

TP/TD gestion des configurations: Powerfull Ansible !

Jean-Marc Pouchoulon

Janvier 2022



1 Objectifs du TP et organisation

1.1 Les compétences à acquérir à la fin de cette séance sont les suivantes :

- Positionner le rôle d'Ansible vis à vis d'autres outils de configuration management comme Docker ou Puppet ;
- Acquérir le vocabulaire d'Ansible (inventaire, playbook, rôles...) ;
- Paramétrer d'Ansible (ansible.cfg) ;
- Effectuer une commande sur un serveur ou un groupe de serveurs (commande ansible, ansible-console) ;
- Réaliser une boucle avec Ansible (instruction when) ;
- Utilisez le mécanisme de notification dans un playbook ;
- Réaliser un test conditionnel avec Ansible (instruction when) afin de créer un playbook basique destiné à fonctionner sur plusieurs distributions ;
- Utiliser un rôle existant ;
- Contextualisation : Orchestrer des switches Arista via Ansible ;
- Contextualisation : Orchestrer une machine Windows ;

1.2 Organisation, recommandations et notation du TP

Vous travaillerez individuellement.

Un compte rendu succinct (fichiers de configuration , copie d'écran montrant la réussite de la construction ...) est demandé et à rendre sur Moodle obligatoirement mais l'évaluation se fera au travers d'un QCM portant sur le cours et l'état d'avancement du TP.

1.3 Pré-requis

- Avoir une connaissance de base du protocole ssh et de son implémentation sous Linux (clefs publiques et privé , passphrase) ;
- Connaître mécanisme de la commande sudo ;
- Bases du "langage" YAML ;

- Bases de la programmation (boucle, conditions) ;
- Bases de l'administration des systèmes sous Linux et Windows ;
- Bases réseaux (adressage) ;

1.4 Environnement de travail conseillé

L'utilisation de vscode avec une extension ansible¹ et un linter (ansible-lint) est conseillée. Vous utiliserez la machine virtuelle debianvm issue de <http://store.iutbeziers.fr> sur laquelle est déjà installé Ansible (directory /home/ansible).

2 Présentation d'Ansible

2.1 Généralités

Ansible est un outil de **gestion de configuration** opensource adossé à la société RedHat. C'est un outil développé en Python. Il excelle dans l'automatisation et à ce titre fait partie de la panoplie du DevOps².

Son champ d'application est très large et sa force réside dans un catalogue très fourni de modules portant sur tous les domaines des systèmes, des réseaux et plus généralement de l'infrastructure. De nombreuses entreprises liées au secteur de l'informatique maintiennent des modules relatifs à leurs matériels sur le site ansible-galaxy.

Les outils de gestion de configuration visent à lutter contre des dérives appelées communément "drift" : quand deux administrateurs réseaux interviennent indépendamment l'un de l'autre sur un même parc vous aurez deux configurations différentes (une pour chaque ingénieur). Il est donc important que les interventions soient normées pour éviter ces dérapages qui nuisent aux diagnostics et entraînent des surcoûts de maintenance.



FIGURE 1 – Le Drift (Dérapage) existe aussi dans la gestion des configurations !

Ansible combiné avec Git est un outil pour éviter ces dérapages.

1. <https://marketplace.visualstudio.com/items?itemName=tomaciazek.ansible>

2. On est sur un grand O car il est Ansible est plus utilisé par les opérationnels que par les développeurs

Quelques définitions ³ :

- Le "**provisioning**" englobe l'ensemble des opérations nécessaires afin de créer une infrastructure. L'"Infrastructure As Code" est un outil pour l'automatisation du provisioning.
- Le "**configuration management**" ou "gestion des configurations" est un processus destiné à maintenir les systèmes informatiques, les serveurs et les logiciels dans l'état souhaité et à préserver la cohérence de cet état. C'est une façon de s'assurer qu'un système fonctionne comme prévu au fil des changements effectués. La gestion des configurations s'applique après une opération de provisioning. (par exemple on installe un serveur via PXE et on utilise un outil comme Ansible pour la partie logicielle en rajoutant une stack LAMP ou LEMP).
- Le "**déploiement**" est un terme relatif au "déploiement du code". On parle de "déploiement continu" si le code est mis à jour et poussé dans la foulée en production. Mais on peut aussi mettre à jour du code en changeant de container et on est dans la gestion des configurations...
- L'"**orchestration**" agit au niveau processus : processus de mise en exploitation, processus de mise à jour d'applications ou de services IT au sens large (application web, DNS, messagerie) et donc sur plusieurs systèmes. L'orchestration peut être considéré plus comme un processus de maintien en condition. (Exemple de Kubernetes qui au travers d'une boucle d'ajustement amène le cluster dans l'état désiré)
- L'automatisation automatise le « provisioning », la « gestion des configurations », le déploiement de code.

