GESTION DES CONFIGURATIONS

Sommaire

Présentation Globale

Définition de gestion des configurations :

Les applications de gestion des configurations permettent d'assurer que l'état d'un système correspond à l'état décrit par un ensemble de scripts de provisionnement. C'est aussi un processus qui permet de maintenir les systèmes informatiques, les serveurs et les logiciels dans l'état souhaité et d'en préserver la cohérence.

Les applications de système des gestions de configurations on comme taches :

- Classer et gérer les systèmes par groupes et sous-groupes
- Modifier de manière centralisée les configurations de base
- Déployer de nouveaux paramétrés sur tous les systèmes applicables
- Automatiser l'identification, l'application de correctifs et la mise à jour des systèmes
- Identifier les conjurations obsolètes, non conformes et peu efficaces
- Hiérarchiser les actions
- Accéder à des mesures de correction prescriptives et appliquer

En résumé :

La gestion des configurations permet d'envoyer une image avec des configurations personnalisé d'un pc à partir d'un environnement standard et les envoient des bonnes configurations dans les bons services (par exemple : la compta, marketing, IT, production…). Il est aussi possible de rajouter des applications, script d'automatisation ou autre sur une machine en service.

Annexe :

Scripts de provisionnement :

Un script de provisionnement est un script qu'on rajoute à l'infrastructure de base.

Environnement standard :

Un environnement standard est un environnement d'exploitation standard contenant l'OS et les configurations de base. C'est la configuration qui est avant l'ajout de la personnalisation de configurations.

Sources :

Définition de gestion des configurations :

https://www.redhat.com/fr/topics/management/what-is-an-soe

https://www.digitalocean.com/community/conceptual-articles/an-introduction-to-configuration-management-with-ansible-fr

Définition d’un environnement d’exploitation standard :

https://www.redhat.com/fr/topics/management/what-is-an-soe

Présentation des applications :

Ansible :

Ansible est un outil de gestion de configuration open source qui permet aux administrateurs système de configurer et de gérer des infrastructures informatiques de manière automatisée. Voici comment il fonctionne :

Architecture client-serveur :

Ansible utilise une architecture client-serveur dans laquelle un hôte de contrôle est utilisé pour communiquer avec des hôtes distants (nodes). L'hôte de contrôle doit avoir Ansible installé, tandis que les nodes n'ont pas besoin d'Ansible mais doivent avoir un système d'exploitation pris en charge et une connexion SSH.

Inventaire :

L'inventaire est un fichier texte qui contient une liste des nodes à gérer. Il peut être stocké localement sur l'hôte de contrôle ou sur un serveur distant. L'inventaire peut être organisé en groupes, ce qui permet de gérer facilement plusieurs nodes en même temps. Par exemple

[Nodes]

Node1 ansible_host=192.168.1.10 ansible_user=sio

Node2 ansible_host=192.168.1.11 ansible_user=sio

[srv_web]

Node1

Node2

Playbooks :

Les playbooks sont des fichiers YAML qui définissent les tâches à exécuter sur les nodes. Les tâches peuvent inclure l'installation de packages, la modification de fichiers de configuration, la création d'utilisateurs, etc. Les playbooks peuvent également utiliser des variables pour permettre une configuration plus flexible.

Modules :

Les modules sont des scripts écrits en Python qui sont utilisés pour exécuter des tâches spécifiques sur les nodes. Ils sont utilisés par les playbooks pour effectuer des actions sur les nodes. Ansible dispose d'un grand nombre de modules intégrés, mais il est également possible de créer des modules personnalisés pour des besoins spécifiques.

Connexion SSH :

Ansible utilise SSH pour se connecter aux nodes distants et exécuter les tâches définies dans les playbooks. Les nodes doivent être accessibles via SSH et l'hôte de contrôle doit être en mesure de se connecter aux nodes sans avoir à entrer un mot de passe.

Exécution des playbooks :

Les playbooks sont exécutés en utilisant la commande "ansible-playbook" sur l'hôte de contrôle. Lorsque la commande est exécutée, Ansible se connecte aux nodes distants et exécute les tâches définies dans les playbooks. Les résultats de l'exécution sont renvoyés à l'hôte de contrôle pour permettre une gestion centralisée.

Gestion des erreurs :

En cas d'erreurs, Ansible fournit des informations détaillées sur les tâches qui ont échoué et les raisons de l'échec. Cela permet aux administrateurs système de diagnostiquer rapidement les problèmes et de les résoudre.

