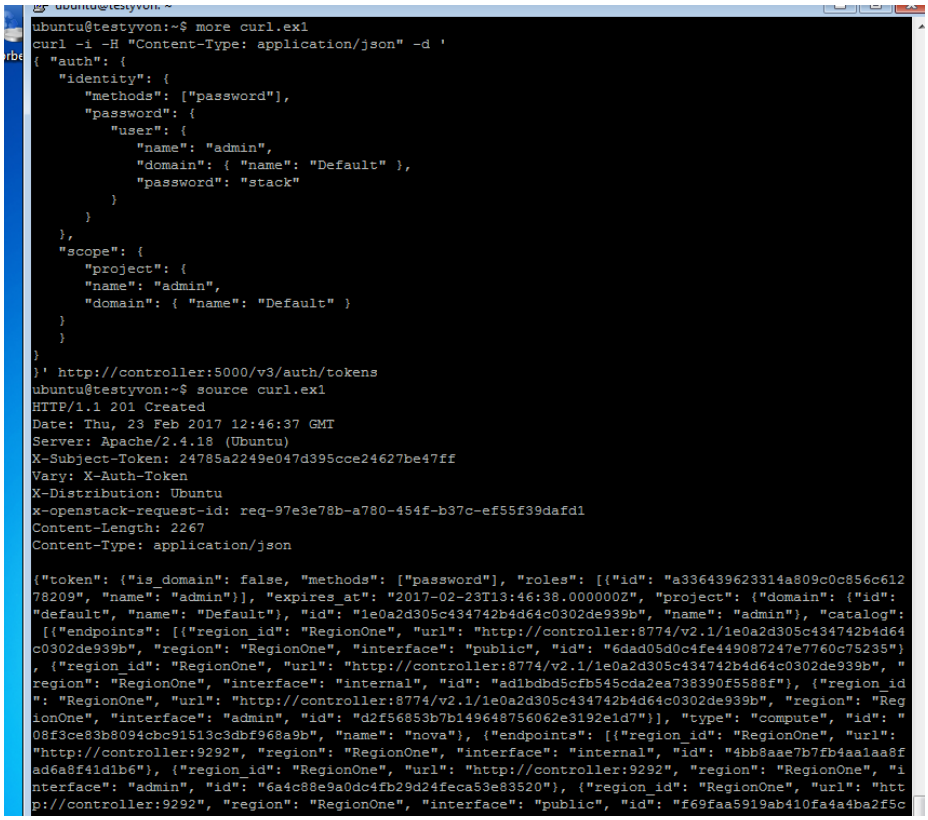
```
ubuntu@testyvon:~$
ubuntu@testyvon:~$ more curl.ex1sscope
curl -i -H "Content-Type: application/json" -d '
{ "auth": {
   "identity": {
      "methods": ["password"],
      "password": {
         "user": {
            "name": "admin",
            "domain": { "name": "Default" },
            "password": "stack"
         }
      }
   }
}
}' http://controller:5000/v3/auth/tokens
ubuntu@testyvon:~$ source curl.ex1sscope
HTTP/1.1 201 Created
Date: Thu, 23 Feb 2017 12:44:17 GMT
Server: Apache/2.4.18 (Ubuntu)
X-Subject-Token: d3df2f201f084499b6548775b45ee85d
Vary: X-Auth-Token
X-Distribution: Ubuntu
x-openstack-request-id: req-8ddab986-ff7d-40cc-9b1e-e8433960900b
Content-Length: 283
Content-Type: application/json

{"token": {"issued_at": "2017-02-23T12:44:17.000000Z", "audit_ids": ["c41FTCDYTU63coX
9nQcnTQ"], "methods": ["password"], "expires_at": "2017-02-23T13:44:17.000000Z", "use
r": {"domain": {"id": "default", "name": "Default"}, "id": "03f2ba67d72443c2942626286
a6a8664", "name": "admin"}}}ubuntu@testyvon:~$
```

1.4 - Obtenir un jeton avec scope (scoped token) à partir d'un login et de curl

ici nous précisons dans la commande, le scope désiré pour le token. La commande renvoie beaucoup plus d'informations puisque nous récupérons également les informations sur les endpoints que nous pouvons appeler.

