

Vagrant et Ansible

Raed Abdennadher, Jean-Etienne Delucinge, Steven Liatti, Gaëtan Ringot

Cours de Virtualisation des SI - Prof. Massimo Brero

Hepia ITI 3^{ème} année

21 janvier 2018



ANSIBLE

Table des matières

1	Vagrant	3
1.1	Qu'est-ce que Vagrant ?	3
1.2	Recherches et tests avec Vagrant	3
1.3	Commandes et scripts pour CentOS 7	4
1.4	Déploiement et configuration	5
2	Ansible	11
2.1	Introduction	11
2.1.1	Qu'est-ce que Ansible ?	11
2.1.2	But final	11
2.1.3	Version utilisée (Ansible et OS)	11
2.1.4	Format à respecter impérativement en cas de création d'un fichier YAML (.yml)	11
2.2	Accès SSH sur les machines	11
2.3	Principe des groupes d'utilisateurs/machines	12
2.4	Utilisation de playbooks	12
2.4.1	Condition When dans ansible	13
2.4.2	Pourquoi on l'utilise ?	13
2.4.3	Rôles (extrait de buzut.fr)	13
2.4.4	Pourquoi un/des rôle(s)	13
2.4.5	Génération du rôle	13
2.5	Bibliographie	14

Table des scripts de code

1	Script Vagrant et libvirt	4
2	vagrant init	5
3	Vagrantfile	5
4	Vagrantfile centos	5
5	Arborescence des fichiers pour Vagrant	6
6	vagrant.sh	7
7	create.sh	8
8	Vagrantfile debian	8
9	bootstrap.sh	8
10	interfaces	9
11	proxy	9
12	apt.conf	9
13	resolv.conf	9
14	sshd_config	10
15	SSH Ansible	11
16	Utilisateurs Ansible	12
17	Ping	12
18	Service	12
19	Variables	13
20	Gestion des rôles	14
21	Appel d'un rôle	14

1 Vagrant

Nous (Raed et Steven) avons commencé à explorer [Vagrant](#) avec [KVM](#) sous l'impulsion du groupe Cluster KVM (Sylvain, Loïc, Nathanaël et Mayron). Ils aimeraient pouvoir déployer et configurer leurs VMs automatiquement.

1.1 Qu'est-ce que Vagrant ?

Vagrant est un outil de gestion du cycle de vie de machines virtuelles. Il permet de créer et de lancer une ou plusieurs machines virtuelles pré-configurées selon des critères définis à l'avance dans des fichiers de configuration (Vagrantfile).