Ansible est capable de provisionner et de configurer.

2.2 Les concepts d'Ansible

- *Ansible* se connecte d'un nœud de contrôle ("node manager"), en général au travers du protocole SSH, vers des nœuds inventoriés. Une commande ou un playbook saisis sur le "node manager" entraînent la copie du code nécessaire à l'exécution des commandes ou de playbooks vers les nœuds inventoriés. Ce code Python est alors exécuté sur ce nœud et le résultat du code est renvoyé vers le manager qui peut l'afficher sur sa console. En conséquence l'interpréteur Python ⁴ et un daemon SSH se doivent d'être présents sur tous les nœuds.

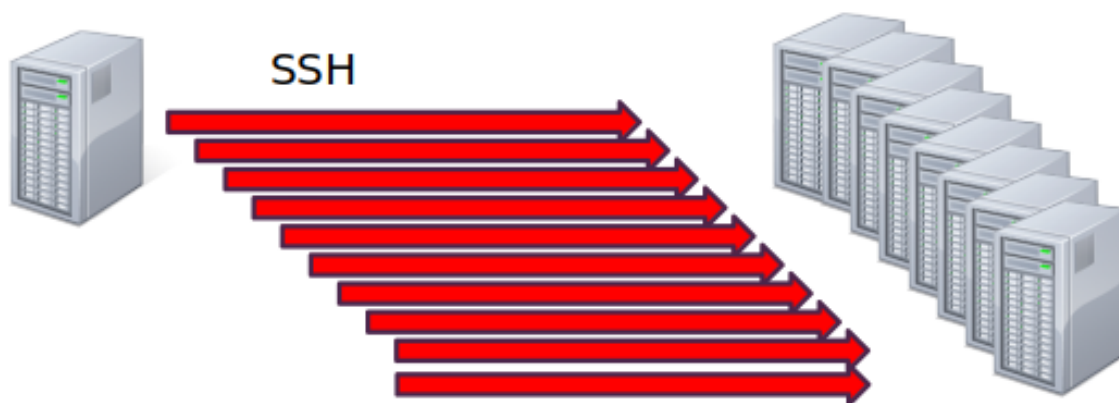


FIGURE 2 – Ansible c'est KISS(Keep It Simple Stupid) et dead simple

3. Ces définitions sont une tentative de classification elles ne sauraient être une vérité absolue du fait de la multiplicité des produits et des cas d'utilisation

4. version 3

SSH n'est pas le seul protocole utilisé par Ansible mais c'est le cas le plus courant⁵. C'est le protocole WinRM qui est le plus naturel en environnement windows afin de se connecter à un hôte distant.

- L'*inventaire* est un fichier au format "ini" ou une source externe⁶ permettant de décrire les nœuds inventoriés et des groupes de nœuds. Pour exemple ce fichier d'inventaire contient la liste des machines des groupes debian et centos. Dans cet exemple on indique l'IP de l'hôte, la clef privée SSH et le port de connexion pour chacun des serveurs :

```
[debian]
debian-0 ansible_host=10.213.0.1 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=2220
debian-1 ansible_host=10.213.0.2 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=2221
debian-2 ansible_host=10.213.0.3 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=2222
debian-3 ansible_host=10.213.0.4 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=2223
debian-4 ansible_host=10.213.0.5 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=2224

[centos]
centos-0 ansible_host=10.213.0.6 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=3220
centos-1 ansible_host=10.213.0.7 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=3221
centos-2 ansible_host=10.213.0.8 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=3222
centos-3 ansible_host=10.213.0.9 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=3223
centos-4 ansible_host=10.213.0.10 ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_port=3224
```

- Les *taches* : une tâche correspond à une action effectuée à l'aide d'un module.
- Les *modules* : un module est créé avec du code Python. Réutilisable, il va interagir avec une API ou un nœud distant afin de réaliser une tâche spécifique. Un module est appelé en ligne de commandes via la commande ansible ou dans un playbook.
- les *playbooks* : Il s'agit d'exécuter plusieurs tâches en même temps. Dans un fichier au format Yaml les tâches à réaliser sont listées séquentiellement :

```
---
- hosts: localhost
  tasks:
  - name: add one package
    ansible.builtin.apt:
      name: apache2
      state: present

  - name: Create a 2048-bit SSH key for user jsmith in ~student/.ssh/id_rsa_test
    ansible.builtin.user:
      name: student
      generate_ssh_key: true
      ssh_key_bits: 2048
      ssh_key_file: .ssh/id_rsa_test
```

Les deux tâches seront donc appliquées l'une après l'autre sur le nœud "localhost". Les deux modules sont des modules "builtin" de "Ansible core" et maintenu directement par les développeurs du produit.