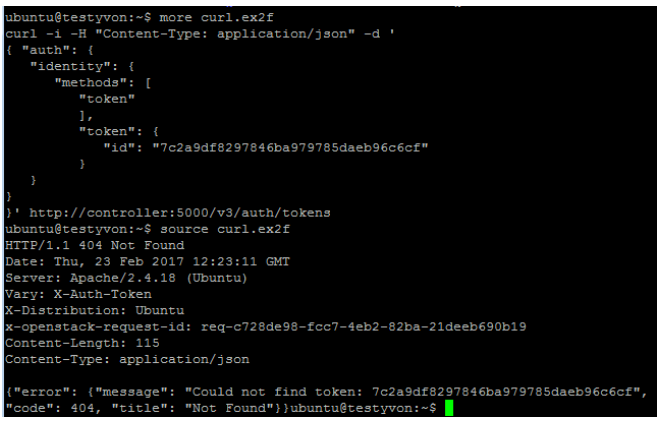
Ceci est donc une récapitulation des principaux services OpenStack que nous pouvons appeler - et comment nous pouvons aussi les appeler. Cette information s'appelle le 'catalog' selon la terminologie d'OpenStack.



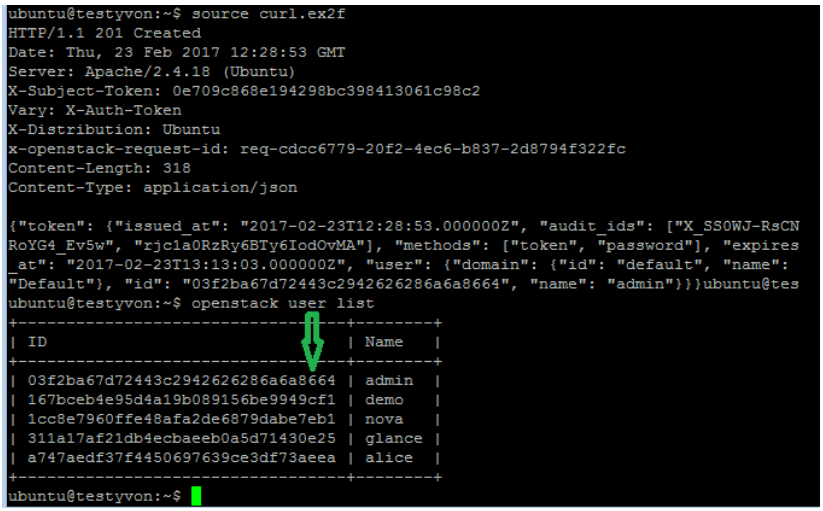
1.5 - Obtenir un jeton à partir d'un jeton et de curl

nous pouvons donc obtenir un jeton à partir d'un jeton (afin par exemple de le renouveler et de prolonger ainsi sa durée de vie).

Pour cela, nous avons stocké le programme `curl` dans un fichier (désigné par curl.ex2f). L'exécution du programme provoque une erreur puisque le jeton mentionné dans la commande curl n'est plus valide.



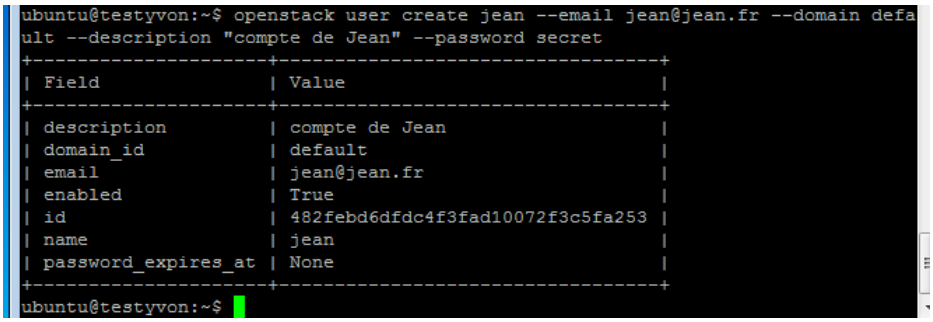
Pour éviter cette erreur, nous reprenons l'identité du jeton obtenu dans la commande précédente et la plaçons dans la commande `curl`.



On voit bien que dans les données du jeton, nous retrouvons l'identité de admin pour qui le jeton est émis. Ce jeton n'a pas de scope et il ne fournit donc pas les extraits du catalogue comme une des requêtes précédentes.