1.2 Recherches et tests avec Vagrant

Le problème initial est que Vagrant peut déployer des machines virtuelles uniquement pour VirtualBox, Hyper-V et VMware par défaut. Nous avons alors cherché un moyen de déployer pour KVM. Un plugin/librairie existe, [vagrant-libvirt](#) (provider pour Vagrant). Nous avons commencé la lecture de cette doc. Nous avons utilisé la version 1.9.1 de Vagrant, qui n'est pas la dernière en date et qui est compatible avec libvirt. Nous avons réalisé toutes ces manipulations sur nos machines personnelles car la plupart du temps les droits root étaient requis et qu'il n'y avait pas forcément les bonnes versions des softs. Nous avons eu de la peine à utiliser Vagrant avec libvirt et KVM car Vagrant n'est pas compatible par défaut avec KVM. Nous avons cherché plusieurs tutoriels et sommes tombés sur cet unique script qui avait l'air prometteur. Avec ce script nous voulions transformer notre image `centos-7-client1-disk1.vmdk` existante pour la donner au groupe KVM (voir ligne 12 dans le code suivant, `centos-7-client1-disk1.vmdk`).

```
1 BOX_NAME=vagrant-build
2 BASE_DIR="`pwd`/machines"
3 BOX_DIR="${BASE_DIR}/${BOX_NAME}"
4
5 mkdir -p ${BASE_DIR}
6
7 VBoxManage createvm --name "${BOX_NAME}" --ostype RedHat_64 --basefolder
  ${BASE_DIR}
8 VBoxManage registervm "${BOX_DIR}/${BOX_NAME}.vbox"
9
10 mkdir -p tmp
11 rm -rf tmp/clone.vdi
12 VBoxManage clonehd centos-7-client1-disk1.vmdk tmp/clone.vdi --format vdi
13 VBoxManage modifyhd tmp/clone.vdi --resize 20480
14 VBoxManage clonehd tmp/clone.vdi "${BOX_DIR}/${BOX_NAME}.vmdk" --format vmdk
15 VBoxManage -q closemedium disk tmp/clone.vdi
16 rm -f tmp/clone.vdi
17
18 VBoxManage storagectl "${BOX_NAME}" --name LsiLogic --add scsi --controller
  LsiLogic
19 VBoxManage storageattach "${BOX_NAME}" --storagectl LsiLogic --port 0
  --device 0 --type hdd --medium "${BOX_DIR}/${BOX_NAME}.vmdk"
20
21 VBoxManage setextradata "${BOX_NAME}"
  "VBoxInternal/Devices/e1000/0/LUN#0/Config/SSH/Protocol" TCP
```

```

22 VBoxManage setextradata "${BOX_NAME}"
   "VBoxInternal/Devices/e1000/0/LUN#0/Config/SSH/GuestPort" 22
23 VBoxManage setextradata "${BOX_NAME}"
   "VBoxInternal/Devices/e1000/0/LUN#0/Config/SSH/HostPort" 22222
24
25 VBoxManage modifyvm "${BOX_NAME}" --usb on --usbhci on
26 VBoxManage modifyvm "${BOX_NAME}" --memory 512
27
28 VBoxManage startvm "${BOX_NAME}" #--type headless
29
30 echo "Sleeping to give machine time to boot"
31 sleep 240
32
33 echo "Uploading ssh key & creating vagrant user"
34 cat ~/.ssh/id_rsa.pub | ssh -p 22222 root@localhost
   "umask 077; test -d .ssh || mkdir .ssh; cat >> .ssh/authorized_keys"
35 ssh -p 22222 root@localhost <<EOT
   useradd vagrant
   echo vagrant | passwd vagrant --stdin
   umask 077
   test -d /home/vagrant/.ssh || mkdir -p /home/vagrant/.ssh
   cp ~/.ssh/authorized_keys /home/vagrant/.ssh
   chown -R vagrant:vagrant /home/vagrant/.ssh
36 EOT
37 scp -P 22222 templates/sudoers root@localhost:/etc/sudoers
38
39 echo -n "Waiting for machine to shutdown"
40 VBoxManage controlvm ${BOX_NAME} acpipowerbutton
41 while [ 'VBoxManage showvminfo --machinereadable ${BOX_NAME} | grep
   VMState=' != 'VMState="poweroff"' ]; do
42     echo -n .
43     sleep 1
44 done
45 echo "Done"
46 vagrant package --base ${BOX_NAME} --output ${BOX_NAME}.box

```

Listing 1 – Script Vagrant et libvirt

Ce script a fonctionné partiellement, car il ne prenait pas l'image qu'on lui a fournit, mais une image par défaut. **C'est pour cela que nous avons abandonné l'idée d'utiliser Vagrant avec KVM et libvirt.** Nous avons toutefois remarqué (27.11.2017) que la librairie a été mise à jour ces derniers jours et, qu'apparemment, elle est compatible avec la dernière version de Vagrant (2.0.1).

