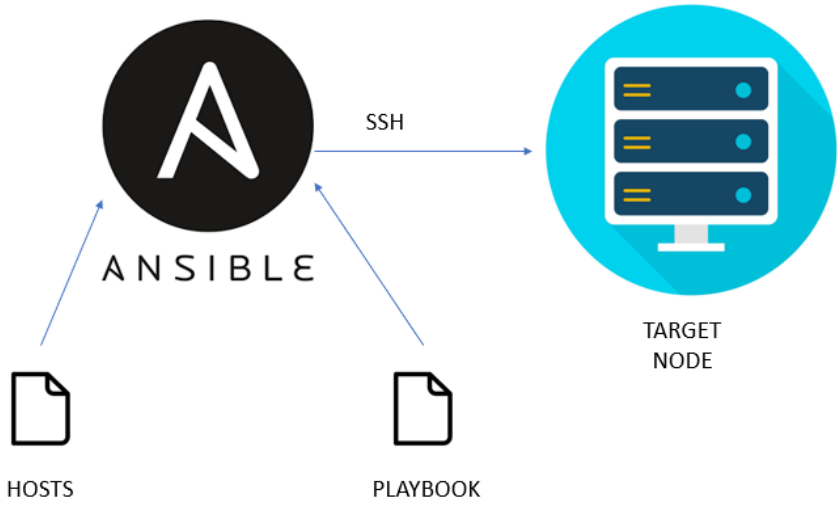
LICENCE DEV
Lab Ansible

Patrice Gommery – Mars 2022



PRE-REQUIS

Accéder au VM dédiées à Docker dans la salle H205.
Avoir reçu son **VMID** (N° de machine dans la salle H205)

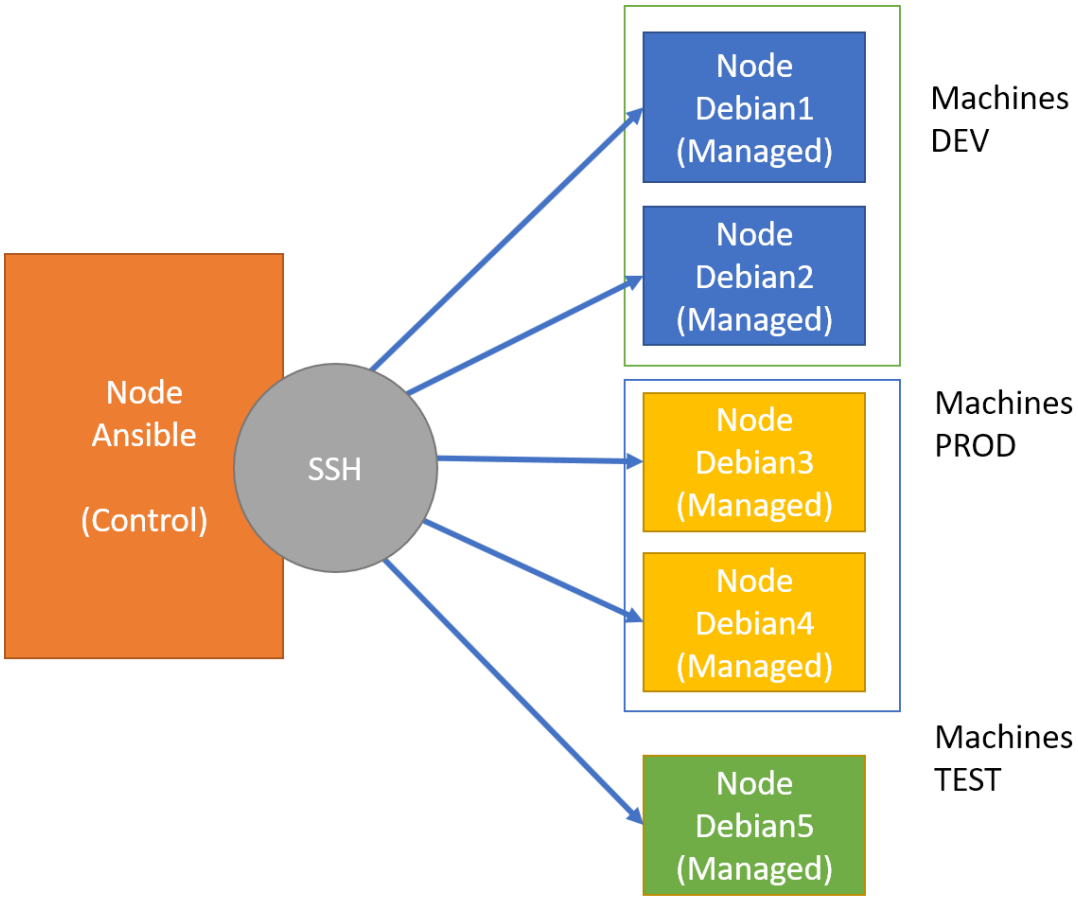
Si vous êtes dans la salle H205 (sur un poste de la salle) :
Ouvrez un terminal et connectez-vous avec ssh root@172.16.**VM.ID** (PASSWORD)