- Les *templates* : ils permettent de générer dynamiquement des fichiers cibles au travers d'un "template" contenant des variables comme l'adresse IP d'un serveur. Elle sera utilisée par exemple afin de configurer un serveur WEB devant écouter sur l'adresse IP de chaque machine. Ansible utilise Jinja2, un gestionnaire de modèles⁷ écrit pour Python. Les «Templates» Jinja2 permettent de gérer des boucles, des tests logiques, des listes ou des variables. Les variables sont souvent issues des "facts" que le modèle setup permet de visualiser ;

5. La liste des outils est donnée par `ansible-doc -t connection -l`

6. un script peut être utilisé à cet effet permettant un inventaire dynamique

7. "Templates" en anglais

- Les *rôles* : permettent de proposer un ensemble ré-utilisable pour des actions complexes comme par exemple transformer un serveur en serveur de bases de données, ou toute autre fonction souhaitée. Les rôles rassemblent des variables, des playbooks dans une arborescence. La commande "ansible-galaxy" permet de trouver des rôles développés par la communauté Ansible.
- Les *collections* : c'est un format de distribution qui peut inclure des "playbooks", des rôles , des modules et des plugins.
- Les *plugins* sont des codes qui améliorent les fonctionnalités de "Ansible core".

2.3 Organisation de la distribution d'Ansible

Ansible se base sur des modules qui vont permettre de répondre à des besoins d'automatisation très divers. Pour exemple on peut citer la gestion des "packages", la création d'utilisateurs Unix, le pilotage des routeurs...

Ansible permet de passer des commandes via le CLI sur un ensemble d'éléments d'infrastructures (serveurs, switch , routeurs...)

```
# Utilisation du module ping pour vérifier la compatibilité de l'accès d'Ansible sur les machines du groupe debian:
ansible -m ping debian
```

Ce foisonnement de ressources a nécessité avec la version 3.0 d'Ansible une réorganisation des éléments de sa suite logicielle.

1. Ansible Base : c'est le runtime minimal d'Ansible. C'est le package python ansible-core qui l'installe.
2. Les collections : c'est le package python ansible qui les installe. On peut obtenir des collections à partir du site ansible galaxy

Saisissez la commande "ansible-galaxy collection list" pour lister les modules installés.

3 Installation et utilisation d'Ansible

3.1 Obtenir de l'aide

3.1.1 En ligne de commandes

L'utilitaire ansible-navigator permet entre autre d'obtenir de l'aide.

```
# obtenir de l'aide sur le module apt
ansible-navigator doc apt -m stdout
# ou
ansible-doc apt
```

3.1.2 Des url utiles :

Les pages suivantes peuvent vous être fort utiles lors du TP :

<https://docs.ansible.com/ansible/latest/collections/>

https://docs.ansible.com/ansible/latest/collections/index_module.html.

https://docs.ansible.com/ansible/latest/user_guide/

3.2 Installation de la VM

Instanciez une VM Debian (2 vcpu, 4 gigas de RAM si possible) à partir de l'OVA fournie par l'enseignant (voir <http://store.iutbeziers.fr>)

Dans la directory /home/ansible de votre machine virtuelle Debian vous trouverez les scripts nécessaires au fonctionnement du TP⁸.

8. Vous pouvez retrouver ces scripts via git clone <https://registry.iutbeziers.fr:11443/pouchou/tp3automatisation.git>, faite un git pull afin de prendre en compte les

Nous allons utiliser des containers Debian11 et Rocky 8 (qui est un "clone" de RedHat) comme environnement de formation. Ces containers Docker comprennent systemd. Dans cette configuration il est nécessaire de rétrograder les NameSpaces de votre machine virtuelle en version 1. Pour ce faire lancez la commande :

```
/home/ansible/revert2cgroupv1.sh
```

3.3 Installation et paramétrage de l'environnement du TP.

Ansible (version 2.12 au moins) est déjà installé sur votre VM. **Donc ne faites pas l'installation d'Ansible via apt** qui est souvent en retard de version.