1.3 Commandes et scripts pour CentOS 7

Vagrant utilise des "box" comme images de base, qu'on peut lui fournir en local (typiquement des fichiers iso ou .img) ou depuis [Vagrant Cloud](#). Dans notre cas, nous avons utilisé l'image centos/7. Voici une marche à suivre pour utiliser Vagrant avec VirtualBox. Tout d'abord, nous créons un dossier pour Vagrant :

```

1 $ mkdir vagrant_folder
2 $ cd vagrant_folder
3 $ vagrant init

```

Listing 2 – vagrant init

vagrant init crée un fichier de configuration nommé Vagrantfile, dont voici le contenu initial :

```

1 Vagrant.configure("2") do |config|
2   config.vm.box = "base"
3 end

```

Listing 3 – Vagrantfile

Ensuite, nous exécutons la commande `vagrant box add centos/7`, qui télécharge la box en question et nous permet de choisir le provider avec lequel nous allons travailler. Une fois cette tâche terminée (téléchargement), devons normalement obtenir le message suivant :

```

1 ==> box: Successfully added box 'centos/7' (v1710.01) for 'VirtualBox'!

```

Nous pouvons maintenant commencer à éditer notre fichier Vagrantfile comme ceci :

```

1 Vagrant.configure("2") do |config|
2   config.vm.box = "centos/7"
3 end

```

Listing 4 – Vagrantfile centos

Finalement, nous lançons la machine virtuelle avec la commande suivante : `vagrant up`. Nous constatons que par défaut le port pour SSH est le 2222, que l'IP est la 127.0.0.1 et que `username = vagrant` et utilise une paire de clé pour l'authentification. Finalement nous pouvons nous connecter à la machine virtuelle avec la commande suivante : `vagrant ssh`

1.4 Déploiement et configuration

Nous avons continué nos expériences avec Vagrant avec VirtualBox comme provider. Nous avons commencé par essayer avec une image CentOS, mais arrivés à l'étape de la configuration du serveur SSH, nous n'avons pas réussi à nous connecter. D'après l'erreur survenue (*ssh_exchange_identification: read: connection reset by peer*) et [nos recherches sur le web](#), il semblerait que ce soit une incompatibilité entre le client SSH des machines hepia et du serveur SSH CentOS. Nous avons finalement réussi à installer, configurer et lancer une machine virtuelle sous Debian (Jessie) et pouvoir s'y connecter en SSH avec l'utilisateur vagrant. Voici les fichiers de configuration et les scripts.

Pour lancer toute la séquence, il faut exécuter `vagrant.sh`. Nous allons maintenant décrire la séquence complète et dans l'ordre d'exécution.

```

1  .
2  |-- debian
3  |   |-- bootstrap.sh
4  |   |-- conf_files_vm
5  |   |   |-- apt.conf
6  |   |   |-- interfaces
7  |   |   |-- proxy
8  |   |   |-- pub_key_server_ansible
9  |   |   |-- resolv.conf
10 |   |   |-- sshd_config
11 |   |-- create.sh
12 |   |-- Vagrantfile
13 |-- vagrant.sh

```

Listing 5 – Arborescence des fichiers pour Vagrant

Ce fichier initial (6) va copier le dossier <template> de l'OS voulu pour la VM dans le dossier <folder_name>. En effet, si on désire créer plusieurs machines virtuelles, il faut créer un dossier pour vagrant par machine virtuelle. Ce script va appeler le script suivant, `create.sh`, dans le dossier <folder_name>, en lui passant le nom de la VM, son IP, le nombre de CPU et la quantité de RAM voulus. Finalement, une fois que la VM a été configurée et redémarrée, on lance le script Ansible depuis la VM serveur Ansible (cf. doc Ansible). Cette dernière étape est nécessaire uniquement car nous n'avons pas les droits pour installer Ansible sur la même machine physique de l'école où est installé Vagrant.

```

1  #!/bin/bash
2
3  if [[ $# != 6 ]]; then
4      echo
5      "Usage: ./vagrant.sh <template> <folder_name> <machine_name> <ip> <cpu> <ram>"
6      echo "<folder_name> is like: debian_apache"
7      echo "<machine_name> is like: vagrant_debian_client"
8      echo "<ip> is like: 10.194.184.196"
9      echo "<cpu> is like: 2"
10     echo "<ram> is like: 512"
11     exit 1
12 fi
13
14 TEMPLATE=$1
15 FOLDER_NAME=$2
16 MACHINE_NAME=$3
17 IP=$4
18 CPU=$5
19 RAM=$6
20
21 VBoxManage controlvm $MACHINE_NAME poweroff
22 VBoxManage unregistervm $MACHINE_NAME --delete
23
24 rm -rf $FOLDER_NAME
25 mkdir $FOLDER_NAME

