Institut Universitaire de Technologie,
Aix-Marseille Université

RAPPORT DE STAGE
Diplôme Universitaire de Technologie
Spécialité Réseaux et Télécommunications

Analyste de production

Grégoire SABAUD

Fundvisory

Responsable entreprise : Mickael HOSDEZ

Responsable académique : Delphine ROUSSEAU

2021

Table des matières

1 Introduction

Dans le cadre de mon cursus en IUT Réseaux et Télécommunications à Marseille, site de Luminy, la réalisation d'un stage de dix semaines en fin d'études était nécessaire à la validation de mon diplôme. Ce stage fut une expérience enrichissante, dans le domaine de l'administration système et de l'automatisation des tâches, mais aussi d'un point de vue personnel sur le monde du travail en entreprise. Eu égard au contexte de la crise sanitaire, ce stage a donc été effectué de manière distancielle, au sein de l'entreprise Fundvisory, située à Buchelay (78).
Il a été réalisé du 19 avril au 26 juin 2021.

J'ai donc intégré l'équipe de développement et d'administration de Fundvisory pour travailler sur Ansible et Rundeck, deux plateformes d'automatisation des tâches.

2 Présentation de l'entreprise

Fundvisory est une entreprise proposant une solution informatique de conseils d'investissements de manière complète, experte, intelligente, personnalisée et règlementaire. Elle est localisée sur deux sites :

- Buchelay, pour son centre de développement ;
- Paris, pour son centre de recherche.

Elle a été créée en 2015 par Romain Deguest, Nicolas Gonzalez et Laurent Jaumotte.
L'entreprise se fait rapidement remarquer, si bien qu'en février 2016, Generali France fait confiance à Fundvisory pour lancer une plateforme d'assurance-vie en ligne.
En Septembre 2016, Aviva France choisi Fundvisory pour « Yousee », un robo-advisor développé par l'entreprise.
En mars 2017, l'entreprise est lauréate du concours de l'innovation numérique de la Banque d'Investissement Publique. Puis, au mois de juin de la même année, Fundvisory remporte une série d'appels d'offres d'ampleur, notamment de Bourse Direct, Nataxis ou d'OFI Asset Management.
En janvier 2018, Fundvisory lance Prospeer, un robo-advisor d'épargne salariale.
En juillet 2018, l'entreprise fait une levée de fonds de près de 1.8 millions d'euros auprès de la Macif et d'Aviva France.

L'entreprise est divisée en deux équipes, une équipe technique et une équipe de recherche. J'ai pu intégrer l'équipe technique, au sein d'une sous division spécialisée dans la production, dirigée par Mickael Hosdez (cf. Figure 1).