En résumé, Ansible permet une gestion de configuration automatisée et reproductible en utilisant des playbooks pour définir les tâches à exécuter sur les nodes distants. Les modules sont utilisés pour effectuer des actions spécifiques sur les nodes, tandis que l'inventaire permet de gérer facilement les nodes en groupes. Ansible utilise SSH pour se connecter aux nodes distants et renvoie les résultats de l'exécution des tâches à l'hôte de contrôle pour une gestion centralisée. En cas d'erreurs, Ansible fournit des informations détaillées pour faciliter la résolution des problèmes.

Source :

Présentation :

https://www.ansible.com

En Plus :

https://www.youtube.com/watch?v=Cisg9bLhLkk

Puppet :

Puppet est un outil de gestion de configuration open source qui permet aux administrateurs système de déployer et de gérer des applications et des infrastructures de manière automatisée et reproductible.

Il utilise une approche déclarative pour spécifier l'état souhaité d'un système et s'assure que cet état est maintenu en permanence. Les administrateurs système peuvent utiliser Puppet pour gérer des configurations de serveurs, des mises à jour de logiciels, des installations d'applications, des réglages de sécurité, et bien plus encore.

Puppet est conçu pour être extensible et modulaire, ce qui permet aux administrateurs système d'adapter l'outil en fonction de leurs besoins spécifiques. Il utilise un langage de description de ressources appelé Puppet DSL (Domain Specific Language) qui permet de décrire de manière déclarative les ressources à gérer.

Puppet peut être utilisé pour gérer des systèmes d'exploitation tels que Linux, Windows, macOS, ainsi que des environnements cloud tels que Amazon Web Services, Microsoft Azure, Google Cloud Platform, etc.

En utilisant Puppet, les administrateurs système peuvent automatiser les tâches répétitives, réduire les erreurs humaines, améliorer la sécurité, assurer la conformité et améliorer l'efficacité de leur infrastructure informatique.

Source :

https://chat.openai.com/chat

Tuto

Node1 connecter en root :

Pour commencer on va créer l'utilisteur node1 avec la commande *adduser node1*

```
root@graph-deb11-node1:~# adduser node1
Ajout de l'utilisateur « node1 » ...
Ajout du nouveau groupe « node1 » (1001) ...
Ajout du nouvel utilisateur « node1 » (1001) avec le groupe « node1 » ...
Le répertoire personnel « /home/node1 » existe déjà.  Rien n'est copié depuis «
/etc/skel ».
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd: password updated successfully
Changing the user information for node1
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Cette information est-elle correcte ? [O/n]
root@graph-deb11-node1:~#
```

Ensuite on va télécharger python qui sera utiliser par ansible : *apt install python3*

Pc Node2 connecter en root :

Pour commencer on va créer l'utilisteur node2 avec la commande *adduser node2*

```
root@sv-deb11-node2:~# adduser node2
Ajout de l'utilisateur « node2 » ...
Ajout du nouveau groupe « node2 » (1001) ...
Ajout du nouvel utilisateur « node2 » (1001) avec le groupe « node2 » ...
Création du répertoire personnel « /home/node2 »...
Copie des fichiers depuis « /etc/skel »...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd: password updated successfully
Changing the user information for node2
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Cette information est-elle correcte ? [O/n]
root@sv-deb11-node2:~# _
```

Ensuite on va télécharger python qui sera utiliser par ansible : *apt install python3*

Pc global connecter en root :

Installer ssh avec la commande *Apt install ssh*

Vérifier le statut *systemctl status sshd*

```
root@graph-deb11:~# systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: e>
     Active: active (running) since Wed 2023-03-29 16:34:46 CEST; 54min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 482 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 503 (sshd)
      Tasks: 1 (limit: 4659)
     Memory: 4.0M
        CPU: 29ms
     CGroup: /system.slice/ssh.service
             └─503 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

mars 29 16:34:46 graph-deb11 systemd[1]: Starting OpenBSD Secure Shell server...
mars 29 16:34:46 graph-deb11 sshd[503]: Server listening on 0.0.0.0 port 22.
mars 29 16:34:46 graph-deb11 sshd[503]: Server listening on :: port 22.
mars 29 16:34:46 graph-deb11 systemd[1]: Started OpenBSD Secure Shell server.
```

Redémarrer le service si ça ne marche pas avec *systemctl restart sshd*

Faire la commande *ssh-keygen -t rsa*

```
root@graph-deb11:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

La première question permet de préciser l'emplacement du dossier de sauvegarde de clé. Si voulez pas préciser un dossier appuyer sur la touche Entrer.

