Gestion de configuration

Ansible

1) Avant de débuter

Ansible est l'un des outils d'automatisations et de gestion des configurations les plus populaires (les autres produits les plus connus étant Chef, CFEngine, Foreman , Katello, Puppet)

Certaines caractéristiques d'Ansible expliquent en partie son succès :

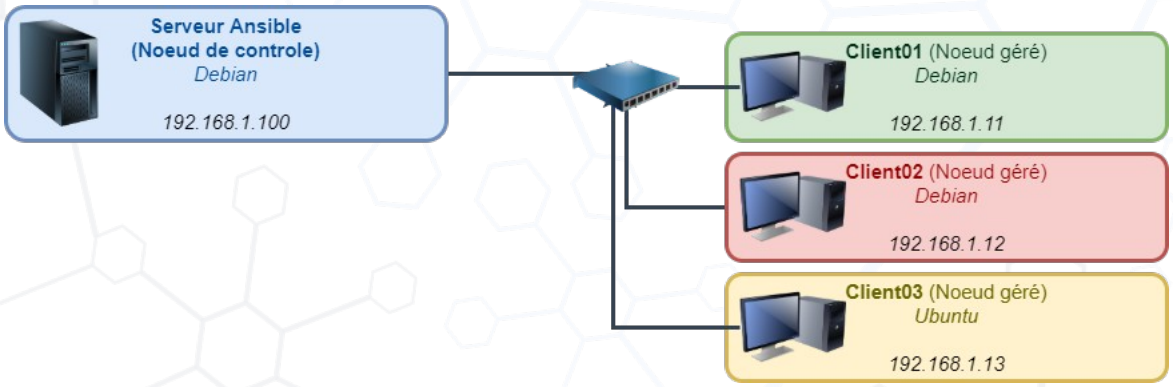
- Il est open source est gratuit
- il est écrit en python, un langage approbation largement reconnu
- il ne nécessite pas d'installation d'agents sur les postes clients
- il fonctionne selon la méthode push et ne nécessite pas de serveur dédié (=peut être utilisé comme fonctionnalité supplémentaire d'un serveur existant)
- il est facile à installer et à utiliser
- il est très flexible et permet d'orchestrer des environnements hétérogènes

- il s'intègre très bien dans des de flux de travail informatique complexe

Afin d'orchestrer les tâches d'automatisations, Ansible doit être installé sur un serveur qui jouera le rôle de nœud de contrôle.

Ce nœud contiendra des fichiers de configuration appelée Playbook, ces fichiers au format YAML contiendront les différentes étapes à exécuter sur une ou plusieurs machines appelées nœuds gérés.

Dans la suite de ce laboratoire, nous utiliserons un serveur Ansible qui servira à effectuer des tâches sur différents clients.



Serveur	OS	Adresse
Serveur Ansible	Debian 11.2	192.168.1.100
Client01	Debian 11.2	192.168.1.11
Client02	Debian 11.2	192.168.1.12
Client03	Ubuntu 21.10	192.168.1.13

Pour chaque nœud géré, il sera nécessaire de disposer d'une connexion SSH (normalement installée par défaut sur la majorité des distributions Linux, par contre l'installation sera à réaliser sous Windows)

Un compte de service nommé "ansible" sera créé sur chaque machine afin de sécuriser a minima les opérations. Ce compte disposera des autorisations sudo et servira notamment pour établir les connexions SSH entre les machines.

2) Installation de Ansible

Il est possible d'installer Ansible sur le contrôleur de nœud de trois manières différentes :

- À partir du dépôt d'Ubuntu*(voir section 2.1-Installer à partir du dépôt par défaut de Debian en page 3)*
- À partir du dépôt APT d'Ubuntu *(voir section 2.2-Installation à partir du dépôt Ubuntu en page 3)*
- En utilisant l'utilitaire pip Python Package Manager *(voir section 2.3-Installation en utilisant PIP en page 4)*

Procéder à l'installation avec UNE SEULE de ces méthodes

2.1) Installer à partir du dépôt par défaut de Debian

Cette méthode est la plus simple, mais la version disponible n'est pas toujours la dernière

Mettre à jour la liste des dépôts.

```
sudo apt-get update
```

Installer Ansible

```
sudo apt-get install ansible -y
```

2.2) Installation à partir du dépôt Ubuntu

Commencer par installer les prérequis pour ajouter le dépôt PPA d'Ubuntu à Debian.

```
sudo apt-get install gnupg2 curl wget -y
```

Ajouter une ligne au fichier de configuration des dépôts de Debian11

```
sudo nano /etc/apt/sources.list
deb http://ppa.launchpad.net/ansible/ansible/ubuntu bionic main
```

Ajouter la clé GPG au trousseau de clé de Debian

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
Executing: /tmp/apt-key-gpghome.eme7hwwzkz/gpg.1.sh --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
gpg: clef 93C4A3FD7BB9C367 : clef publique « Launchpad PPA for Ansible, Inc. » importée
gpg:Quantité totale traitée : 1
gpg:                importées : 1
```

Mettre à jour la liste des dépôts

```
sudo apt-get update
```

Installer Ansible

```
sudo apt install ansible
```

2.3) Installation en utilisant PIP

Installer les pré-requis Python :