```

```

25 cp -r debian/* $FOLDER_NAME
26 cd $FOLDER_NAME
27 ./create.sh $MACHINE_NAME $IP $CPU $RAM
28 cd ..
29 rm -rf $FOLDER_NAME
30
31
32 ##### Ansible part #####
33 ssh root@10.194.184.190
"ansible-playbook /etc/ansible/roles/main_service.yml -e \"service=Web port_http=80 dom

```

Listing 6 – vagrant.sh

Script de création (7) : on commence par modifier le fichier interfaces et Vagrantfile avec les arguments fournis (IP, nom, CPU et RAM) avec l'utilitaire sed (lignes 21 à 24). On continue par supprimer si besoin une éventuelle image ayant le même nom (ligne 26), puis on lance l'installation (appel à Vagrantfile) (ligne 27) et une fois l'installation terminée, nous arrêtons la machine (ligne 28). Les deux dernières lignes concernent VirtualBox, la 1ère permet à la machine virtuelle d'accéder au LAN (mode bridge) et la dernière redémarre la VM.

```

1  #!/bin/bash
2
3  if [[ $# != 4 ]]; then
4      echo "Usage: ./create.sh <machine_name> <ip> <cpu> <ram>"
5      echo "<machine_name> is like: vagrant_debian_client"
6      echo "<ip> is like: 10.194.184.196"
7      echo "<cpu> is like: 2"
8      echo "<ram> is like: 512"
9      exit 1
10 fi
11
12 NICTYPE="82540EM"
13 ADAPTER=enp0s31f6
14 FOLDER=conf_files_vm
15
16 MACHINE_NAME=$1
17 IP=$2
18 CPU=$3
19 RAM=$4
20
21 sed -i -e "s/\(address \).*\/\1$IP/" $FOLDER/interfaces
22 sed -i -e "s/\(vb.name = \).*\/\1\"$MACHINE_NAME\"/" Vagrantfile
23 sed -i -e "s/\(vb.memory = \).*\/\1\"$RAM\"/" Vagrantfile
24 sed -i -e "s/\(vb.cpus = \).*\/\1\"$CPU\"/" Vagrantfile
25
26 vagrant destroy
27 vagrant up
28 vagrant halt
29 VBoxManage modifyvm $MACHINE_NAME --nic1 bridged --nictype1 $NICTYPE
--bridgeadapter1 $ADAPTER

```

```
30 VBoxManage startvm $MACHINE_NAME --type headless
```

Listing 7 – create.sh

Ce fichier (8) est le fichier de configuration d'une VM par Vagrant. Sont définis le nom de l'image (téléchargée sur [Vagrant Cloud](#)) (ligne 2), le script shell qui sera exécuté à la fin de l'installation (ligne 3), le provider et le nom de la VM (lignes 3 et 4) et la RAM et le CPU (lignes 6 et 7). À noter que tout le contenu du dossier où se trouve ce fichier Vagrantfile sera copié à la racine de la VM, dans le dossier /vagrant.

```
1 Vagrant.configure("2") do |config|
2   config.vm.box = "debian/jessie64"
3   config.vm.provision :shell, path: "bootstrap.sh"
4   config.vm.provider :virtualbox do |vb|
5     vb.name = "vagrant_debian_client"
6     vb.memory = 1024
7     vb.cpus = 2
8   end
9 end
```