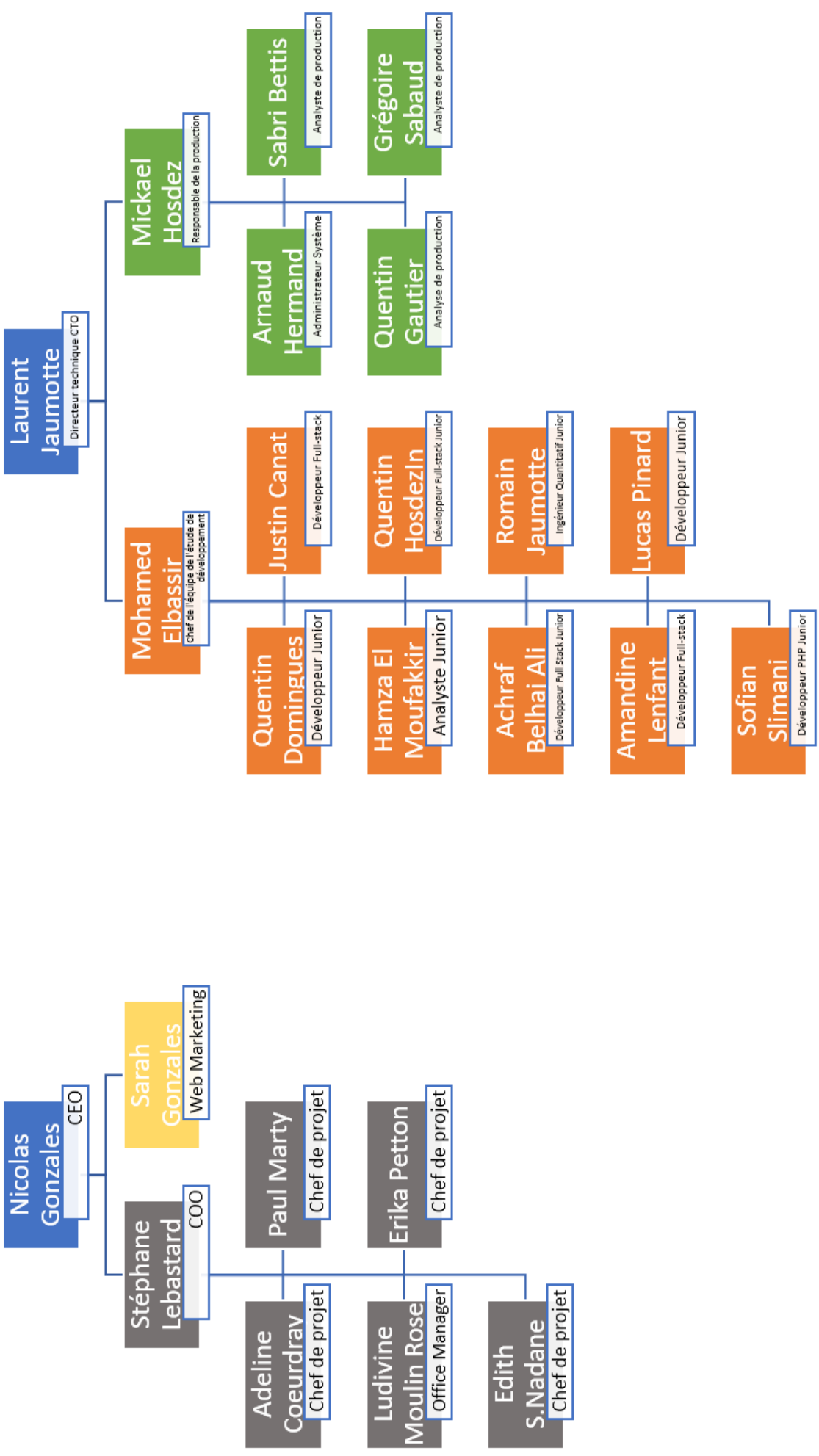

Figure 1 : Organigramme de l'entreprise

3 Ansible

3.1 Description d'Ansible

3.1.1 Inventaires et connexions

Ansible est un logiciel d'automatisation des tâches développé en python par la société Red Hat. Il est Open Source, son code est donc disponible gratuitement sur internet. Ansible permet d'automatiser toute tâche d'un système sous Linux ou Windows, que ce soit sur machine locale ou distante. En effet, Ansible permet, via des connexions **SSH***, d'exécuter les tâches demandées sur des machines distantes, quel que soit leur nombre (d'une seule jusqu'à plusieurs milliers).

Les machines sur lesquelles Ansible va se connecter sont précisées dans un fichier dit « Inventaire », sous format **INI*** ou sous format **YAML***. Un fichier inventaire exemple peut être trouvé à l'emplacement de configuration de base d'Ansible.

Le fichier inventaire lu par Ansible lorsqu'une liste de tâche est exécutée, va dépendre d'une certaine liste de priorités. La priorité la plus haute reviendra au fichier indiqué avec l'argument -i lors du lancement de l'exécution d'une liste de tâches. Si cet argument n'est pas précisé, le fichier choisi sera l'inventaire de base d'Ansible, présent dans le répertoire de configuration (sous Linux, par défaut /etc/ansible/hosts).

Les machines, appelées hôtes, pourront être rassemblés dans des groupes d'hôtes, qui peuvent eux-mêmes être regroupés dans des groupes plus importants. Un seul hôte peut appartenir à plusieurs groupes. Par exemple, l'hôte au nom de "R&T" pourra faire partie d'un groupe "Luminy", ainsi que d'un groupe "IUT", lui-même faisant partie d'un groupe "Aix-Marseille Université" (cf. figure 2).

```
[MachinesVirtuelles]
VM2 ansible_host=10.10.1.71
VM3 ansible_host=10.10.1.72
```

Figure 2 : Deux hôtes dans un groupe

Pour fonctionner via des connexions SSH, l'utilisation de **clés SSH*** est nécessaire. Si, pour des raisons de sécurité, les clés SSH demandent une passphrase (une phrase de passe), on pourra la fournir lorsqu'on lancera notre programme Ansible avec l'argument "--ask-pass". Si les machines ont toutes des passphrases différentes, on peut utiliser un agent SSH. Si l'utilisateur veut éviter l'utilisation d'un agent SSH, alors on peut indiquer un fichier contenant une clé privée pour chaque machine hôte dans le fichier inventaire.

3.1.2 Playbooks et rôles

Les tâches qui s'exécuteront sur les machines hôtes sont décrites dans une liste de tâches appelée playbook. Ces fichiers sont écrits en format YAML. Chaque tâche inscrite dans le playbook représente l'exécution d'un module d'Ansible. Par défaut, les tâches s'exécuteront une fois par machine hôte. Si une tâche doit être exécutée une seule fois, on peut lui donner l'attribut « run_once : True ». Elle s'exécutera alors sur la première machine hôte inscrite dans l'inventaire.
Si une tâche doit s'exécuter sur une machine précise, on peut lui donner l'attribut « delegate_to : » suivi de l'adresse IP de la machine. Dans la même logique, si, pour des raisons de sécurité, une tâche doit s'exécuter sur la machine exécutant Ansible, on peut donner l'attribut « delegate_to : localhost ».
Enfin, si une tâche nécessite les droits du **super-utilisateur***, on peut donner l'attribut « become : yes » à la tâche exécutée.