Allez sous /home/ansible et faites un "git pull" pour être sûr d'avoir la dernière version du script. Lancez le script create-cont.sh. Il va générer 5 containers Debian, 5 containers Centos et 3 containers de switches L3 Arista (EOS).

Ces containers sont tous accessibles en ssh.

Le script configure /etc/ansible et /etc/hosts pour que vous puissiez accéder aux containers comme si vous étiez dans un environnement de production.

Les containers Linux sont accessibles directement via ssh et sans mot de passe grâce à la clef générée au début du TP :

```
ssh debian-0 # de 0 à 4
ssh rocky-0
```

Les containers ceos Arista n'ont pas besoin à ce stade du TP d'être accessibles et seront traités ensuite.

4 Prise en main d'Ansible

4.1 Vérification et "debug" basique

1. Vérifiez avec la commande *ansible* et le module *ping* que vos cibles Ansible sont vivantes. Si vous le souhaitez utilisez l'option "host_key_checking = false" dans le fichier ansible.cfg afin d'éviter des warnings désagréables si vous recréez les containers.
2. Utilisez l'utilitaire ansible-console pour lancer la commande 'ip a' sur toutes les machines.
3. Quelle est le protocole réseau utilisée par Ansible ?
4. Analyser le fonctionnement de la commande ansible avec l'option -vvv. Que pouvez vous en déduire du fonctionnement d'Ansible ? expliquer comment Ansible peut être "agentless" ?
5. Récupérez les facts contenant les adresses IP de tous les containers via un filtre.
6. Créez un groupe container qui regroupe tous les containers et vérifiez via les commandes :

```
ansible-inventory --list all
ansible-navigator
```

que le groupe est listé. Existe-t-il un équivalent par défaut ? un groupe listant les nœuds sans groupe ?

4.2 Installation d'Apache via les modules dnf et apt d'Ansible core

1. Sur deux des containers créés installez un serveur web Apache sous Debian et Centos à l'aide de la commande ansible et des modules Ansible de gestion des paquets apt et dnf. Le package s'appelle apache2 pour Debian et httpd pour Rocky Linux.
2. Le playbook suivant installera Apache et PHP, un fichier info.php et qui démarrera le serveur web Apache sur vos containers Debian.

```
--
- hosts: debian
  tasks:
    - name: Installer Apache
      ansible.builtin.apt:
        name: apache2
        state: present
        update_cache: true

    - name: Installer Php7
      ansible.builtin.apt:
        name: libapache2-mod-php7.4
        state: present

    - name: Démarrer le service Apache
      ansible.builtin.service:
        name: apache2
        state: started
        enabled: true

    - name: Copier le fichier phpinfo
      ansible.builtin.copy:
        src: info.php
        dest: /var/www/html/index.php
        owner: www-data
        group: www-data
        mode: 0664
```

3. Avant de le lancer avec la commande `ansible-playbook`, vérifiez le bon enchaînement des tâches avec les options `--check` et `--diff` (pas d'exécution réelle), `--list-hosts`, `--list-tasks` de la commande `ansible-playbook`. Testez aussi la conformité de votre playbook avec le "linteur" `ansible-lint`.
4. Générez un playbook équivalent pour les containers Rocky Linux.
5. Lister les faits à l'aide du module `setup`. A quoi servent ces faits ?
6. Récupérez les faits contenant les adresses IP de tous les containers via un filtre.
7. Créer une tâche nommée "reload" ⁹ qui "reload" le server web ¹⁰. Tagué cette tâche "relance" puis utilisez le tag pour n'exécuter que cette tâche dans le playbook.
8. A quoi servent les tags "never" et "always" (testez) ?
9. Utilisez les faits et les templates jinja2 afin de modifier la directive fichier `ports.conf` sous Debian pour que lors de l'installation du module `apache2` de Debian n'écoute que sur l'IP interne du container. Vous utiliserez les mots clefs `notify` et `handler` pour relancer le service `apache` dans le playbook. ¹¹

5 Programmation avec Ansible : conditions et boucles

1. Dans les playbooks créés précédemment installez avec une seule tâche les deux packages `apache` et la library `php` en utilisant la clause `loop` ¹².
2. Utilisez la clause `when` afin de réunir les deux playbooks précédents en un seul ¹³.

5.1 Utilisation d'un rôle Ansible

Installer le rôle `firewall` sur tous les containers (voir <https://galaxy.ansible.com/geerlingguy/firewall> avec la commande `ansible-galaxy`.