```
sudo apt-get install python3 python3-pip -y
```

Installer Ansible

```
sudo pip3 install ansible
```

2.4) Vérification de l'installation d'Ansible

Pour connaître le chemin du binaire Ansible :

```
which ansible
/usr/bin/ansible
```

Pour connaître la version d'Ansible (exemple de sortie de la commande, dépend de la méthode d'installation)

```
ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.9.2 (default, Feb 28 2021, 17:03:44) [GCC 10.2.1 20210110]
```

3) Configuration d'Ansible

3.1) Configuration du compte de service ansible

Sur chaque machine (**Serveur**, **Client01**, **Client02**, **Client03**) un utilisateur ansible sera créé. Ce compte sera utilisé pour communiquer entre toutes les machines impliquées dans l'automatisation des tâches.

En tant que root :

```
adduser ansible
Ajout de l'utilisateur « ansible » ...
Ajout du nouveau groupe « ansible » (1001) ...
Ajout du nouvel utilisateur « ansible » (1001) avec le groupe « ansible » ...
Création du répertoire personnel « /home/ansible »...
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 ansible
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]
```

Le mot de passe doit être de complexité et d'entropie élevées

```
sudo nano /etc/sudoers
```

```
ansible ALL=(ALL:ALL) NOPASSWD:ALL
```

Sur le **serveur**, agir en tant qu'utilisateur ansible :

```
su ansible
```

À partir du **serveur**, vérifier la présence du service SSH sur le client **Client01**

```
ssh 192.168.1.11
The authenticity of host '192.168.1.11 (192.168.1.11)' can't be established.
ECDSA key fingerprint is SHA256:Pwrjq9P5GEWISg9TcR9hCSgLSrXKnCVPUjZVAdtVMII.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.11' (ECDSA) to the list of known hosts.
ansible@192.168.1.11's password:
…
$
exit
déconnexion
Connection to 192.168.1.11 closed.
```

À reproduire sur les machines **Client02**, **Client03**.

À partir du **serveur**, générer une paire de clés, ne pas entrer de phrase de passe et accepter toutes les valeurs par défaut (appuyer sur entrée à chaque fois) :

```
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ansible/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa
Your public key has been saved in id_rsa.pub
The key fingerprint is:
SHA256:CLMjAAxZJANlta2d4UehW9vqhfOuoG+2eIESyWpzzzA ansible@debian
The key's randomart image is:
+---[RSA 3072]----+
|X==..   .        |
|o=   o . .       |
|.. .+ + o        |
| .+  B B o       |
| ...+.* S .      |
|.o.E.... o       |
|. o.= ..+ .      |
|     =+o +       |
|    o=o.ooo      |
+----[SHA256]-----+
```

Deux fichiers ont été créés dans le répertoire home de l'utilisateur ansible, correspondant aux clés privée et publique. Il faut désormais copier la clé publique sur les nœuds gérés **Client01** , **Client02**, **Client03**.

```
ssh-copy-id -i /home/ansible/.ssh/id_rsa.pub 192.168.1.11
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "id_rsa.pub .pub"
/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
prompted now it is to install the new keys
ansible@192.168.1.11's password:
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh '192.168.1.11'"
and check to make sure that only the key(s) you wanted were added.
```

À partir de **Client01**, modifier le serveur SSH afin de n'autoriser les connexions SSH qu'en utilisant des clés.

```
nano /etc/ssh/sshd_config
```

Dé-commenter et modifier les lignes suivantes :

```
PubkeyAuthentication yes
PasswordAuthentication no
PermitRootLogin prohibit-password
PermitEmptyPasswords no
```

Puis redémarrer le service ssh

```
sudo systemctl restart sshd
```

À partir du **serveur**, tenter une connexion SSH sans mot de passe sur le client **Client01**

```
ssh 192.168.1.11
…
$
exit
déconnexion
Connection to 192.168.1.11 closed.
```

À reproduire sur les machines **Client02**, **Client03**.

3.2) Création du fichier de configuration des hôtes (Hosts Inventory)

Il faut maintenant déclarer les nœuds gérés par le contrôleur de noeuds.

Toute la configuration se fait dans le fichier /etc/ansible/hosts

```
sudo mkdir /etc/ansible/
sudo nano /etc/ansible/hosts
[client01]
192.168.1.11 ansible_ssh_user=ansible

[client02]
192.168.1.12 ansible_ssh_user=ansible

[client03]
192.168.1.13 ansible_ssh_user=ansible
```

Nom du groupe
C'est le nom qui sera utilisé comme cible d'exécution

Adresse IP de la machine cible
Plusieurs adresses peuvent être renseignées dans le groupe

Paramètre permettant de définir le nom à utiliser lors de la connexion au client en SSH

Il est possible d'utiliser des noms DNS à la place des adresses IP.

A tout moment, vous pouvez obtenir la liste des nœuds gérés en utilisant la commande :

```
ansible-inventory --list -y
```

```
all:
  children:
    client01:
      hosts:
        192.168.1.11:
          ansible_ssh_user: ansible
    client02:
      hosts:
        192.168.1.12:
          ansible_ssh_user: ansible
    client03:
      hosts:
        192.168.1.13:
          ansible_ssh_user: ansible
    clients_debian:
      hosts:
        192.168.1.11: {}
        192.168.1.12: {}
    clients_ubuntu:
      hosts:
        192.168.1.13: {}
    ungrouped: {}
```

3.3) Premières automatisations

a)Exemple de commande

À partir du **serveur**, tester la connectivité du **client01** :