Listing 8 – Vagrantfile debian

Script (9) exécuté lorsque l'installation de la machine virtuelle est terminée, par elle-même. Nous copions tous les fichiers de configuration pré-écrits et mettons à jour les applications installées, ceci faisant également office de test de connectivité internet. La clé publique est celle du serveur Ansible.

```
1 #!/bin/bash
2
3 FOLDER=conf_files_vm
4
5 cp /vagrant/$FOLDER/interfaces /etc/network/interfaces
6 cat /vagrant/$FOLDER/proxy >> /etc/profile
7 cp /vagrant/$FOLDER/apt.conf /etc/apt/apt.conf
8 cp /vagrant/$FOLDER/resolv.conf /etc/resolv.conf
9 cat /vagrant/$FOLDER/pub_key_server_ansible >>
  /home/vagrant/.ssh/authorized_keys
10
11 sudo apt update
12 #sudo apt -y upgrade
13 sudo apt -y install net-tools nano htop
14
15 mv /etc/ssh/sshd_config /etc/ssh/sshd_config_old
16 cp /vagrant/$FOLDER/sshd_config /etc/ssh/sshd_config
```

Listing 9 – bootstrap.sh

Fichier de configuration des interfaces réseau pour Debian.

```
1 source /etc/network/interfaces.d/*
2
3 auto lo
4 iface lo inet loopback
5
6 auto eth0
7 iface eth0 inet static
8     address 10.194.184.196
9     netmask 255.255.255.0
10    gateway 10.194.184.1
```

Listing 10 – interfaces

Configuration du proxy (contrainte du réseau de l'hepia).

```
1 MY_PROXY="http://129.194.185.57:3128/"
2
3 HTTP_PROXY=$MY_PROXY
4 HTTPS_PROXY=$MY_PROXY
5 FTP_PROXY=$MY_PROXY
6 http_proxy=$MY_PROXY
7 https_proxy=$MY_PROXY
8 ftp_proxy=$MY_PROXY
9
10 export HTTP_PROXY HTTPS_PROXY FTP_PROXY http_proxy https_proxy ftp_proxy
```

Listing 11 – proxy

Définition du proxy pour apt également.

```
1 Acquire::http::proxy "http://129.194.185.57:3128/";
2 Acquire::https::proxy "https://129.194.185.57:3128/";
3 Acquire::ftp::proxy "ftp://129.194.185.57:3128/";
```

Listing 12 – apt.conf

Fichier de configuration DNS (mauvaise adresse par défaut).

```
1 Acquire::http::proxy "http://129.194.185.57:3128/";
2 Acquire::https::proxy "https://129.194.185.57:3128/";
3 Acquire::ftp::proxy "ftp://129.194.185.57:3128/";
```

Listing 13 – resolv.conf

Configuration du serveur SSH sur Debian. Le seul changement qui a été fait par rapport à la version par défaut incluse dans cette image Debian se trouve à la dernière ligne, PasswordAuthentication yes au lieu de PasswordAuthentication no, pour autoriser la connexion SSH avec mot de passe.

```

1 Port 22
2 Protocol 2
3 HostKey /etc/ssh/ssh_host_rsa_key
4 HostKey /etc/ssh/ssh_host_dsa_key
5 HostKey /etc/ssh/ssh_host_ecdsa_key
6 HostKey /etc/ssh/ssh_host_ed25519_key
7 UsePrivilegeSeparation yes
8
9 KeyRegenerationInterval 3600
10 ServerKeyBits 1024
11 SyslogFacility AUTH
12 LogLevel INFO
13
14 LoginGraceTime 120
15 PermitRootLogin without-password
16 StrictModes yes
17
18 RSAAuthentication yes
19 PubkeyAuthentication yes
20 AuthorizedKeysFile %h/.ssh/authorized_keys
21
22 IgnoreRhosts yes
23 RhostsRSAAuthentication no
24 HostbasedAuthentication no
25
26 PermitEmptyPasswords no
27 ChallengeResponseAuthentication no
28 X11Forwarding yes
29 X11DisplayOffset 10
30 PrintMotd no
31
32 PrintLastLog yes
33 TCPKeepAlive yes
34 AcceptEnv LANG LC_*
35 Subsystem sftp /usr/lib/openssh/sftp-server
36
37 UsePAM yes
38 UseDNS no
39 PasswordAuthentication yes