Le format YAML qu'utilisent les playbooks permet la déclaration et l'utilisation de variables. Dans Ansible, nous pouvons déclarer des variables de différentes façons qui ont chacune une priorité bien précise. Ainsi, si une variable est déclarée plusieurs fois, la déclaration possédant la priorité la plus élevée donnera sa valeur à la variable.

Les tâches qui peuvent être exécutées par Ansible existent sous forme de modules. Il existe des modules officiels, intégrés à la version de base d'Ansible et des modules créés par la communauté, directement téléchargeables via la plateforme Ansible-Galaxy. Lors du stage, je n'ai utilisé que des modules officiels.

La création de playbooks facilitant la gestion de comptes utilisateurs sur des serveurs distants a été ma principale tâche au cours de ce stage. On m'a donc fourni trois **machines virtuelles***, ayant Ubuntu comme système d'exploitation, afin de travailler dans un environnement sans danger pour l'infrastructure déjà existante de l'entreprise. J'ai donc mis en place un environnement Maître/Esclave où Ansible serait installé sur la machine maître (cf. Figure 3).

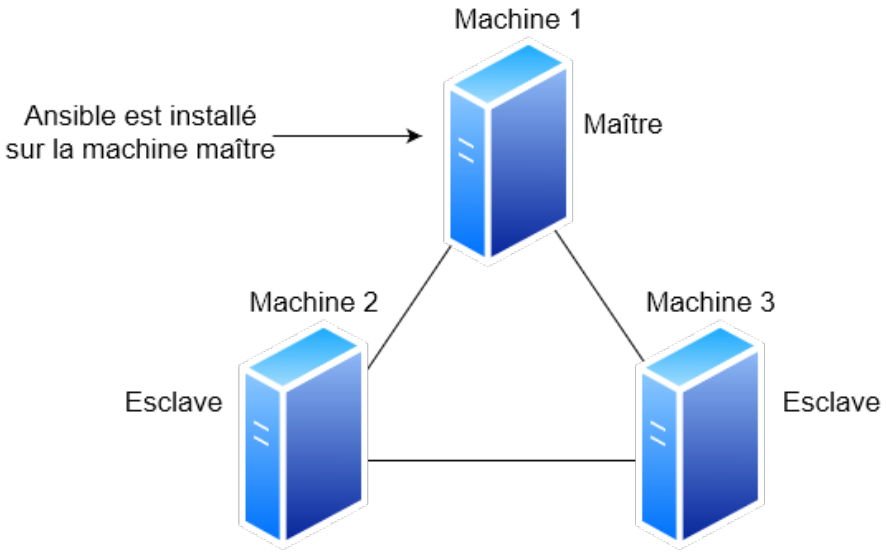

Figure 3 : Infrastructure de travail

Après la mise en place des clés SSH, j'ai commencé par créer un playbook pour ajouter un compte utilisateur sur les machines distantes. Ce compte doit comporter un identifiant renseigné au préalable par l'utilisateur, un mot de passe généré aléatoirement, une adresse e-mail associée au compte et un **shell***.

La gestion d'un plus grand nombre de tâches devient plus compliquée sur un seul fichier YAML. C'est pourquoi Ansible propose de créer des « rôles », une arborescence de répertoires comprenant :

- Un répertoire « defaults » contenant les fichiers définissant les variables ayant la plus faible priorité du rôle, celles qui peuvent être écrasées par tout autre type de déclaration.
- Un répertoire « files » contenant les fichiers nécessaires à l'exécution du playbook.
- Un répertoire « handlers » contenant toutes les tâches qui s'exécuteront sous des conditions précises, par exemple lorsqu'un changement est produit lors de l'exécution d'une tâche.
- Un répertoire « meta » contenant les données de description du rôle (appelées métadonnées).
- Un répertoire « tasks » contenant tous les fichiers de tâches créés pour le rôle.
- Un répertoire « templates » contenant des fichiers modèles qui pourraient être copier sur les machines hôtes.
- Un répertoire « tests » contenant un inventaire ayant par défaut le seul hôte « localhost ». Il sert à appeler le rôle et le tester.
- Un répertoire « vars » contenant des fichiers de déclarations de variables ayant une priorité supérieure à ceux du répertoire « defaults »

Chacun de ces répertoires contient un fichier « main.yml ». Si un répertoire n'est pas utilisé pour le rôle, alors il peut être supprimé.

Le fichier « main.yml » du répertoire « tasks » sera le fichier exécuté lorsqu'on appellera le rôle via un playbook. Il contient la liste des tâches du rôle qui vont s'exécuter.
L'appel d'un rôle se fera dans un playbook via le module « roles » d'Ansible. Un seul playbook peut appeler plusieurs rôles.