```
ansible -m ping client01
192.168.1.11 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Tester la connectivité du **client03** (câble débranché) :

```
ansible -m ping client03
192.168.1.13 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host
192.168.1.13 port 22: No route to host",
    "unreachable": true
}
```

Tester la connectivité du groupe [clients_debian] **client01** **client02** (câble débranché) :

```
ansible -m ping clients_debian
192.168.1.11 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.12 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Tester la connectivité de tous les clients

```
ansible -m ping all
192.168.1.11 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
192.168.1.12 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
```

```
}
192.168.1.13 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh: connect to host
192.168.1.13 port 22: No route to host",
    "unreachable": true
}
```

b)Modifier la cible d'exécution

Il existe plusieurs manières de définir la cible d'exécution des nœuds par exemple en enrichissant le fichier précédent avec des déclarations suivantes dans le fichier hosts, on peut utiliser d'autres commandes :

```
[clients_debian]
192.168.1.11 ansible_ssh_user=ansible
192.168.1.12 ansible_ssh_user=ansible

[clients_ubuntu]
192.168.1.13 ansible_ssh_user=ansible

[clients_linux]
192.168.1.11 ansible_ssh_user=ansible
192.168.1.12 ansible_ssh_user=ansible
192.168.1.13 ansible_ssh_user=ansible
```

Cible	Exemple de commande
Tous les nœuds (toutes les machines connues dans le fichier host)	ansible -m ping **all** ou ansible -m ping *****
Un seul nœud	ansible -m ping **client01**
Plusieurs nœuds	ansible -m ping **client01:client02** ou ansible -m ping **client01,client02**
Un groupe	ansible -m ping **clients_debian**
Plusieurs groupes	ansible -m ping **clients_debian:clients_ubuntu**
Exclure un groupe (ici tous les nœuds du groupe clients_linux qui n'apparaissent pas dans le groupe clients_ubuntu)	ansible -m ping **clients_linux :!clients_ubuntu**
Intersection de groupes (ici tous les nœuds qui apparaissent à la fois dans le groupe clients_linux ET clients_debian)	ansible -m ping **clients_linux:&clients_debian**

c)Exécuter une commande avec des paramètres

Pour utiliser une commande avec des paramètres supplémentaires, il faut utiliser le module Shell d'Ansible.

Par exemple pour obtenir l'espace disque les différentes partitions de la machine cible client01, on peut utiliser la commande :

```
ansible -m shell -a "df -h" client01
192.168.1.11 | CHANGED | rc=0 >>
Sys. de fichiers Taille Utilisé Dispo Uti% Monté sur
udev                969M       0  969M   0% /dev
tmpfs               199M    1,3M  198M   1% /run
/dev/sda1            15G    6,6G  7,4G  48% /
tmpfs               992M       0  992M   0% /dev/shm
tmpfs               5,0M    4,0K  5,0M   1% /run/lock
tmpfs               199M    108K  199M   1% /run/user/1000
tmpfs               199M     60K  199M   1% /run/user/1001
```

4) Principe de base de déploiement d'application

4.1) Déploiement simple

Pour installer une application simplement via Ansible, il est possible d'utiliser le module Shell.

Par exemple, commencer par vérifier que l'application rig (qui permet de générer des adresses postales aléatoires) n'est pas installée sur **client01**.

```
rig
bash: rig : commande introuvable
```

Ensuite, lancer l'installation à partir du **serveur** avec la commande :

```
ansible -m shell -a "sudo apt-get install rig" client01
192.168.1.11 | CHANGED | rc=0 >>
...
Lecture des listes de paquets…
Construction de l'arbre des dépendances…
Lecture des informations d'état…
Les NOUVEAUX paquets suivants seront installés :
  rig
0 mis à jour, 1 nouvellement installés, 0 à enlever et 0 non mis à jour.
...
(Lecture de la base de données... 187392 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../rig_1.11-1+b2_amd64.deb ...
Dépaquetage de rig (1.11-1+b2) ...
Paramétrage de rig (1.11-1+b2) ...
...
```

Cette méthode fonctionne dans notre cas, mais pose quelques problèmes :

- les systèmes d'exploitation n'utilisent pas tous la commande sudo pour obtenir des privilèges d'exécution élevée (sudo, su, pfexec, doas, pbrun, dzdo, ksu, runas, machinectl), ce qui signifie que notre action ne serait certainement pas effectuée de la même manière (ou pas exécutée du tout) dans l'ensemble de notre parc informatique hétérogène,
- il n'est pas possible de vérifier l'état du système cible avant et après installation (par exemple vérifier que l'application n'est pas déjà installée, que les dépôts sont à jour, qu'un service lié et démarré, que des fichiers de configuration sont présents…),
- dans le cas où un grand nombre d'opérations nécessaires pour effectuer le déploiement, la mise au point, l'enchaînement des opérations, le suivi des résultats ainsi que la maintenabilité de l'ensemble vont s'avérer complexe.

4.2) Déploiement en utilisant des playbooks

Afin de simplifier la création et la gestion de l'automatisation de tâches dans un environnement où les machines sont différentes, Ansible permet l'utilisation de fichiers appelés playbooks.

Un playbook est un simple fichier au format YAML définissant un certain nombre d'actions et de comportements qui est hébergé sur le nœud de contrôle. Lorsqu'un playbook est exécuté, le nœud de contrôle envoie les commandes aux nœuds gérés concernés, en les adaptant au système d'exploitation cible

Plusieurs playbook peuvent être définis, de manière à correspondre aux différentes tâches d'administration à automatiser dans l'entreprise (déploiement de postes, configuration de logiciels, mise à jour, utilisation d'imprimante...).

a)Description d'un playbook

Chaque ligne d'un fichier YAML définit une paire clés/valeur, le séparateur utilisé est le caractère « : ».