Si vous n'êtes pas sur un des postes de la salle H205 :
Ouvrez un terminal et connectez-vous avec :
ssh vpnmmi@vpnmmi.h205.online (Demandez le mot de passe à l'enseignant)
ssh licdev@192.168.3.2 (Même mot de passe que vpnmmi)
et pour terminer : **ssh root@172.16.VM.ID (PASSWORD)**

Préambule :

Ansible est un outil de déploiement utilisé pour la configuration et la mise à jour de parc machines sous Linux. Sa particularité , contrairement à d'autre outil comme puppet est qu'il ne nécessite pas l'installation d'agent sur les postes cibles. Il fonctionne simplement avec SSH et utilise le langage python.

Le Lab Ansible



Pour cette initiation, nous allons construire (avec Docker) un Lab de test nous permettant de simuler un parc de machines sous Debian. Notre Lab sera constitué d'une machine sur laquelle nous installerons ansible, de machines de DEV, 2 machines de PROD et une machine de TEST pour les premiers essais. L'usage de Docker est ici doublement intéressant parce qu'il va nous permettre de mettre en place ce lab rapidement, mais aussi parce qu'il permettra une remise à zéro des machines rapide en cas d'échecs des tests.

N'hésitez pas à reprendre les PDF des deux premières séances pour réaliser ce lab en complète autonomie.

Avant de commencer je vous propose de faire un peu de ménage sur vos machines

IMPORTANT : N'effacez pas les fichiers docker-compose.yml que vous avez créés lors de la séance précédente. Celui du premier exercice (wordpress) doit se trouver dans /root/exo, celui du second exercice (wireshark) doit se trouver dans /root/exo2 .

Pour faire le ménage :

```
docker container stop $(docker container ls -qa)
docker container rm $(docker container ls -qa)
docker image rm $(docker image ls -q)
docker volume rm $(docker volume ls -q)
```

Partie 1 : IMAGES

Dans notre lab, nous aurons donc besoin de deux types de machines : Le node ANSIBLE et les nodes Debian. Quel que soit le type, nous baserons toutes les machines sur la même image : debian:buster-slim (Une version allégée de Debian10) . Commencez par télécharger cette image sur votre serveur :

```
docker image pull debian:buster-slim
```

Ensuite, en utilisant des Dockerfile et cette image de base, créez deux autres images avec les caractéristiques suivantes :

Image node-buster:1.0
paquets à installer :

- nano
- openssh-server
- openssh-client
- net-tools
- iputils-ping
- python3

Image node-ansible:1.0
paquets à installer:

même chose que node-buster avec ansible en plus.