9. https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html

10. Apache relit sa configuration lors de l'envoi du signal HUP

11. voir https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html

12. https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html

13. https://docs.ansible.com/ansible/latest/user_guide/playbooks_conditionals.html#playbooks-conditionals

Paramétrer les containers cibles pour n'accepter en entrée que les ports 22,80,8080. Verifier la mise en place des règles dans un container "Rocky" et un "Debian".

6 Utilisation professionnelle d'Ansible

6.1 Création d'un container via Ansible

1. Créez un container 203-debian-mine avec Ansible et son module docker. Ce container utilisera l'image suivante : registry.iutbeziers.fr/debianiut. Il sera à l'état running après le passage du playbook et sera accessible sur le port 80 de l'hôte.

6.2 Gestion des routeurs/switch Arista avec Python

Ansible permet de piloter de nombreuses marques de routeurs et switches.

6.2.1 Installation d'une plateforme de test Arista vEOS

Nous utiliserons le package Python docker-topo qui génère des topologies avec des routeurs Arista sous forme de containers. Nous allons nous en servir afin de générer une topologie simple (3 routeurs).

Il faut d'abord pouvoir récupérer l'ip des 3 containers au travers d'une fonction shell :

```
docker-ip() {  
    docker inspect --format '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' "$@"  
}  
# récupérez l'idée d'un container (docker ps)  
docker ps  
docker-ip id-du-container
```

Vous avez maintenant 3 routeurs accessibles depuis votre hôte.

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
6c487625295c	registry.iutbeziers.fr/ceosimage:4.25.0F	"/sbin/init systemd..."	40 seconds ago	Up 39 seconds	0.0.0.0:9002->443
8e84ae6c213f	registry.iutbeziers.fr/ceosimage:4.25.0F	"/sbin/init systemd..."	42 seconds ago	Up 41 seconds	0.0.0.0:9001->443
4e2843595f78	registry.iutbeziers.fr/ceosimage:4.25.0F	"/sbin/init systemd..."	43 seconds ago	Up 42 seconds	0.0.0.0:9000->443
521bb14d170e	registry.iutbeziers.fr/pythoniut:heavy	"/bin/bash"	52 minutes ago	Up 52 minutes	4443/tcp, 8080/tcp,

La commande 'bash' vous permet d'accéder à la partie Linux du routeur et la commande 'Cli' de passer en mode de configuration "cisco like".

Vous pouvez vous aider de la documentation suivante :

<https://readthedocs.org/projects/ansible-arista-howto/downloads/pdf/latest/>

6.2.2 Configuration du routeur/switch Arista via ssh

1. Configurez un mot de passe administrateur sur vos 3 routeurs

```
conf t  
username aaa privilege 15 secret mdp  
end  
wr mem
```

Le routeur est aussi accessible via ssh.

```
ssh aaa@192.168.16.5  
Password: mdp  
Last login: Tue Mar 12 20:39:15 2019 from 192.168.16.2  
ceos3>
```


2. Modifiez votre `/etc/ansible/hosts` afin qu'il corresponde aux adresses IP des routeurs récupérées par la commande `docker-ip`.
3. Vérifiez la connectivité sur le groupe `arista` (`ceos1`, `ceos2`, `ceos3`).
4. Changez le `hostname` des trois membres du groupe `Arista` (`ceos1`, `ceos2`, `ceos3`).
5. Utilisez le module `arista.eos.eos_command` afin d'afficher la version EOS des 3 routeurs Arista.
6. A l'aide du module `arista.eos.eos_facts` récupérer seulement les "facts" liées à l'interface en vous aidant de (https://docs.ansible.com/ansible/latest/collections/arista/eos/eos_facts_module.html#ansible-collections-arista-eos-eos-facts-module)
7. Configurez manuellement une adresse ip sur l'interface `Ethernet1` de `ceos1`. Supprimez-la ensuite avec Ansible. Ajoutez une adresse IPV4 à `Ethernet1` et IPV6 à `Ethernet2`.
8. Sauvegardez la configuration des trois switchs via Ansible.
9. Quelles sont les deux points d'entrée d'Ansible dans un switch L3 Arista EOS ?