1.6 quelques commandes supplémentaires

- voir les différents utilisateurs: `$ openstack user list`
- voir les différents projets : `$ openstack project list`
- voir les différents groupes: `$ openstack group list`
- créer le domaine acme: `$ openstack domain create acme`
- créer un nouvel utilisateur dans un domaine:
- `$ openstack user create jean --email jean@jean.fr \`
- `--domain default --description "compte de Jean" \`
- `--password secret`

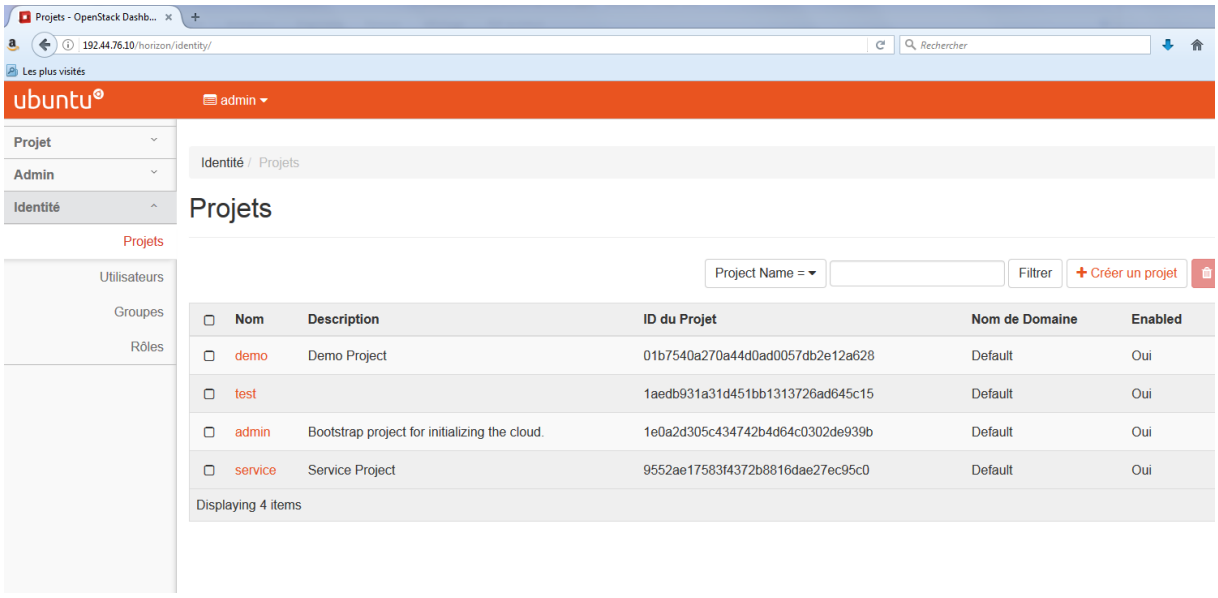


Dans la séquence suivante de commandes OpenStack, on liste les domaines, les rôles puis les associations entre les utilisateurs et les rôles (sera fait dans le TP ci après). A noter également que ce songt les UUID qui apparaissent mais qui ne sont donc pas très lisibles (ou exploitables!) et de ce fait avec l'argument '`names`', il y a possibilité de voir les noms en clair.

```
ubuntu@testyvon:~$ openstack domain list
+---------+---------+---------+--------------------+
| ID      | Name    | Enabled | Description        |
+---------+---------+---------+--------------------+
| default | Default | True    | The default domain |
+---------+---------+---------+--------------------+
ubuntu@testyvon:~$ openstack role assignment list --role-domain default --names
+--------+----------------+-------+-----------------+--------+-----------+
| Role   | User           | Group | Project         | Domain | Inherited |
+--------+----------------+-------+-----------------+--------+-----------+
| admin  | admin@Default  |       | admin@Default   |        | False     |
| member | demo@Default   |       | demo@Default    |        | False     |
| admin  | nova@Default   |       | service@Default |        | False     |
| admin  | glance@Default |       | service@Default |        | False     |
+--------+----------------+-------+-----------------+--------+-----------+
ubuntu@testyvon:~$ openstack role assignment list
+---------------+---------------+-------+-----------------+--------+-----------+
| Role          | User          | Group | Project         | Domain | Inherited |
+---------------+---------------+-------+-----------------+--------+-----------+
| a336439623314 | 03f2ba67d7244 |       | 1e0a2d305c43474 |        | False     |
| a809c0c856c61 | 3c2942626286a |       | 2b4d64c0302de93 |        |           |
| 278209        | 6a8664        |       | 9b              |        |           |
| 1e39742853374 | 167bceb4e95d4 |       | 01b7540a270a44d |        | False     |
| 9c3a93dd6c354 | a19b089156be9 |       | 0ad0057db2e12a6 |        |           |
| dd2990        | 949cf1        |       | 28              |        |           |
| a336439623314 | 1cc8e7960ffe4 |       | 9552ae17583f437 |        | False     |
| a809c0c856c61 | 8afa2de6879da |       | 2b8816dae27ec95 |        |           |
| 278209        | be7eb1        |       | c0              |        |           |
| a336439623314 | 311a17af21db4 |       | 9552ae17583f437 |        | False     |
| a809c0c856c61 | ecbaeeb0a5d71 |       | 2b8816dae27ec95 |        |           |
| 278209        | 430e25        |       | c0              |        |           |
+---------------+---------------+-------+-----------------+--------+-----------+
ubuntu@testyvon:~$
```

