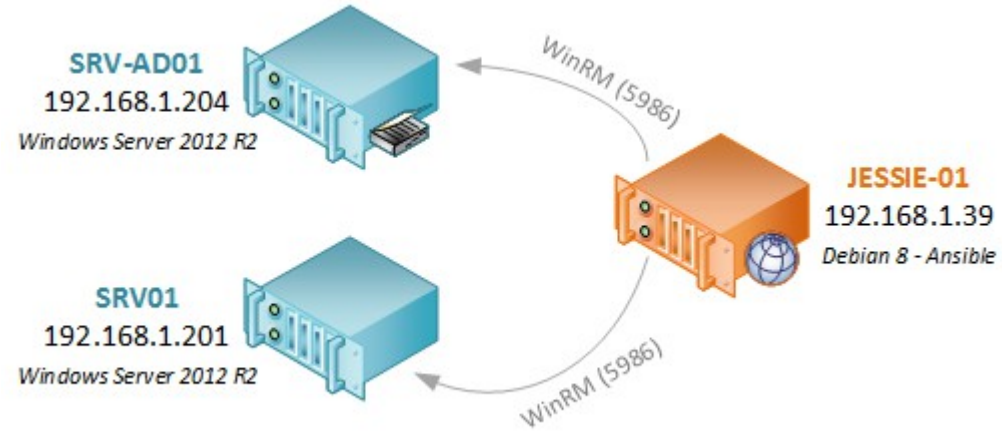
Ansible Introduction:

Automatiser l'exécution des taches sur plusieurs machines distantes en même temps, à partir d'un serveur.

Installation : Il faut savoir qu'Ansible ne peut pas s'installer sur Windows, même en passant par Cygwin. Autrement dit, le « serveur » Ansible doit forcément être sous Linux.

D'ailleurs, nous n'utiliserons pas SSH comme cela est le cas lorsqu'on administre des serveurs Linux avec Ansible, simplement car Windows ne connaît pas SSH. Il a son propre module d'administration à distance en ligne de commande : WinRM (Windows Remote Management).

Exemple d'infrastructures :



Le fichier de configuration d'Ansible est : **/etc/ansible/ansible.cfg**.

Actions Ad-Hoc :

Exécuter des commandes rapides à partir du binaire ansible. Par exemple, exécuter une action simple qui consiste à redémarrer 10 machines de votre infrastructure (administré par Ansible).

Actions PlayBooks :

Exécuter des actions à partir du binaire **ansible-playbook**. Un Playbook est écrit en YAML et permet de réaliser des actions plus complexes.

Avec les « **actions Ad-Hoc** » vous pourrez faire des choses basiques alors qu'avec les « **actions PlayBooks** » vous pourrez effectuer de la véritable automatisation avancée. Les deux sont intéressants mais les PlayBooks sont la véritable force d'Ansible.

Installer Ansible sous Debian 8 :

Ouvrez un terminal sur votre Linux, Debian, et commencez par installez python-pip qui va

être utile pour installer Ansible et pywinrm.

pywinrm : Librairie Python pour utiliser WinRM.

Saisissez les trois commandes ci-dessous pour installer **python-pip**, puis à l'aide de **pip installer ansible et pywinrm**.

```
apt-get install python-pip

pip install ansible

pip install http://github.com/diyan/pywinrm/archive/master.zip#egg=pywinrm
```

Fichier hosts : Ajout des clients Windows

Le fichier hosts est utilisé pour définir l'ensemble des groupes d'hosts que vous souhaitez administrer. Dans le répertoire « **/etc/ansible** », créez ce fichier et éditez-le avec un éditeur comme vim ou nano ou ....

vim /etc/ansible/hosts

Note : Si vous désirez changer le nom du fichier hosts, cela doit se modifier dans le fichier de configuration ansible.cfg, avec la directive hostfile.

Dans le fichier /etc/ansible/hosts, on écrit ce groupe de machines qu'on va gérer (c'est juste un exemple. C'est à vous de mettre vos propres machines.):

```
[windows]

# Domain Controller – 192.168.1.204

SRV-AD01.it-connect.fr

# Server – 192.168.1.201

SRV01.it-connect.fr
```

Les lignes qui commencent par # sont des lignes commentées. Il faut savoir également que les lignes vides sont ignorées.

Enregistrez et fermez le fichier, nos hôtes clients sont désormais référencés dans Ansible.

Pour cacher les mots de passes, il faut utiliser ansible-vault.

Voir **https://www.it-connect.fr/debutez-avec-ansible-et-gerez-vos-serveurs-windows**

Pour simplifier, on va mettre les mots de passes en clair.

Pour cela, on doit avoir deux types de fichiers :

1. inventory.ini : qui va contenir la liste des machines classées par catégories (ça remplace le hosts qu'on a vu précédemment)
2. play_book.yml : qui va contenir la liste des taches à exécuter

La commande classique est :

ansible-playbook -i inventory.ini play_book.yml

Exemples de fichier d'inventory :

1. Si on veut exécuter sur le localhost (sans passer par ssh) :

localhost ansible_connection=local

2. un fichier inventory.ini de machines distantes classique :

```
[all]
192.168.101.67          ansible_user=mourad ansible_password='Pass.2023'
#ubuntu20targ1          ansible_user=mourad ansible_password='Pass.2023'
```

Exemple d'action ad-hoc :

On fait un ping à toutes les machines de notre fichier inventory

ansible all -m ping -i inventoty.txt

Exemples de fichier de play-book :

il commence toujours par la chaîne des 3 tirets sans guillemets : « --- ».

httpd.yml :

```
---
  - name: httpd
    hosts: all
    tasks:
      - name: ensure apache is at the latest version
        yum:
          name: httpd
          state: latest
      - name: ensure apache is running
        service:
          name: httpd
          state: started
```

##################

ls.yml :

```
---
- hosts: all
  tasks:
```

```
    - name: Run ls command
      command: ls
      register: ls_output

    - debug:
        var: ls_output.stdout_lines

##################################
```

mkdir.yml :

```
---
- hosts: all
  tasks:
    - name: Create a directory if it doesn't exist
      file:
        path: ~/tmp
        state: directory

#########################
```

print_hostname.yml :

```
---
- name: print hostname infos
  hosts: all
  tasks:
    - name: display
      debug:
        msg: "The new hostname is {{ ansible_hostname }} and the OS is {{ ansible_version }}"

#######################
```

script.yml :

```
---
- hosts: all
  tasks:
    - name: Run script.sh using script module
      script: ~/ansible/script.sh
      register: ls_output

    - debug:
        var: ls_output.stdout_lines
####################################
```

Exemple simple de script.sh :

```
d=$(date +"%Y_%m_%d_%H_%M_%S")

rmdir tmp*
mkdir tmp_$d

echo "
        $d
"
#########################
```