```
nom:potter
prenom:harry
```

Il est possible d'imbriquer des groupes d'éléments :

```
type:personnage
    nom:potter
    prenom:harry
```

Le niveau d'indentation (alignement des colonnes) indique l'appartenance d'une information à une autre :

```
type:personnage
      nom:potter
      prenom:harry
      adresse:
            numero: 4
            rue: |
                  Privet Drive
                  Little Whinging
            ville: Surrey
```

Ici, le personnage à un nom, un prénom, une adresse.

L'adresse est elle-même composée d'un numéro, d'une ou plusieurs rues et d'une ville.

Remarquez que puisque la rue peut être composée de plusieurs lignes, il faut indiquer le caractère « | ».

Les listes d'éléments sont précédées du caractère « - »

```
caractéristiques:
      - sort:Accio
        type: télékinésie
        effet: Attirer quelque chose
      - sort: Lumos
        type: environnement
        effet: Faire de la lumière
      - sort: Wingardium leviosa
        type: télékinésie
        effet:Soulever un objet
```

Attention, l'indentation est importante, il faut correctement aligner les éléments pour que le fichier YAML fonctionne.

De plus, il ne faut pas utiliser de tabulation, mais des caractères espaces pour la mise en forme.

Il est conseillé de configurer l'éditeur de texte que ce dernier remplace les tabulations par des espaces pour éviter toute erreur (les éditeurs type développement possèdent cette fonctionnalité). Dans tous les cas, il faut privilégier la lisibilité :

```
type:personnage
      nom          : potter
      prenom       : harry
      adresse      :
            numero     : 4
            rue        : |
                  Privet Drive
                  Little Whinging
            ville      :      Surrey
caractéristiques:
      - sort       : Accio
        type       : télékinésie
        effet      : Attirer quelque chose
      - sort       : Lumos
        type       : environnement
        effet      : Faire de la lumière
      - sort       : Wingardium leviosa
        type       : télékinésie
        effet      : Soulever un objet
```

La forme générale d'un playbook Ansible est la suivante (les *valeurs mises en évidence* sont remplacées dans le fichier final) :

```
- hosts: nom des nœuds ou des groupes
  become: yes ou no pour indiquer si une élévation des privilèges est nécessaire (root ou administrateur)
  tasks:
    - name: nom de la tâche 1
          nom du module 1: paramètre du module = valeur – les paramètres dépendent du module utilisé
    - name: nom de la tâche 2
          nom du module 2: paramètre du module = valeur – les paramètres dépendent du module utilisé
```

b)Exploitation d'un playbook

Le fichier suivant permet de déployer le serveur web nginx, de le démarrer et de (les *commentaires sont mis en évidence* et n'apparaissent pas dans le fichier final) :

```
- hosts: client01 cible de l'automatisation
  become: yes nécessite une élévation des privilèges
  tasks:
    - name: Installation de nginx Description de la tâche
        apt:Module apt, permet d'installer la dernière versions de nginx à partir des dépôts configurés sur la cible
          name :nginx
          state :latest
```

```
  - name: Démarrer nginx Description de la tâche
    service: Module service, permet de contrôler les services
       name: nginx nom du service
       state: started État attendu : démarré
  - name: Récupérer depuis quand la machine fonctionne
    shell: uptime Module shell : ici permet d'exécuter la commande uptime
    register: var_affichage
Définit une variable nommée var_affichage contenant les résultats de la commande de même niveau (ici uptime)

  - debug: Affichage des informations de débogage
        var: var_affichage.stdout_lines Contenu à afficher
```

Le fichier est enregistré sous ~/.ansible/ install_nginx.yaml

Note

Les lignes :

```
    apt:
      name : nginx
      state : latest
```

peuvent être simplifiées en :

```
    apt: name=nginx state=latest
```

Pour vérifier la syntaxe du fichier, utiliser la commande (aucune modification n'est effectuée) :

```
ansible-playbook -i ~/.ansible/ install_nginx.yaml --check
```

Pour lancer le playbook (et réellement exécuter les opérations) :

```
ansible-playbook -i ~/.ansible/ install_nginx.yaml
PLAY [client01]
*********************************************************************************
TASK [Gathering Facts]
*********************************************************************************
ok: [192.168.1.11]

TASK [Installation de nginx]
*********************************************************************************
changed: [192.168.1.11]