```
root@graph-deb11:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
```

Ensuite si vous voulez un mot de passe pour la connexion ssh. Si vous ne voulez pas avoir de mot de passe faite 2 fois Entrer

Et donc voici la clé avec son chemin

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:GHxOaBfOUhvlxE3zIOR6cEXrqmb31ThTKICYmJ1Qv/k root@graph-deb11
The key's randomart image is:
+---[RSA 3072]----+
|    ... ++=+*    |
|     * X O.o.=   |
|    o @ @ = . .  |
|     . O * o   . |
|      . S . o . .|
|         o . . + |
|          E   = .|
|        o..  . o |
|       o.. ..    |
+----[SHA256]-----+
```

Faire la commande : *ssh-copy-id sio@192.168.0.50* puis faite yes et entrer le mot de passe de l'utilisateur sio.

```
root@graph-deb11:~# ssh-copy-id sio@192.168.0.50
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.50 (192.168.0.50)' can't be established.
ECDSA key fingerprint is SHA256:eqGN0jqgyU5D6XmZr71+CylYNCuF71ugMq4wx1eBKis.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
sio@192.168.0.50's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'sio@192.168.0.50'"
and check to make sure that only the key(s) you wanted were added.

root@graph-deb11:~#
```

Faire la commande : *ssh-copy-id node1@192.168.0.10* puis faite yes entrer le mot de passe de l'utilisateur node1.

```
root@graph-deb11:~# ssh-copy-id node1@192.168.0.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
ECDSA key fingerprint is SHA256:KnNsUCydfQW+FzE9lj69AxlLlEqOag1k4puN5NOnr6c.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
node1@192.168.0.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'node1@192.168.0.10'"
and check to make sure that only the key(s) you wanted were added.

root@graph-deb11:~#
```

Faire la commande : *ssh-copy-id node2@192.168.0.11* puis faite yes entrer le mot de passe de l'utilisateur node2.

```
root@graph-deb11:~# ssh-copy-id node2@192.168.0.11
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.11 (192.168.0.11)' can't be established.
ECDSA key fingerprint is SHA256:eqGN0jqgyU5D6XmZr71+CylYNCuF71ugMq4wx1eBKis.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
node2@192.168.0.11's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'node2@192.168.0.11'"
and check to make sure that only the key(s) you wanted were added.
```

Node1 en root en connexion ssh avec la commande *: ssh node1@192.168.0.10* sur le pc global :

```
root@graph-deb11:~# ssh node1@192.168.0.10
Linux graph-deb11-node1 5.10.0-21-amd64 #1 SMP Debian 5.10.162-1 (2023-01-21) x8
6_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 29 17:40:21 2023 from 192.168.0.1
node1@graph-deb11-node1:~$ su -l root
Mot de passe :
root@graph-deb11-node1:~#
```

Pour que ansible puisse se connecter à distance sur cet Node il faut un utilisateur avec les privilèges sudo. Et pour ca on va l'installer

Faire : *apt insatall sudo*

On va créer un groupe sudo si celui-ci n'est pas créer (normalement il est déjà créé) : *groupadd sudo*

On va rajouter l'utilisateur node1 dans le groupe sudo : *usermod -aG sudo node1*

On va vérifier s'il est bien dans le groupe sudo avec la commande : *groups node1*

```
root@graph-deb11-node1:~# usermod -aG sudo node1
root@graph-deb11-node1:~# groups node1
node1 : node1 sudo
root@graph-deb11-node1:~#
```

Ensuite on va donner l'autorisation à tout machine (dont ansible) de pouvoir se connecter sans mot de passe en ajoutant un fichier de configuration. *touch visudo -f /etc/sudoers.d/node1*

Puis on le modifier avec nano : *nano visudo -f /etc/sudoers.d/node1*

Puis écrire dans le fichier : node1 ALL=(ALL) NOPASSWD: ALL

```
  GNU nano 5.4                          visudo
node1 ALL=(ALL) NOPASSWD:ALL
```

Puis relancer la machine avec la commande *reboot*

Node2 en root connexion ssh avec la commande : *ssh node2@192.168.0.11* sur le pc global :

```
sio@graph-deb11:~$ ssh node2@192.168.0.11
The authenticity of host '192.168.0.11 (192.168.0.11)' can't be established.
ECDSA key fingerprint is SHA256:eqGN0jqgyU5D6XmZr71+CylYNCuF71ugMq4wx1eBKis.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.11' (ECDSA) to the list of known hosts.
node2@192.168.0.11's password:
Linux sv-deb11-node2 5.10.0-21-amd64 #1 SMP Debian 5.10.162-1 (2023-01-21) x86_6
4

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
node2@sv-deb11-node2:~$ su -l root
Mot de passe :
root@sv-deb11-node2:~#
```