3.2 Projets réalisés sur Ansible

3.2.1 Rôle de création d'utilisateur

La création d'utilisateur via Ansible regroupe un grand nombre de tâches. L'élaboration d'un rôle « CreateUser » a été nécessaire pour exécuter toutes les tâches liées à la création de comptes sur les machines hôtes (cf. Figure 5). Ce rôle est appelé dans un playbook nommé « createuser.yaml » (cf. Figure 4).

```
---

- name: Create new users
  hosts: all
  become: yes
  roles: [ createuser ]
```

Figure 4 : playbook d'appel de rôle

```
# tasks file for createuser

- include: generatepassword.yaml

- include: hashpassword.yaml

- include: passwordforrundeck.yaml

- include: create_new_user.yaml

- include: sendpasswordbymail.yaml
```

Figure 5 : main.yaml du répertoire « tasks » du rôle Createuser

La première tâche du rôle est de générer un mot de passe automatiquement. Ce mot de passe doit être constitué de lettres minuscules, majuscules, de chiffres et de caractères spéciaux. Par défaut, Ubuntu ne possède pas de paquet pouvant générer automatiquement des mots de passe. Après quelques recherches, j'ai retenu le programme APG (Automated Password Generator), qui correspond aux critères de génération automatique. Le playbook exécute donc la commande « apg -a 1 -n 1 -M SNCL -E \"\'\$\;\\\`` » via le module « Shell ». On génère un mot de passe non prononçable (-a 1) une seule fois (-n 1) dans lequel il doit y avoir des lettres minuscules, majuscules, des chiffres, et des caractères spéciaux (-M SNCL). L'argument « -E » permet d'exclure la génération de certains caractères. Les caractères pouvant être reconnus par le Shell ont donc été exclus de la génération pour éviter tout problème dans les fichiers de configuration système ou dans les commandes qui vont traiter le mot de passe venant d'être généré.

Une fois le mot de passe généré, il est nécessaire de le hasher pour qu'il n'apparaisse pas de manière lisible dans les fichiers de configuration. Comme pour la génération de mot de passe, il n'y a pas de paquet pré-installé pour le hashage. Après des recherches, je me suis tourné vers le programme « mkpasswd ». Les mots de passe générés sont donc hashés via ce paquet au format SHA-512.

Par la suite, le compte utilisateur sera créé sur les machines hôtes grâce au module « User » d'Ansible. On doit donc renseigner :

- Le nom du compte utilisateur
- Son mot de passe hashé
- Le shell qu'il utilisera
- Le type de compte créé (ici, utilisateur)
- Le chemin de son répertoire de travail
- L'adresse e-mail de l'utilisateur, en commentaire du compte

Enfin, on envoie un mail via le module « Mail » d'Ansible à l'adresse de l'utilisateur pour indiquer que son compte à été créé. Ce mail comporte également le mot de passe non hashé de l'utilisateur.

3.2.2 Rôle de suppression d'utilisateur

Il s'agit ici de créer un rôle visant à supprimer une liste d'utilisateur des machines distantes. L'utilisateur exécutant le playbook peut donc rentrer une liste de comptes à supprimer par Ansible. Une variable booléenne nommée « remove_home » est demandée. Si elle est définie comme « True », les répertoires de travail des utilisateurs listés seront supprimés, ainsi que tout leur contenu. Si elle est définie comme « False », les répertoires de travail seront maintenus.
Pour supprimer les comptes un par un, une boucle est nécessaire. On peut demander à Ansible de faire une boucle pour tous les éléments d'une liste. Le module « User » est donc utilisé pour supprimer les comptes les uns après les autres, en déclarant leur état comme « Absent » sur la machine (cf. Figure 6).

```
---
  - block:

      - name: Remove Users
        user:
          name: '{{ item }}'
          state: absent
          remove: '{{ remove_home }}'
        loop: "{{ _raw_params }}"

      - name: Remove groups
        group:
          name: '{{ item }}'
          state: absent
        loop: "{{ _raw_params }}"

```

Figure 6 : Playbook de suppression des utilisateurs contenus dans la liste entrée par l'utilisateur au moment de l'exécution son exécution

Si un compte de la liste n'est pas présent sur une machine, Ansible passe la tâche et supprime le compte correspondant à l'élément suivant de la boucle.

3.2.3 Rôle de modification d'informations utilisateur

Il s'agit ici de créer un rôle visant à modifier les informations d'un utilisateur, c'est-à-dire permettre :

- Un changement d'adresse e-mail associée au compte utilisateur
- Une modification des groupes dont fait partie l'utilisateur
- Une modification du Shell utilisé (notamment si le compte doit être désactivé sans être supprimé)
- Un changement du mot de passe du compte, toujours de manière aléatoire, pour des raisons de sécurité.