```

Listing 14 – sshd_config

Une fois que toutes ces étapes sont terminées, nous pouvons nous connecter à cette machine virtuelle (depuis le réseau hepia), par cette commande et avec mot de passe **vagrant** :

```
ssh vagrant@10.194.184.xxx.
```

2 Ansible

2.1 Introduction

2.1.1 Qu'est-ce que Ansible ?

Ansible est un logiciel libre d'automatisation des applications et de l'infrastructure informatique. Déploiement d'application, gestion de Configuration et livraison en continue.

2.1.2 But final

Nous (Gaetan et Jean-Etienne) avons commencé à explorer Ansible et ces possibilités pour pouvoir déployer des installations, mises à jour, ... automatiquement dans un réseau pour pouvoir l'intégrer avec Vagrant (Read et Steven). Au final nous devons déployer une machine virtuelle avec un service et des paramètres donner.

2.1.3 Version utilisée (Ansible et OS)

Pour notre projet nous avons installés un CentOS 7 avec une configuration minimale, dans laquelle nous avons par la suite ajouté :

- EPEL release
- Ansible (version 2.4.1.0)
- Python (version 2.7.5)
- Nano

Attention, la mise à jour d'Ansible peut fortement créer des soucis de compatibilités.

2.1.4 Format à respecter impérativement en cas de création d'un fichier YAML (.yml)

- Chaque fichier doit commencer par "—" (3 tirets normaux) hors rôle
- L'indentation est très importante, utiliser uniquement des **ESPACES**
- Suivant l'OS, il est important de les différencier, car les commandes ne sont pas multiplate-formes et entraine la non-fonctionnalité du script ansible

2.2 Accès SSH sur les machines

Si des problèmes de connexion en ssh arrivent , utiliser les commandes suivantes :

```
1 $ ssh-add
2 $ ssh-keygen -t rsa -C "ansible@ip_ansible"
3 $ ssh-copy-id user@ipduclient
```

Listing 15 – SSH Ansible

Permettent de régler les problèmes en créant puis copiant des clés SSH afin de par la suite, ne pas avoir de problèmes d'identification. **À noter** : la génération de clés n'est pas nécessaires si déjà effectué une fois.

2.3 Principe des groupes d'utilisateurs/machines

Il est à noter que le fichier se situant dans le dossier `/etc/ansible/hosts` se trouvent les différents groupes de machines et utilisateurs pour ansible. Un utilisateur PEUT être dans PLUSIEURS groupes différents. Lors d'une commande ansible, il est possible de faire appel à un certain groupe, en rajoutant celui-ci à la fin de la commande ansible. Exemple avec le groupe **Users** :
`ansible -m ping Users`

Afin de rajouter un groupe, il faut reprendre la syntaxe suivante :

```
1 [nom_groupe]
2 nom_host1 ansible_ssh_host= adresse ip 1
3 nom_host2 ansible_ssh_host= adresse ip 2
4 nom_host3 ansible_ssh_host= adresse ip 3
```

Listing 16 – Utilisateurs Ansible

Mais attention, il faut penser à créer le fichier `/etc/ansible/group_vars/nom_groupe`. Il contient toutes les informations de configuration de connexion, comme `ansible_ssh_user`, qui n'est autre que le nom d'utilisateur utilisé pour se connecter au groupe en question.

2.4 Utilisation de playbooks

Pour lancer un playbook dans Ansible il faut exécuter la commande suivante :
`ansible-playbook chemin_du_fichier/Nom_du_fichier -e "nom_variable=valeur_var"`.

Valeurs importantes à mettre dans le fichier en question :

- `hosts` : Qui indique le groupe ou la machine qui exécutera nos futures tâches/services
- `remote_user` : nom d'utilisateur utilisé pour exécuter les tâches en question.
- `tasks` : Indiquant le début de la liste des tâches qui vont être exécutées par la suite.