Pour que ansible puisse se connecter à distance sur cet Node il faut un utilisateur avec les privilèges sudo. Et pour ça on va l'installer

Faire : *apt insatall sudo*

On va créer un groupe sudo si celui-ci n'est pas créer (normalement il est déjà créé) : *groupadd sudo*

On va rajouter l'utilisateur node1 dans le groupe sudo : *usermod -aG sudo node2*

On va vérifier s'il est bien dans le groupe sudo avec la commande : *groups node2*

```
root@sv-deb11-node2:~# usermod -aG sudo node2
root@sv-deb11-node2:~# groups node2
node2 : node2 sudo
root@sv-deb11-node2:~#
```

Ensuite on va donner l'autorisation à tout machine (dont ansible) de pouvoir se connecter sans mot de passe en ajoutant un fichier de configuration. *touch visudo -f /etc/sudoers.d/node2*

Puis on le modifier avec nano : *nano visudo -f /etc/sudoers.d/node2*

Puis écrire dans le fichier : node2 ALL=(ALL) NOPASSWD: ALL

```
  GNU nano 5.4                          visudo
node2 ALL=(ALL) NOPASSWD:ALL
```

Puis relancer la machine avec la commande *reboot*

Node de contrôle en root connexion ssh avec la commande : *ssh sio@192.168.0.50* sur le pc global :

On va installer ansible avec la commande : *Apt installe ansible*

Ensuite le serveur ssh avec la commande *apt install openssh-server*

Taper la commande *systemctl status ssh*

```
root@graph-deb11-globaleansible:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: e
     Active: active (running) since Mon 2023-03-27 11:22:05 CEST; 48min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 2071 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 2072 (sshd)
      Tasks: 1 (limit: 2321)
     Memory: 1.0M
        CPU: 32ms
     CGroup: /system.slice/ssh.service
             └─2072 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

mars 27 11:22:05 graph-deb11-master systemd[1]: Starting OpenBSD Secure Shell s
mars 27 11:22:05 graph-deb11-master sshd[2072]: Server listening on 0.0.0.0 por
mars 27 11:22:05 graph-deb11-master sshd[2072]: Server listening on :: port 22.
mars 27 11:22:05 graph-deb11-master systemd[1]: Started OpenBSD Secure Shell se
```

Si le service n'est pas en cours d'exécution alors taper la commande *systemctl start ssh*

Ensuite on va créer un lien ssh entre les nodes et le node de contrôle en faisant les mêmes étapes lors de l'installation sur le pc global

Faite la commande : *ssh-keygen -t rsa*

```
root@sv-deb11-ansible:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:RYvt3iWkH4X7WtyUFPsoxKFxO6XRPc0deQSx3xVMWdQ root@sv-deb11-ansible
The key's randomart image is:
+---[RSA 3072]----+
|          o +.*%%|
|         + * B.*E|
|        . = O oo+|
|         o + +.o=|
|        S o = o.=|
|         . o B o |
|          . o + .|
|             o   |
|              .  |
+----[SHA256]-----+
root@sv-deb11-ansible:~#
```

Faite la commande : *ssh-copy-id node1@192.168.0.10* et fait yes

```
root@sv-deb11-ansible:~# ssh-copy-id node1@192.168.0.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
ECDSA key fingerprint is SHA256:KnNsUCydfQW+FzE9lj69AxlLlEqOag1k4puN5NOnr6c.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
node1@192.168.0.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'node1@192.168.0.10'"
and check to make sure that only the key(s) you wanted were added.

root@sv-deb11-ansible:~#
```

Faite la commande : *ssh-copy-id sio@192.168.0.1* et fait yes

```
root@sv-deb11-ansible:~# ssh-copy-id sio@192.168.0.1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.1 (192.168.0.1)' can't be established.
ECDSA key fingerprint is SHA256:KnNsUCydfQW+FzE9lj69AxlLlEqOag1k4puN5NOnr6c.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
sio@192.168.0.1's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'sio@192.168.0.1'"
and check to make sure that only the key(s) you wanted were added.

root@sv-deb11-ansible:~#
```

ssh-copy-id -l node2@192.168.0.11 et fait yes

```
root@sv-deb11-ansible:~# ssh-copy-id node2@192.168.0.11
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa
.pub"
The authenticity of host '192.168.0.11 (192.168.0.11)' can't be established.
ECDSA key fingerprint is SHA256:eqGN0jqgyU5D6XmZr71+CylYNCuF71ugMq4wx1eBKis.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
node2@192.168.0.11's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'node2@192.168.0.11'"
and check to make sure that only the key(s) you wanted were added.

root@sv-deb11-ansible:~#
```