TASK [Démarrer nginx]
*********************************************************************************
ok: [192.168.1.11]

TASK [Récupérer depuis quand la machine fonctionne]
*********************************************************************************
changed: [192.168.1.11]

TASK [debug]
*********************************************************************************
ok: [192.168.1.11] => {
    "command_output.stdout_lines": [
        " 20:20:56 up  7:08,  2 users,  load average: 0,50, 0,21, 0,07"
    ]
}
```

```
PLAY RECAP
*********************************************************************************
192.168.1.11               : ok=6    changed=2    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Pour vérifier que le serveur web soit bien opérationnel, il suffit de faire :

```
ansible -m shell -a "sudo systemctl status nginx" client01
...
192.168.1.11 | CHANGED | rc=0 >>
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
enabled)
     Active: active (running) since Tue 2022-04-05 20:20:52 CEST; 13min ago
…
```

A partir de n'importe quel poste, ouvrir un navigateur pointant sur l'adresse du **client01** (192.168.1.11). Vous devez obtenir la page par défaut de Nginx :

Pour modifier la cible du playbook, il suffit de changer le fichier .yaml, puis relancer le playbook. Par exemple, pour installer désormais sur les **client01** et **client02** :

```
- hosts: client01:client02
```

c)Exploitation d'un playbook de rollback

Nous allons maintenant écrire une second playbook permettant d'effacer les modifications effectuées par le playbook précédent. Ces playbook effectuent des actions dites de rollback, c'est à dire de retour à l'état initial.

Ces playbook de rollback sont important, puisqu'ils permettent de s'assurer qu'un système retrouve son état d'origine (avant modification), par exemple dans le cas où une migration se passerait mal.

> Le playbook de rollback ne remplace pas une sauvegarde, qui reste obligatoire.

Contenu du fichier deinstall_nginx.yaml

```
- hosts: client01
  become: yes
  tasks:
    - name: Arrêter nginx
      service:
```

```
      name: nginx
      state: stopped Arrêt du service Nginx
  - name: Supprimer le service nginx
    apt:
      name: nginx
      state: absent
  - name: Supprimer l'utilisateur nginx
    user:
      name: nginx
      state: absent Suppression du de l'utilisateur nginx
  - name: Supprimer le groupe nginx
    group:
      name: nginx
      state: absent
  - name: Supprimer les logs de nginx
    file:
      path: '/var/log/nginx'
      state: absent Suppression des logs Nginx
  - name: supprimer le répertoire d'installation de nginx
    file:
      path: '/etc/nginx'
      state: absent Suppression du répertoire d'installation
```

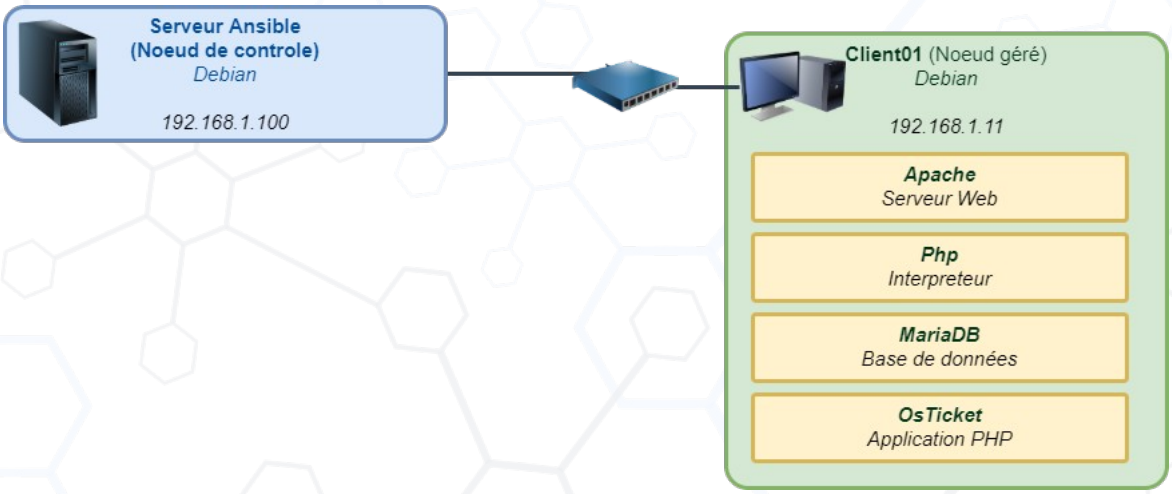
Puis exécution :

```
ansible-playbook deinstall_nginx.yaml
PLAY [client01]
*********************************************************************************
TASK [Gathering Facts]
*********************************************************************************
ok: [192.168.1.11]

TASK [Arrêter nginx]
*********************************************************************************
changed: [192.168.1.11]

TASK [Supprimer le service nginx]
*********************************************************************************
changed: [192.168.1.11]

TASK [Supprimer l'utilisateur nginx]
*********************************************************************************
changed: [192.168.1.11]

TASK [Supprimer le groupe nginx]
*********************************************************************************
changed: [192.168.1.11]

TASK [Supprimer les logs de nginx]
*********************************************************************************
changed: [192.168.1.11]

TASK [supprimer le répertoire d'installation de nginx]
*********************************************************************************
changed: [192.168.1.11]

PLAY RECAP
```

```
*****************************************************************************
192.168.1.11                : ok=7    changed=6    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

5) Automatisations multiples

5.1) Architecture cible du déploiement



Il est possible d'effectuer des déploiements plus complexes. Par la suite, nous allons déployer une architecture multi tiers sur le **client01**. Cette architecture contiendra plusieurs éléments techniques :

- Un serveur Web permettant d'héberger les fichiers
- un interpréteur PHP permettant d'exécuter la partie traitement de l'application
- une base de données Maria DB contenant les informations nécessaires à l'exécution de l'application
- l'application elle-même.

L'installation de ces éléments sera réalisée de manière automatique, les configurations étant réalisées par Ansible.

Un playbook sera réalisé par éléments techniques (Un pour l'installation d'Apache, un pour celle de PHP, un pour celle de MariaDB, Un pour celle d'OsTicket).

Cette architecture modulaire permettra la réutilisation des différents playbooks pour d'autres projets éventuels. De la même manière il sera possible de modifier la cible de déploiement de manière à réaliser des installations automatiques sur d'autres machines.

5.2) Construction des playbooks

a)Déploiement d'Apache

Contenu du fichier apache.yaml :