Chaque tâche exécutée ne doit pas être dépendante d'une autre. Donc, si l'administrateur fournit une variable correspondant à l'utilisation d'un Shell précis (comme le /bin/nologin pour désactiver le compte) mais ne précise rien d'autre dans la commande, le playbook n'exécutera que la tâche correspondant à la modification du Shell.
Dans cet optique, Ansible permet de donner aux tâches des conditions d'exécution, via l'attribut « when : ». Ainsi, lorsque la condition dictée après « when : » est remplie, la tâche s'exécute. Par exemple, pour modifier un mot de passe, le groupe de tâches s'exécutera si l'utilisateur spécifié existe bien sur les machines hôtes et si la variable « password_modification » est égale à « True » (cf. Figure 7 et 8).

```
- name: verifier si utilisateur existe
  shell: id -u "{{ username }}"
  register: user_exist
  ignore_errors: True
```

Figure 7 : Vérification de l'existence de l'utilisateur sur les machines hôtes

```
  when:
    - user_exist.stdout != ""
    - password_modification == "True"
```

Figure 8 : Conditions d'exécution du groupe de tâches

On a donc :

- Une variable « shellused » qui, lorsqu'elle est définie, lance la tâche modifiant le shell du compte par son contenu.
- Une variable « usergroups » qui, lorsqu'elle est définie, lance la tâche modifiant les groupes auxquels l'utilisateur appartient.
- Une variable « mailaddress » qui, lorsqu'elle est définie, lance le groupe de tâches modifiant les informations contenues dans le champ « commentaire » du fichier « /etc/passwd ». La nouvelle adresse électronique, contenue dans la variable « mailaddress », remplace donc l'adresse obsolète. Suite à ce changement, un mail est envoyé à la nouvelle adresse pour avertir l'utilisateur.
- Une variable booléenne « password_modification » qui, si elle est égale à True, lance le groupe de tâches modifiant le mot de passe de l'utilisateur. Ce mot de passe est généré et hashé selon les mêmes critères que lors de la création de l'utilisateur. Le mot de passe est ensuite envoyé à l'adresse mail de l'utilisateur.

Dans le cas d'un changement d'adresse mail et de mot de passe, le groupe de changement d'adresse mail sera toujours antérieur à celui qui concerne le mot de passe.

3.2.4 Rôle de listage d'utilisateurs présents sur machine hôtes

Parallèlement aux objectifs qui m'ont été donnés, j'ai créé un rôle listant tous les comptes utilisateurs sur les machines distantes. Cela m'évite de me connecter en SSH et de vérifier le contenu du fichier « /etc/passwd » constamment sur les deux hôtes. Pour cela, le playbook exécute un script bash sur chacun des hôtes. Ce script est contenu dans un fichier présent dans le répertoire « file » du rôle et permet, lors de son exécution, de lister tous les comptes utilisateurs de la machine. Les comptes systèmes ne sont pas inclus dans la liste (cf. Figure 9).

```
gsabaud@buc-gsabaud1:~$ ansible-playbook /etc/ansible/playbooks/listusers.yaml -K
BECOME password:

PLAY [Lists all users from remote machines] ************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************
ok: [Machine2]
ok: [Machine3]

TASK [listusers : List existing users] *****************************************************************************************************************************************
changed: [Machine2]
changed: [Machine3]

TASK [listusers : set_fact] ****************************************************************************************************************************************************
ok: [Machine2]
ok: [Machine3]

TASK [listusers : Print user account list] *************************************************************************************************************************************
ok: [Machine2] => {
    "changed": false,
    "msg": [
        "The user accounts on Machine2 are :",
        "Jerem Jostophe Julo Lutykse gsabaud rundeck"
    ]
}
ok: [Machine3] => {
    "changed": false,
    "msg": [
        "The user accounts on Machine3 are :",
        "Jerem Jostophe Julo Lutykse gsabaud rundeck"
    ]
}

PLAY RECAP *********************************************************************************************************************************************************************
Machine2                   : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
Machine3                   : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

gsabaud@buc-gsabaud1:~$
```

Figure 9 : Liste des comptes présents sur les deux hôtes

Grâce à cette procédure, j'ai appris à utiliser le module « Debug » d'Ansible, qui permet notamment d'afficher des messages pendant l'exécution d'un playbook (cf. Figure 9) ainsi que le module « Script » qui permet d'exécuter un script présent localement sur des machines hôtes.
Le chemin du programme exécutant le script (par exemple, pour un script écrit en bash, le chemin sera /bin/bash) peut être précisé, si nécessaire. Cependant, il doit être présent sur la machine hôte.

4 Rundeck

4.1 Description et configuration de Rundeck

Rundeck est un logiciel également développé par l'entreprise Red Hat. Il existe en deux versions : une version « Community » gratuite en open-source, et une version « Enterprise » payante, avec quelques fonctionnalités supplémentaires.
Rundeck permet de fournir une interface web pour aider à l'automatisation des tâches. Nous l'avons utilisé car il intègre un module spécialement fait pour l'exécution de playbooks Ansible.

Rundeck reprend les mêmes principes qu'Ansible sans les importer pour autant. Chaque liste d'actions à exécuter sur les machines hôtes ne sera pas appelé playbook, mais « job ». Tous les jobs sont rangés dans des « projets ». Ces projets, lors de leur exécution, s'appliqueront à des machines hôtes, que Rundeck appelle « nodes ».

Toutes ces informations sont stockées dans une base de données relationnelles. Par défaut, celle de Rundeck est basée sur H2, un logiciel de gestion de base de données. Cependant, pour imiter l'architecture existante de l'entreprise, je devais faire fonctionner Rundeck sur une base de données écrite en PostgreSQL et installée sur une machine distante.