Ensuite créer le chemin /etc/ansible avec la commande *mkdir /etc/ansible* et le fichier hosts avec *touch /etc/ansible/hosts*

```
root@sv-deb11-ansible:~# mkdir /etc/ansible
root@sv-deb11-ansible:~# touch /etc/ansible/hosts
```

Puis modifiez le, avec la commande nano ou vi : *nano /etc/ansible/hosts*

Exemple de configuration de node

```
 GNU nano 5.4                    /etc/ansible/hosts *
[nodes]
node1 ansible_host=192.168.0.10 ansible_user=node1
node2 ansible_host=192.168.0.11 ansible_user=node2

[global]
192.168.0.1 ansible_user=sio

[srv_web]
node1

[srv_ntfs]
node2
```

- Le premier groupe est « nodes » qui contient l'adresse ip, l'utilisateur et le mot de passe pour chaque node.
- Le deuxième groupe est « globale » contenant que le pc global avec l'adresse ip et utilisateur
- Le troisième groupe est srv_web contenant node1.
- Le quatrième est srv_ntfs contenant node 2.

Ces groupes vont permettre de facilité le déploiement des configurations en envoyer les fichiers seulement au groupe voulu.

Ensuite pour vérifier on fait un *ansible all -m ping*

```
root@sv-deb11-ansible:~# ansible all -m ping
192.168.0.1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
root@sv-deb11-ansible:~#
```

Méthode 1

Ensuite on va mettre à jours l'interpréteur python sur le groupe nodes en faisant la commande : *ansible nodes -m apt -a "name=python3 state=present" -b*

Voici le résultat si python3 était déjà installé

```
root@sv-deb11-ansible:~# ansible nodes -m apt -a "name=python3 state=present" -b
node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1679654820,
    "cache_updated": false,
    "changed": false
}
node1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1679909323,
    "cache_updated": false,
    "changed": false
}
```

Si vous voulez le supprimez faite la commande ansible nodes -m apt -a "name=python3 state=absent" -b

Méthode 2

Il existe une autre manière de télécharger une application avec les playbook. Pour ça il faut créer un fichier yml. Ce fichier peut être stocker n'importe où. Pour le créer il suffit d'utiliser touch exemple *touch /etc/ansible/playbook1.yml*

Voici un exemple qui permet d'installer apache2 sur le groupe nodes

```
  GNU nano 5.4                  /etc/ansible/playbook.yml *
---
- hosts: nodes
  become: true
  tasks:
    - name: Install apache2
      apt:
        name: apache2
        state: present
```

Pour activer le fichier il suffit de faire : *ansible-playbook -i /etc/ansible/hosts /etc/ansible/playbook.yml*

A l'encontre de la premier méthode le fichier playbook permet d'utiliser different module des modules créer par ansible ou des modules personnalisés. Pour ce cas ansible utilise le module apt avec les paramètres name et statut.

```
root@sv-deb11-ansible:~# ansible-playbook -i /etc/ansible/hosts /etc/ansible/pla
ybook.yml

PLAY [nodes] *******************************************************************

TASK [Gathering Facts] *********************************************************
ok: [node2]
ok: [node1]

TASK [Install apache2] *********************************************************
ok: [node2]
ok: [node1]

PLAY RECAP *********************************************************************
node1                      : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
node2                      : ok=2    changed=0    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

root@sv-deb11-ansible:~#
```

Voici un autre exemple de l'utilisation de la méthode2

Fichier yml qui a pour objectif de créer un répertoire test1 sur les nodes

```
 GNU nano 5.4                  /etc/ansible/cr_rep.yml
---
- name: création d'un répertoir test et un fichier l1
  hosts: nodes
  become: yes

  tasks:
    - name: créer un répertoir test
      file:
       path: /home/ansible/test0
       state: directory
       mode: '0777'
    - name: créer un fichier l1
      file:
       path: /home/ansible/test0/l1.txt
       state: touch
       mode: '0655'
       owner: root
       group: root
```

```
root@sv-deb11-ansible:~# ansible-playbook -i /etc/ansible/hosts /etc/ansible/cr_
rep.yml

PLAY [création d'un répertoir test et un fichier l1] ***************************

TASK [Gathering Facts] *********************************************************
ok: [node2]
ok: [node1]

TASK [créer un répertoir test] *************************************************
changed: [node2]
changed: [node1]

TASK [créer un fichier l1] *****************************************************
changed: [node2]
changed: [node1]

PLAY RECAP *********************************************************************
node1                      : ok=3    changed=2    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
node2                      : ok=3    changed=2    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```