2. - Interface graphique d'administration via Horizon

connectez vous sur l'interface de gestion de OpenStack avec Horizon. Vous pouvez via l'interface graphique créer des utilisateurs, des projets, des rôles etc. Explorez les différentes fonctionnalités proposées.....

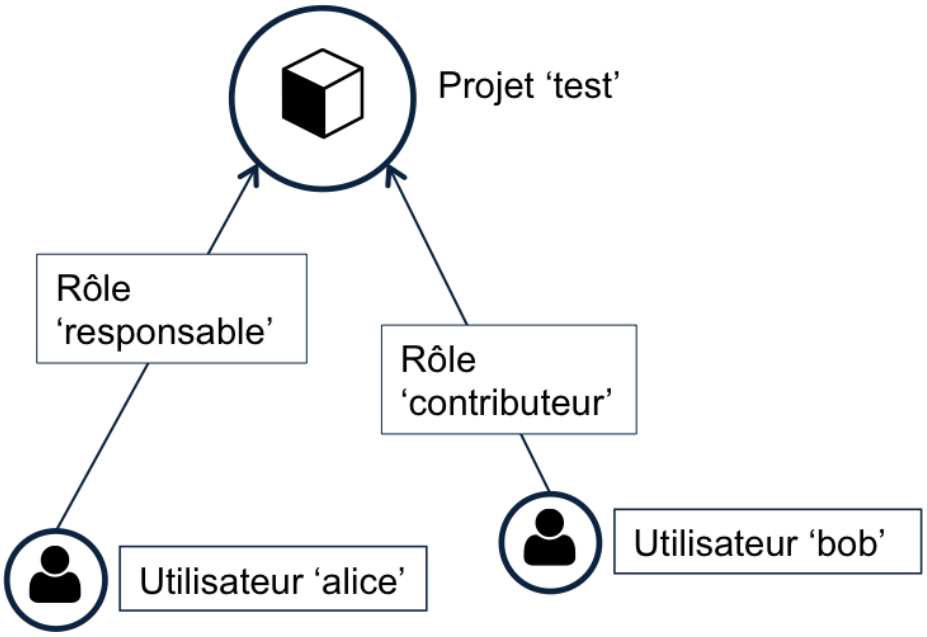


3. - Exemple complet avec expression de politique de sécurité

Ishakh FALL, Ophélie HAMON et Lucas STADELMANN (élèves ingénieurs en formation FIP à Télécom Bretagne) ont proposé cet exemple dans le cadre d'un projet de 3ème année.

3.1. Description du scénario

Dans ce scénario, nous allons mettre en place deux utilisateurs d'un projet (dénomés Bob et Alice). Chacun d'eux aura un rôle spécifique vis-à-vis de ce projet. Nous pouvons représenter ce cas d'usage de la manière suivante :



Chacun des rôles dispose de droits spécifiques selon sa nature:

- Rôle de responsable : peut créer, lire, modifier et supprimer les instances de machines virtuelles et les images du projet, y compris celles des contributeurs
- Rôle de contributeur : peut créer, lire, modifier et supprimer uniquement ses propres instances de machines virtuelles et ses images du projet

3.2. Création des différents utilisateurs, rôles et affectations

Pour réaliser le scénario décrit précédemment, il est nécessaire de créer :

- un projet : "test"
- deux utilisateurs : "alice" et "bob"
- deux rôles : "responsable" et contributeur

Les commandes ci-dessous permettent d'exécuter ces actions.