L'architecture sur laquelle je travaillais était constituée de la machine maître exécutant Ansible et Rundeck, une machine esclave comportant la base de données en PostgreSQL et une autre machine esclave servant juste de node (cf Figure 10).

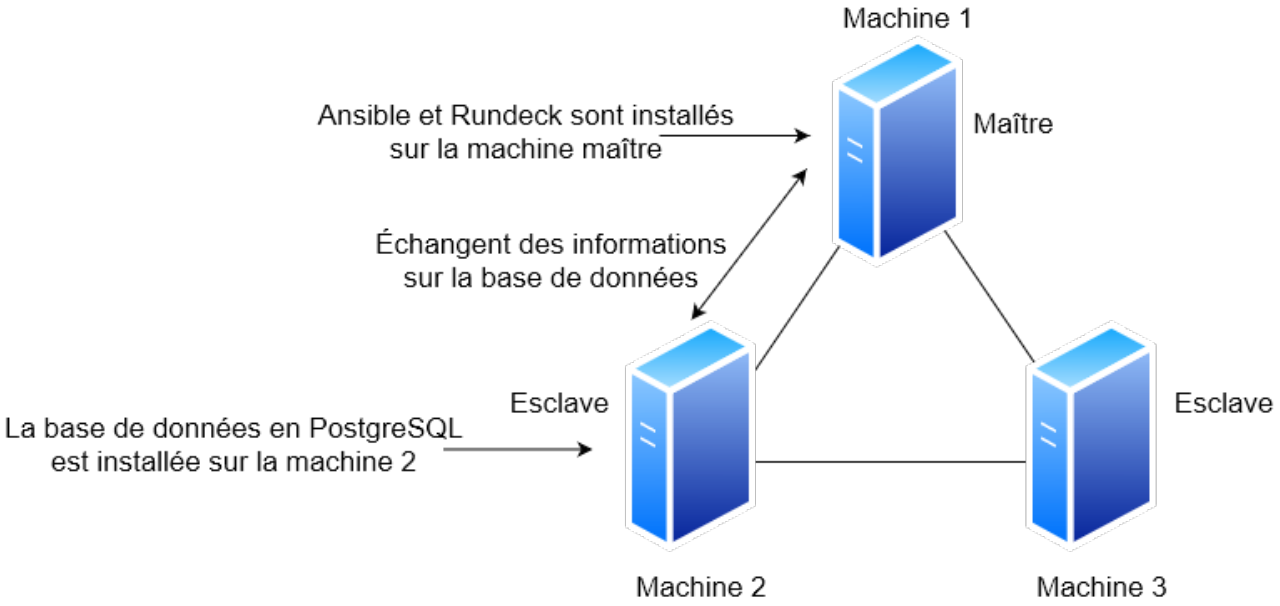

Figure 10 : Architecture de travail avec Rundeck

La configuration de la base s'opère via l'invite de commande PostgreSQL. Une base de données et un utilisateur sont créés. L'utilisateur peut se connecter à cette base de données via un login et un mot de passe. On précise ensuite, dans les fichiers de configuration de Rundeck, le type de base de données (ici PostgreSQL), son emplacement (l'adresse IP de la machine distante contenant la base) et son nom. Il est également nécessaire de préciser le login et le mot de passe utilisés pour se connecter à la base de données.
Par défaut, PostgreSQL n'accepte que des connexions venant de la machine sur laquelle il est installé. Il faut donc modifier sa configuration pour qu'il accepte une connexion venant de l'adresse IP de la machine faisant tourner Rundeck.

Mon objectif était d'importer tous les projets Ansible sur lesquels j'avais travaillé (à part pour le projet personnel de listing de comptes) et les faire fonctionner avec l'interface web de Rundeck. La première étape était d'importer les machines hôtes d'Ansible et les convertir en nodes. Cependant, lorsque Rundeck envoyait un **ping*** vers les autres machines virtuelles, il n'obtenait pas de réponse. Cela est dû au fait que lors de son installation, Rundeck créé un compte système à son nom sur la machine qui l'héberge. Il utilise ensuite ce compte pour exécuter ses commandes et ses jobs. Les clés SSH que j'avais créé en début de stage étaient donc obsolètes. Il fallait donc en recréer pour laisser le compte système Rundeck accéder facilement aux différentes nodes.

Rundeck avait également un problème concernant toutes les tâches assignées en exécution locale (via l'attribut « delegate_to : localhost »). Une clé SSH spéciale devait donc être créée pour qu'il puisse se connecter à la machine Maître lors de l'exécution de jobs.

4.2 Importation des playbook sur Rundeck

En prolongement de la création du projet, Rundeck nous renvoie sur un onglet appelé « Job ». Cet onglet liste les différentes tâches du projet. C'est dans les paramètres du projet que l'on va configurer les nodes auxquels peuvent s'appliquer les jobs. Il existe pour cela un onglet de configuration « Nodes » comportant sune page appelée « Sources » dans laquelle on va pouvoir importer les machines hôtes présentes dans les inventaires Ansible. Pour cela, Rundeck dispose d'un module appelé « Ansible Resource Model Source ». On configure cette source de nodes en lui donnant le chemin du fichier de configuration de base d'Ansible, l'inventaire que l'on veut importer, le type de connexion SSH que l'on utilisera pour se connecter aux nodes, l'utilisateur SSH et le chemin du fichier contenant les clés SSH.