```
- hosts: client01
  become: yes
  tasks:
    - name: Installer le serveur Apache
      apt:
        name: apache2
        state: present Installation du service apache
    - name: S'assurer qu'Apache est actif au démarrage du système
      service:
        name: apache2
        state: started Démarrer le service tout de suite
        enabled: yes Démarrer le service lors du démarrage du système
```

```
ansible-playbook apache.yaml
PLAY [client01]
*********************************************************************************
TASK [Gathering Facts]
*********************************************************************************
ok: [192.168.1.11]

TASK [Installer le serveur Apache]
*********************************************************************************
changed: [192.168.1.11]

TASK [S'assurer qu'Apache est actif au démarrage du système]
*********************************************************************************
ok: [192.168.1.11]

PLAY RECAP
*********************************************************************************
192.168.1.11               : ok=3    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

Vérification sur **Client01** :

```
sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
enabled)
     Active: active (running) since Thu 2022-04-07 16:56:51 CEST; 1min 41s ago
```

b)Déploiement de PHP

Contenu du fichier php.yaml :

```
- hosts: client01
  become: yes
  tasks:
    - name: php requirements
```

```
    apt:
      name: Définir la liste des paquets PHP à installer
        - libapache2-mod-php
        - php-cli
        - php-fpm
        - php-mysqlnd
        - php-json
        - php-mbstring
        - php-xml
        - php-gd
        - php-curl
        - php-php-gettext
        - php-intl
        - php-opcache
        - php-apcu
        - php-pecl-http
        - php-common
        - php-imap
      state: latest Installer la dernière version des paquets
```

```
ansible-playbook php.yaml
PLAY [client01]
*********************************************************************************
TASK [Gathering Facts]
*********************************************************************************
ok: [192.168.1.11]

TASK [php requirements]
*********************************************************************************
changed: [192.168.1.11]

PLAY RECAP
*********************************************************************************
192.168.1.11               : ok=2    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

c)Déploiement de MariaDB

Contenu du fichier mariadb.yaml

```
- hosts: client01
  become: yes
  tasks:
    - name: Installer les utilitaires
      apt:
        name: Définir la liste des prérequis
          - curl
          - software-properties-common
          - gnupg2
          - python3
          - pip
        state: latest Installer la dernière version
```

```
  - name: Vérifier la présence de PyMySQL
    pip:
        name: pymysql Installer PyMySQL qui permet à Ansible de contrôler la base de données MariaDB
      state: present S'assurer que le module python est présent
  - name: Récupérer la configuration du dépôt de MariaDB
    get_url: Télécharger le fichier à partir du dépôt de MariaDB
      url: https://downloads.mariadb.com/MariaDB/mariadb_repo_setup
      dest: . Le placer dans le répertoire en cours
  - name: Ajouter le dépot MariaDB
      command: bash mariadb_repo_setup --mariadb-server-version=10.5 Lancer le script venant d'être téléchargé
  - name: Installer MariaDB
    apt:
      name: Installer les utilitaires MariaDB
        - mariadb-server
        - mariadb-client
      state: latest

  - name: Démarrer MariaDB
    service:
      name: mariadb
      state: started Démarrer le service immédiatement
      enabled: yes Le rendre disponible au redémarrage du système
  - name: Définir le mot de passe de root pour MySQL
    mysql_user:
        login_unix_socket: /var/run/mysqld/mysqld.sock Définir la méthode de connexion utilisant le socket réseau
      login_user: root Utilisateur actuellement autorisé à modifier les informations de la base de données
      login_password: admin Mot de passe correspondant
      name: root Définir l'utilisateur de la base de données
      password: admin Définir le mot de passe de la base de données
      state: present
  - name: Copier le fichier .my.cnf file contenant les informations d'identification
    copy: Copier du fichier contenant les identifiants de l'utilisateur root
      src: my.cnf
      dest: /root/.my.cnf
      owner: root Définir le propriétaire du fichier
      mode: 0600 Définir les droits d'accès des utilisateurs sur le fichier
  - name: Créer la base de données pour OSTicket
    become: yes
    mysql_db:
      login_unix_socket: /var/run/mysqld/mysqld.sock
      login_user: root
      login_password: admin
      name: "osticket" Création de la base de données nommées osticket
      state: present
  - name: Créer un utilisateur pour la base de données
    mysql_user: Création d'un utilisateur spécifique pour la base de données OsTicket
      login_unix_socket: /var/run/mysqld/mysqld.sock
```