Création du projet "test"

```
openstack project create test
```

Lister les projets créés (et vérification que le projet est bien créé)

```
openstack project list
```

Création de 2 utilisateurs, "Alice" et "Bob", sur le projet "test" avec leur mot de passe

```
openstack user create --project test --password pass4alice alice
openstack user create --project test --password pass4bob bob
```

Lister les utilisateurs créés

```
openstack user list
```

Création des rôles "responsable" et "contributeur"

```
openstack role create responsable
openstack role create contributeur
```

Lister les rôles créés

```
openstack role list
```

Donner le rôle "responsable" à "alice" dans le cadre du projet 'test'

```
openstack role add --user alice --project test responsable
```

Donner le rôle "contributeur" à "bob" dans le cadre du projet 'test'

```
openstack role add--user bob --project test contributeur
```

Afficher la liste des rôles et avec les utilisateurs correspondants

```
openstack role assigment list --role-domain default --names
```

3.3. Politiques de sécurité pour 2 services d'OpenStack: Nove et Glance

Nous allons ensuite modifier les fichiers *policy.json* des services Nova et Glance. De cette manière, nous prendrons en compte les nouveaux rôles précédemment créés et vous pourrez expériementer comment une politique de sécurité est exprimée et comment elle est appliquée.

Attention: pensez à sauvegarder les fichiers que vous allez modifier ! Les erreurs de syntaxe sont fatales et avec les fichiers de sauvegarde, vous pourrez reprendre sans problème la suite du TP.

Le *policy.json* de Keystone ne sera pas modifié. En effet, ce dernier définit des actions de plus haut niveau, propres à la manipulation des objets métiers. Il est donc préférable que la gestion du projet, ainsi que des utilisateurs et des rôles associés à ce projet, soit réservée à un administrateur plutôt qu'au responsable du projet. D'une manière générale, une erreur ou une mauvaise manipulation sur une règle de ce fichier peut avoir des résultats catastrophiques sur l'ensemble de votre cluster OpenStack : il vaut donc mieux maitriser complètement l'impact des modifications avant d'y procéder.

3.3.1. Politique de sécurité pour Nova

Le fichier qui liste les politiques de sécurité associées au service Glance se trouve dans le répertoire suivant : `/etc/glance/policy.json`.

Dans un premier temps, nous allons exprimer les règles de sécurité qui s'appliqueront aux rôles de "responsable" et de "contributeur. Puis, nous nous allons les traduire sous la forme de règles qui seront insérées dans le fichier `policy.json`.

Avec le rôle de contributeur, un utilisateur aura le droit de :

- download/upload une image
- créer/lire une image
- modifier/supprimer une image qu'il a lui-même créée

Avec le rôle de responsable, un utilisateur aura le droit de :

- download/upload une image
- créer/lire/modifier/supprimer/publier une image
- lire/ajouter/modifier des tâches
- modifier le cache des images

Nous avons donc intérêt à rajouter des prédicats qui vérifieront des conditions. Ces prédicats sont déclarés en partie gauche d'une règle (c'est-à-dire d'une ligne dans le fichier de politique) et seront utilisés en partie droite en étant préfixé par 'rule:'