Une fois que l'importation des nodes est effectuée, on peut passer à celle des jobs.
L'implémentation des playbooks Ansible aux jobs de Rundeck se fait de manière assez similaire à celle des nodes : il existe un module de base pour les exécuter. Suite à la création du job, on le nomme puis on édite son « Workflow », la liste de tâche qu'il aura à exécuter. Chaque tâche est appelée « étape ». On va donc rajouter une étape au job, en sélectionnant le module « Ansible Playbook Workflow Node Step ». Ce module permet d'exécuter des playbooks Ansible préexistants. Il existe également un module permettant d'écrire un playbook Ansible directement dans l'interface web de Rundeck, le module « Ansible Playbook Inline Workflow Node Step ».

La configuration chaque étape nécessite que l'on fournisse le chemin d'accès du fichier de configuration d'Ansible, le chemin du Playbook que l'on veut exécuter, les informations de connexion SSH (type de connexion, utilisateur sur lequel se connecter, chemin des clés).

Le processus demande ensuite de gérer les nodes sur lesquels le job va agir. Dans la configuration du job, il y a un onglet « Nodes » dans lequel on peut sélectionner les nodes préalablement importées. On peut également exclure certaines nodes de l'exécution. Par exemple, une node « localhost » est toujours présente par défaut ; l'exclure de l'exécution est préférable (cf. Figure 11).

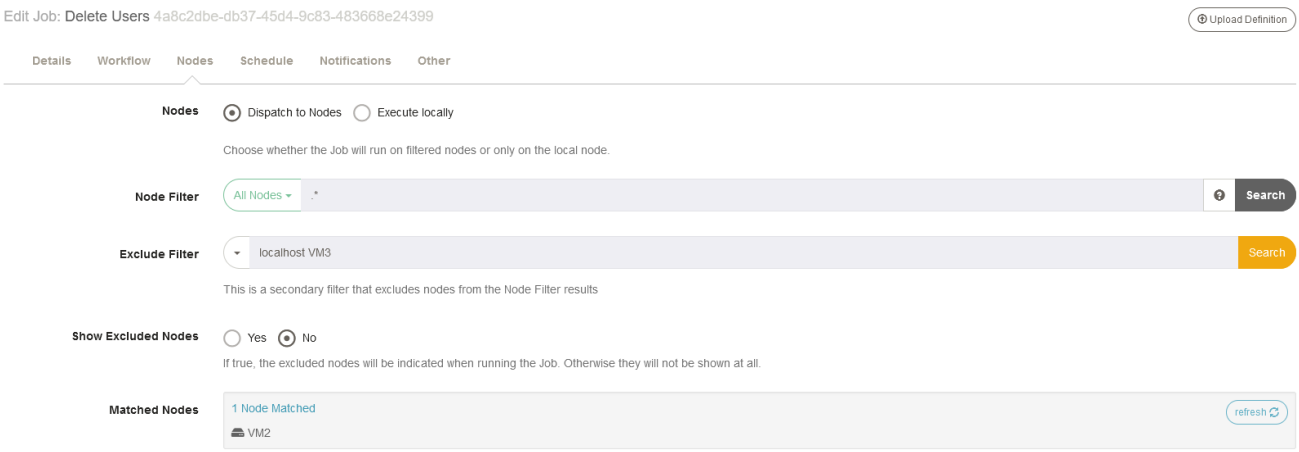

Figure 11 : Configuration de node sur le Job "Delete Users" avec nodes localhost et VM3 désactivées

Le job peut maintenant être exécuté. Cependant le fonctionnement des playbooks à importer fonctionne sur l'entrée de variables utilisateur (par exemple, pour le playbook « deleteusers », l'utilisateur doit rentrer la liste des comptes qu'il veut supprimer sur les machines hôtes ainsi qu'une variable booléenne indiquant s'il veut supprimer leurs répertoires de travail). Pour pallier ce problème, Rundeck permet de créer des « options » sous la forme de formulaires que l'utilisateur doit compléter avant d'exécuter le job.

On a donc, par exemple, pour le playbook de création d'utilisateur (cf. Figure 12) :

- Une option obligatoire demandant le nom du nouveau compte
- Une option obligatoire demandant l'adresse mail associée au compte
- Une option facultative demandant le shell utilisé par le compte (par défaut, l'option sera « /bin/bash »).
- Une option obligatoire demandant le mot de passe des droits super-utilisateur pour que Rundeck et Ansible puissent exécuter la tâche.