Attention ! : `hosts` a priorité sur `remote_user` !

Exemple simple avec un ping :

```
1 ---
2 - hosts: Users
3   remote_user: root
4
5   tasks:
6     - ping:
```

Listing 17 – Ping

Dans le cadre d'un service, celui-ci doit être appelé grâce à la façon suivante :

```
1 - service:
2   name: Nom_du_service_à_exécuter
3   state: Etat_futur_du_service (started, stopped, restarted, reloaded)
```

Listing 18 – Service

Il est possible de récupérer des variables mises avec l'option -e (voir plus haut) pour ce faire le formatage du fichier se fait de la façon suivante :

- Nom_variable : "Nom_variable" = peut récupérer une liste/tableau
- Nom_variable : "Nom_variable" = peut récupérer une string

```
1  ---
2  vars:
3      Nom_variable: "{{ Nom_variable }}"
4
5  tasks:
6  - include: tasks1.yml
7  when: 1
8
9  - include: tasks2.yml
10 when: 2
```

Listing 19 – Variables

2.4.1 Condition When dans ansible

Avec une condition when on peut faire un "if" avec une condition, ce qui permet d'appeler des rôles/playbook, de faire des tâches et etc selon une condition.

2.4.2 Pourquoi on l'utilise ?

Nous avons utilisé la condition "ansible_os_family" (intégrer dans les services d'ansible) pour différencier par rapport à la famille de l'OS pour utiliser les bonnes commandes liées à celui-ci. Nous avons aussi utilisé pour notre variable "service" qui permet de savoir quel service installer (Web,dhcp,...).

2.4.3 Rôles (extrait de buzut.fr)

Les rôles représentent une manière d'abstraire les directives includes. Grâce aux rôles, il n'est plus utile de préciser les divers includes dans le playbook, ni les paths des fichiers de variables etc. Le playbook n'a qu'à lister les différents rôles à appliquer. En outre, depuis les tasks du rôle, l'ensemble des chemins sont relatifs. Inutile donc de préciser le nom du fichier suffit, Ansible s'occupe du reste.

2.4.4 Pourquoi un/des rôle(s)

Car cela nous permet de généraliser une installation par service (web,dhcp,..) plus facilement et réutilisable par d'autres personnes !

2.4.5 Génération du rôle

Pour générer un rôle, il faut utiliser "ansible-galaxy + nom + init". Il va générer les dossiers/fichiers de base pour notre rôle. Notre dossier contiendra les dossiers/fichiers suivants :

```

1 - Roles
2   - "Nom du role"
3     - Defaults : les variables par défaut qui seront à disposition du
rôle.
4       - main.yml
5     - Vars : Variables à disposition du rôle cependant elles ont vocation
à être modifiées par l'utilisateur et elles prennent le dessus sur celle du
dossier defaults si elles sont renseignées.
6       - main.yml
7     - Tasks : ici on renseigne nos taches (comme dans un playbooks
normal).
8       - main.yml
9     - Meta : Sert à renseigner les dépendances liées à nos rôles (ssl, et
etc).
10      - main.yml
11     - README : renseigne sur comment utilisé les rôles, variables à
définir et etc.

```

Listing 20 – Gestion des rôles

On appelle un rôle de la façon suivante :

```

1 ---
2 tasks:
3 - include_role: role.yml

```

Listing 21 – Appel d'un rôle

2.5 Bibliographie

- <https://github.com/ansible/ansible/issues/19584>
- https://docs.ansible.com/ansible/latest/yum_module.html
- https://docs.ansible.com/ansible/latest/service_module.html
- https://docs.ansible.com/ansible/latest/apt_module.html
- https://docs.ansible.com/ansible/latest/apt_repository_module.html
- <https://serversforhackers.com/c/an-ansible-tutorial>
- <https://buzut.fr/tirer-toute-puissance-dans-ansible-roles/>