Voici le fichier de configuration correspondant qui complète la version initiale du fichier `policy.json`:

```
{

-- on définit ci après l'ensemble des prédicats qui seront utilisés ultérieurement
   "is_admin":  "role:admin",
   "is_responsable":  "role:responsable",
   "is_contributeur":  "role:contributeur",
   "is_owner": "user_id:%(user_id)s",
   "responsable_or_owner": "rule:is_responsable or rule:is_owner",
   "responsable_or_contributeur": "rule:is_responsable or rule:is_contributeur",
   "default": "",

on définit ci-après l'ensemble des conditions pour qu'une fonction d'un service (via une API) soit appelable par
un utilisateur. Remarquez en particulier l'utilisation des opérateurs booléens pour constituer des conditions
plus élaborées.

   "add_image": "rule:is_admin or rule:responsable_or_contributeur",
   "delete_image": "rule:is_admin or rule:responsable_or_owner",
   "get_image": "rule:is_admin or rule:responsable_or_contributeur",
   "get_images": "",
   "modify_image": "rule:is_admin or rule:responsable_or_owner",
   "publicize_image": "rule:is_admin or rule:is_responsable",
   "copy_from": "",

   "download_image": "rule:is_admin or rule:responsable_or_contributeur",
   "upload_image": "rule:is_admin or rule:responsable_or_contributeur",

   "delete_image_location": "",
   "get_image_location": "",
   "set_image_location": "",

   "add_member": "",
   "delete_member": "",
   "get_member": "",
   "get_members": "",
   "modify_member": "",

   "manage_image_cache": "rule:is_admin or rule:is_responsable",

   "get_task": "rule:is_admin or rule:is_responsable",
   "get_tasks": "rule:is_admin or rule:is_responsable",
   "add_task": "rule:is_admin or rule:is_responsable",
   "modify_task": "rule:is_admin or rule:is_responsable",

on définit ci après l'ensemble des fonctions et services qui sont appelables sans condition : tous les utilisateurs
peuvent les appeler - la condition toujours vraie est dénotée "".

   "deactivate": "",
   "reactivate": "",
```

```
  "get_metadef_namespace": "",
  "get_metadef_namespaces": "",
  "modify_metadef_namespace": "",
  "add_metadef_namespace": "",

  "get_metadef_object": "",
  "get_metadef_objects": "",
  "modify_metadef_object": "",
  "add_metadef_object": "",

  "list_metadef_resource_types": "",
  "get_metadef_resource_type": "",
  "add_metadef_resource_type_association": "",

  "get_metadef_property": "",
  "get_metadef_properties": "",
  "modify_metadef_property": "",
  "add_metadef_property": "",

  "get_metadef_tag": "",
  "get_metadef_tags": "",
  "modify_metadef_tag": "",
  "add_metadef_tag": "",
  "add_metadef_tags": ""
}
```

A partir de votre fichier, faite un appel pour une API avec un rôle spécifique. Que se passe t il si l'appel est autorisé ? s il est interdit ? modifiez maintenant une des conditions toujours vraies (c'est-à-dire "") par la condition jamais (c'est-à-dire toujours fausse).

3.3.2. Politique de sécurité pour Nova

Le fichier qui liste les politiques de sécurité associées au service Nova se trouve dans le répertoire suivant : `/etc/nova/policy.json`.

Dans un premier temps, vous allez consulter le fichier des politiques de sécurité et déterminer quels sont les appels d'API qu'un utilisateur peut réaliser ou non.

Puis dans un deuxième temps, exprimez les règles de sécurité qui s'appliqueront aux rôles de “responsable” et de “contributeur. Puis, traduisez-les sous la forme de règles qui seront insérées dans le fichier `policy.json`.

Avec le rôle de contributeur, un utilisateur aura le droit de :

- créer une instance
- gérer le réseau et les volumes d'une instance qu'il a lui-même créé
- modifier une instance qu'il a lui-même créé
- lire/modifier/supprimer les métadonnées d'une instance qu'il a lui-même créé
- démarrer/stopper une instance qu'il a lui-même créé
- verrouiller/déverrouiller une instance qu'il a lui-même créé
- sauver/dé-sauver une instance qu'il a lui-même créé
- suspendre/reprendre une instance qu'il a lui-même créé
- mettre en pause/reprendre une instance qu'il a lui-même créé
- ajourner/dé-ajourner une instance qu'il a lui-même créé
- prendre un snapshot/un backup d'une instance qu'il a lui-même créé
- modifier la taille d'une instance qu'il a lui-même créé
- reconstruire/redémarrer/supprimer une instance qu'il a lui-même créé

Avec le rôle de responsable, l'utilisateur aura le droit de :

- créer une instance
- gérer le réseau et les volumes d'une instance
- modifier une instance
- lire/modifier/supprimer les métadonnées d'une instance
- démarrer/stopper une instance
- verrouiller/déverrouiller une instance
- sauver/dé-sauver une instance
- suspendre/reprendre une instance
- mettre en pause/reprendre une instance
- ajourner/dé-ajourner une instance
- prendre un snapshot/un backup d'une instance
- modifier la taille d'une instance
- reconstruire/redémarrer/supprimer une instance

exprimez les différentes règles en lien avec ces autorisations que l'on souhaite accorder en fonction du rôle de l'appelant de l'API. Exprimez ensuite sous la forme de règles qui seront insérées dans le fichier `policy.json`.