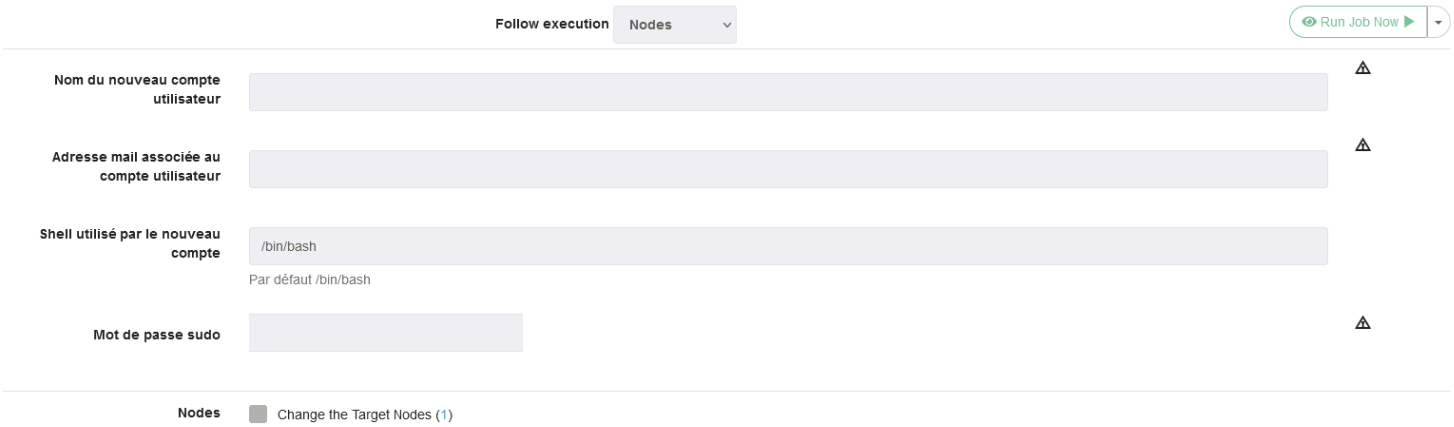

Figure 12 : Formulaire avant exécution du job de création d'utilisateur

J'ai donc importé les trois playbooks créés précédemment avec Ansible selon cette méthode de formulaire utilisateur.

5 Conclusion

Ces dix semaines de stage réalisées au sein de Fundvisory m'auront beaucoup apporté, tant sur le côté technique qu'au niveau humain.

Il m'a d'abord permis de mettre en avant les connaissances acquises en administration système et en automatisation des tâches que j'ai pu approfondir via l'apprentissage d'Ansible et Rundeck.

J'ai très souvent été amené à travailler en autonomie, notamment lors des recherches sur les logiciels ou les formats utilisés. J'ai ainsi pu développer ma méthodologie de travail.

Bien entendu, réaliser ce stage en distanciel en raison de la crise sanitaire ne m'a pas permis de connaître la vie en entreprise et les rapports humains sont différents. Malgré tout, cela m'a conduit à aller chercher l'information et faire preuve d'initiatives ce qui, je pense, a été apprécié par mes responsables.

Ce stage m'a également permis de confirmer l'intérêt que je porte à l'administration système et au développement informatique.

6 Remerciements

Tout d'abord, je tiens à remercier Mickael Hosdez et Arnaud Hermand, pour leur accueil chaleureux au sein de l'entreprise et pour leur patience face à mes nombreuses questions au cours de ce stage.

Je souhaite souligner que l'ensemble du personnel de Fundvisory a été extrêmement accueillant et bienveillant pour pallier les conditions plus difficiles liées au travail imposé en distanciel.

J'adresse également des remerciements à Sabri Bettis, étudiant en alternance chez Fundvisory, qui m'a beaucoup aidé à comprendre la logique derrière le format YAML et Ansible.

Enfin, je remercie le corps enseignant de l'IUT de Réseaux et Télécommunications pour l'attention dont ils ont fait preuve durant ces deux dernières années marquées par la crise sanitaire.

7 Glossaire

SSH : Secure Shell, protocole de communication permettant l'accès à une console de commande d'une machine distante.

INI : format de fichier texte créé par Microsoft en 1985.

YAML : format de fichier texte libre créé en 2001. Il est en grande partie basé sur les listes et les tableaux associatifs.

Clés SSH : méthode d'authentification SSH permettant de se connecter à une machine distante via une « phrase de passe », aussi appelée passphrase, pour éviter de taper le mot de passe du compte auquel on se connecte sur la machine distante. L'authentification se fait après vérification de deux clés (une privée et une publique) enregistrées auparavant sur les machines, par les deux ordinateurs.

Machines virtuelles : Logiciel permettant d'imiter le comportement d'un véritable ordinateur.

Shell : Logiciel fournissant une interface utilisateur à une machine. Il va interpréter les commandes que l'utilisateur exécute.

Super-utilisateur : Utilisateur possédant tous les droits sur un ordinateur. Aussi appelé « utilisateur root ».

Ping : Commande envoyant un paquet d'une machine vers une autre pour vérifier si elles sont connectées.

8 Bibliographie

https://docs.ansible.com – Documentation officielle d'Ansible

https://docs.rundeck.com – Documentation officielle de Rundeck