```
        login_user: root Utilisateurs actuellement autorisés à modifier les informations de la base de données
      login_password: admin Mot de passe correspondant
      name: "osticket_admin" Définir l'utilisateur spécifique de la base de données OsTicket
        password: "123AZEqsd!" Définir le mot de passe de la base de données pour cet utilisateur
        host: "%" Machines à partir desquels l'utilisateur pourra se connecter à la base de donner – Ici le caractère % indique n'importe quelle machine
      priv: "osticket.*:ALL,GRANT" Définir les droits SQL de l'utilisateur Sur tous les objets de la base osticket. Ici tous les droits sont accordés, y compris les droits de délégation des droits à d'autres utilisateurs
      state: present
```

Contenu du fichier my.cnf

```
[client]
user=root
password=123AZEqsd!
```

```
ansible-playbook mariadb.yaml
PLAY [client01] ****************************************************************
TASK [Gathering Facts] *********************************************************
ok: [192.168.1.11]

TASK [Installer les utilitaires] ***********************************************
ok: [192.168.1.11]

TASK [Verifier la présence de PyMySQL] *****************************************
ok: [192.168.1.11]

TASK [Récupérer la configuration du dépôt de MariaDB] **************************
ok: [192.168.1.11]

TASK [Ajouter le dépot MariaDB] ************************************************
changed: [192.168.1.11]

TASK [Installer MariaDB] *******************************************************
changed: [192.168.1.11]

TASK [Démarrer MariaDB] ********************************************************
ok: [192.168.1.11]

TASK [Définir le mot de passe de root pour MySQL] ******************************
ok: [192.168.1.11]

TASK [Copier le fichier .my.cnf file contenant les informations d'identification]
***
ok: [192.168.1.11]

TASK [Créer la base de données pour OSTicket] **********************************
changed: [192.168.1.11]

TASK [Créer un utilisateur pour la base de données] ****************************
ok: [192.168.1.11]

PLAY RECAP *********************************************************************
192.168.1.11               : ok=11   changed=3    unreachable=0    failed=0
```

```
skipped=0    rescued=0    ignored=0
```

d)Déploiement d'OSTicket

Contenu du fichier osticket.yaml

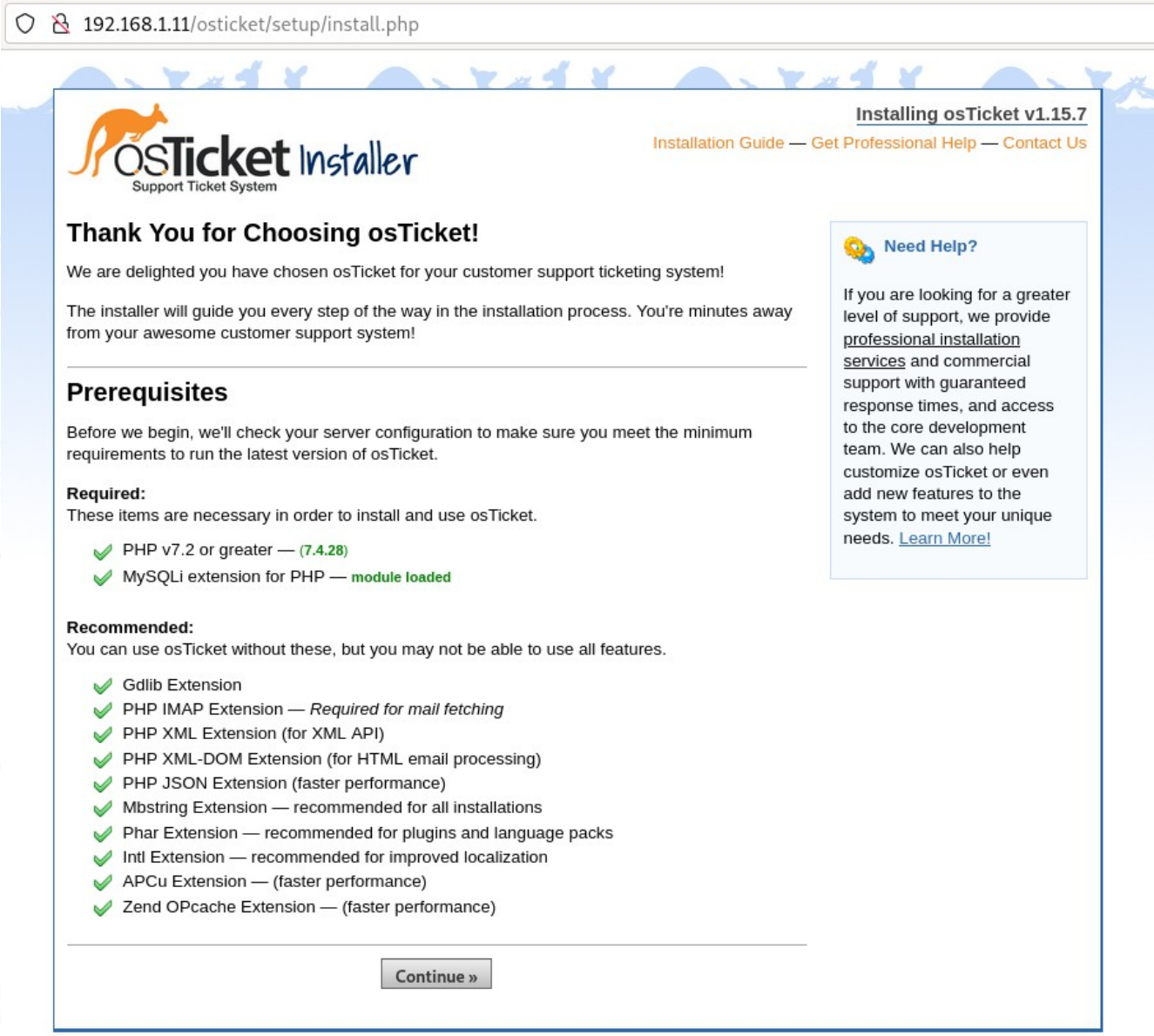
```
- hosts: client01
  become: yes
  tasks:
    - name: Recuperation des fichiers OsTicket sur le dépôt GIT
      git: Utilisation de la commande GIT pour cloner un dépôt
         repo: https://github.com/osTicket/osTicket Chemin du dépôt – tous les fichiers de ce dépôt seront récupérés
        dest: /opt/osticket Chemin de destination des fichiers copiés
        version: "1.15.x" Version d'OsTicket récupérée
      become_user: root