6.3 Utilisation d'Ansible pour piloter une machine Windows

6.3.1 Installation de WinRM sur une machine Windows

Vous pouvez obtenir une VM windows 11 sur https://developer.microsoft.com/fr-fr/windows/downloads/virtual-machines/?WT.mc_id=-blog-scottha

Ansible utilise WinRM (Windows Remote Shell) pour accéder et piloter une machine Windows. Vous activerez néanmoins le serveur ssh sur la VM windows avec un compte local (utilisateur `do` password `do`) afin de copié-collé les commandes. WinRM n'utilise pas le protocole SSH mais TLS.

Configurez le :

Son installation est réalisée par les commandes PowerShell suivantes :

```
Get-NetConnectionProfile | Set-NetConnectionProfile -NetworkCategory Private
$Script='https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1'
(New-Object System.Net.WebClient).DownloadFile($Script,'ConfigureRemotingForAnsible.ps1')
PowerShell -NoProfile -ExecutionPolicy Bypass -Command ./ConfigureRemotingForAnsible.ps1 -SkipNetworkProfileCheck
↪ -EnableCredSSP
Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False
```

L'installation peut être validée par ces commandes sous PowerShell en tant qu'administrateur :

```
PS C:\WINDOWS\system32> Get-Service WinRM

Status Name          DisplayName
-----
Running WinRM          Gestion à distance de Windows (Gest...
PS C:\WINDOWS\system32> winrm get winrm/config/client
Client
NetworkDelayms = 5000
URLPrefix = wsman
AllowUnencrypted = false
Auth
Basic = true
Digest = true
Kerberos = true
Negotiate = true
Certificate = true
CredSSP = false
DefaultPorts
HTTP = 5985
HTTPS = 5986
```

```
TrustedHosts

PS C:\WINDOWS\system32> winrm e winrm/config/listener
Listener
Address = *
Transport = HTTP
Port = 5985
Hostname
Enabled = true
URLPrefix = wsman
CertificateThumbprint
ListeningOn = 127.0.0.1, 172.28.60.241, 192.168.1.56, 192.168.64.1, 192.168.217
```

Il faut rendre le démarrage automatique :

```
Set-Service -Name WinRM -StartupType Automatic
start-service -name WinRM # pour lancer le service (parfois nécessaire aussi après une mise à jour)
```

1. Combien y a t il de modes d'authentification pour winRM?
2. Décrivez winRM.

6.3.2 Variables pour les hôtes Windows dans le fichier d'inventaire

Les hôtes Ansible doivent être déclarés dans le fichier `/etc/ansible/hosts` du serveur de la façon suivante avec le groupe PC tpansible :

```
[PCtpansible]
pc1 ansible_host="10.244.0.131"
pc2 ansible_host="10.244.0.132"
...
```

Il faut aussi créer sous la directory `/etc/ansible` un folder `group_vars` qui contient le fichier PC du nom du groupe de machine dans `/etc/ansible/host`. Ce fichier contient les éléments de configurations factorisés pour un groupe.

```
ansible_ssh_user: do
ansible_winrm_operation_timeout_sec: 90
ansible_winrm_read_timeout_sec: 300
ansible_winrm_transport: credssp
ansible_ssh_pass: do
ansible_winrm_server_cert_validation: ignore
ansible_ssh_port: 5986
ansible_connection: winrm
```

6.3.3 Gestion des logiciels installés avec Chocolatey

Chocolatey est un gestionnaire de package Windows. Ansible est en capacité de l'utiliser pour installer des packages.

Les commandes de Chocolatey sont accessibles ici :

<https://chocolatey.org/docs/commands-reference>.

Quelques commandes utiles :

```
C:\WINDOWS\system32>choco list -l
Chocolatey v0.10.11
7zip 18.5.0.20180730
7zip.install 18.5.0.20180730
...
31 packages installed.
```

```
choco install -y imagemagick
choco upgrade imagemagick
```

6.3.4 Tâches à réaliser

1. Vérifiez le bon fonctionnement d'Ansible via le package win_ping.
2. Réalisez les opérations suivantes :
 - Installation de chocolatey à l'aide du rôle ansible deekayen.chocolatey sur la VM cible.
 - Création d'une directory C :\Temp.
 - Activation du serveur SSH natif de Windows 11.
 - Installation silencieuse de Wireshark en utilisant les arguments suivants :

```
arguments:
- /S
- /NCRC
- /desktopicon=yes
- /quicklaunchicon=yes
```

- Installation de Libre Office via chocolatey.

7 Tips & tricks

7.1 rolling update

```
-name: "rolling update"
hosts: apache
serial:["10%", "100%"]
tags: apache2
gather_facts: no
roles:
- role: "mediawiki/configuration"
```