    - name: Déploiement d'OsTicket via PHP
         command: php manage.php deploy --setup /var/www/html/osticket Installation des fichiers du dépôt en utilisant des commandes PHP
      args:
        chdir: /opt/osticket Chemin à partir duquel la commande PHP sera exécutée
      become: yes

    - name: Création du fichier de configuration ost-config.php
      copy: Copie du fichier de configuration par défaut
        src: /var/www/html/osticket/include/ost-sampleconfig.php Définir le fichier source
        dest: /var/www/html/osticket/include/ost-config.php Définir le fichier de destination
        owner: www-data Définir l'utilisateur propriétaire du fichier
        group: www-data Définir le groupe propriétaire du fichier
        remote_src: yes
      become: yes

    - name: Récupération des plugins OsTicket
      git: Récupération des plugins d'OsTicket
        repo: https://github.com/osTicket/osTicket-plugins.git Dépôt à cloner
        dest: /opt/osticket-plugins Destination des fichiers
      become: yes

    - name: Déploiement des plugins
      copy: Copie des fichiers dans le répertoire de destination Apache
        src: /opt/osticket-plugins/ Répertoire source
        dest: /var/www/html/osticket/include/plugins/ Répertoire destination
        remote_src: yes
        group: www-data Définir le groupe propriétaire des fichiers
        owner: www-data Définir l'utilisateur propriétaire des fichiers
      become: yes

    - name: Installation des plugins principaux via PHP
         command: php make.php hydrate Installation des plug-ins en utilisant la commande PHP
```

```
    args:
        chdir: /var/www/html/osticket/include/plugins/ Chemin à partir duquel la commande PHP sera exécuté
    become_user: www-data Utilisateur autorisé à exécuter la commande

  - name: Ajustement des permissions pour Apache
    file:
        path: /var/www/html/osticket Racine des fichiers dont les permissions doivent être modifiées
      state: directory Type de structure concernée – ici un répertoire
      owner: www-data Définir le propriétaire des fichiers
      group: www-data Définir le groupe propriétaire des fichiers
      recurse: yes Modifier les permissions récursivement
    become: yes
```

ansible-playbook osticket.yaml

```
PLAY [client01] ************************************************************

TASK [Gathering Facts] *****************************************************
ok: [192.168.1.11]

TASK [Recuperation des fichiers OsTicket sur le dépôt GIT] *****************
ok: [192.168.1.11]

TASK [Déploiement d'OsTicket via PHP] **************************************
changed: [192.168.1.11]

TASK [Création du fichier de configuration ost-config.php] *****************
ok: [192.168.1.11]

TASK [Récupération des plugins OsTicket] ***********************************
ok: [192.168.1.11]

TASK [Déploiement des plugins] *********************************************
changed: [192.168.1.11]

TASK [Installation des plugins principaux via PHP] *************************
changed: [192.168.1.11]

TASK [Ajustement des permissions pour Apache] ******************************
changed: [192.168.1.11]

PLAY RECAP *****************************************************************
192.168.1.11               : ok=8    changed=4    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

À l'issue de l'exécution des différents playbooks, un navigateur pointé sur l'adresse du **Client01 (**192.168.1.11) permet d'obtenir l'interface graphique de finalisation de l'installation d'OsTicket.



6) Sites d'intérêt

https://www.redhat.com/fr/topics/automation/learning-ansible-tutorial	Apprendre les bases d'Ansible