Pour les deux images, modifier le fichier **sshd_config** pour autoriser la connexion **ssh** avec **root** en utilisant les commandes suivantes :

```
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
RUN sed -i 's/#PermitRootLogin/PermitRootLogin/' /etc/ssh/sshd_config
```

Forcez le mot de passe de root à **password** avec la ligne:

```
RUN sed -i 's#root:\*#root:sa3tHJ3/KuYvI#' /etc/shadow
```

Terminez le Dockerfile par :

```
RUN mkdir /run/sshd
ENTRYPOINT ["/usr/sbin/sshd"]
CMD ["-D"]
```

Testez vos images en créant des containers temporaires .

- Vérifiez leurs IP (ifconfig) , testez la connexion ssh avec root (et password)
- vérifiez la version d'ansible sur le container ansible (ansible --version)
- testez python avec la commande python3

EXERCICE : (Important pour la suite)
Sur votre VM, créez un dossier **/root/ansible** *(il nous servira de volume dans notre stack lab)*
Dans ce dossier recopier les fichiers **ansible.cfg** et **hosts** de votre container de test pour l'image node-ansible. ASTUCE : Une fois dans le container , vous pouvez utiliser la commande **scp** pour envoyer directement les fichiers dans le dossier de votre VM.

N'oubliez pas de supprimer vos containers (pas vos images) de test avant de continuer

Partie 2 : STACK LAB

Vous avez maintenant deux images fonctionnelles pour mettre en place notre lab complet .

```
docker image ls
REPOSITORY      TAG           IMAGE ID       CREATED          SIZE
node-ansible    1.0          bc8ae091e1e9   6 minutes ago    238MB
node-buster     1.0          7c8afb433ecf   43 minutes ago   128MB
```

Il ne vous reste plus qu' à utiliser docker-compose pour construire le lab dans sa globalité.
Les contraintes sont les suivantes :

Pour commencer votre fichier **docker-compose.yml**, utilisez :

```
version: "3.8"
```

Description de l'environnement :

Un container nommé **ansible** qui contiendra :
Image : **node-ansible :1.0**
Volume local : **/root/ansible** redirigé vers **/etc/ansible** sur le container
pas de port à exposer

Un container nommé debian-test qui contiendra
Image : **node-buster :1.0**
Exposition du port **80** , redirection vers leport **8081** local
Pas de volumes

Deux containers nommé dev1 et dev2 qui contiendrons
Image : **node-buster :1.0**
Exposition du port **80** , redirection vers leport **8082** local pour dev1
Exposition du port **80** , redirection vers leport **8083** local pour dev2
Pas de volumes

Deux containers nommé prod1 et prod2 qui contiendrons
Image : **node-buster :1.0**
Exposition du port **80** , redirection vers leport **8084** local pour prod1
Exposition du port **80** , redirection vers leport **8085** local pour prod2
Pas de volumes

Une fois le **docker-compose.yml** créé, montez votre stack avec la commande :

```
docker-compose up -d
```

Votre stack est maintenant démarré. Vous pouvez le vérifier avec la commande :

```
docker container ls
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                    NAMES
b9adaf7f5e54   node-buster:1.0       "/usr/sbin/sshd -D"      11 seconds ago   Up 3 seconds    0.0.0.0:8081->80/tcp     debian-test
9b7688e43ce2   node-buster:1.0       "/usr/sbin/sshd -D"      11 seconds ago   Up 5 seconds    0.0.0.0:8085->80/tcp     prod2
0954ae240626   node-buster:1.0       "/usr/sbin/sshd -D"      11 seconds ago   Up 4 seconds    0.0.0.0:8083->80/tcp     dev2
be1867ee17df   node-ansible:1.0      "/usr/sbin/sshd -D"      11 seconds ago   Up 3 seconds                             ansible
c0519d92de93   node-buster:1.0       "/usr/sbin/sshd -D"      11 seconds ago   Up 6 seconds    0.0.0.0:8082->80/tcp     dev1
5cc6acb97d68   node-buster:1.0       "/usr/sbin/sshd -D"      11 seconds ago   Up 6 seconds    0.0.0.0:8084->80/tcp     prod1
```

Connectez-vous au container ansible avec la commande :

```
Docker container exec -ti ansible /bin/sh
```

Et vérifier que le dossier /etc/ansible pointe bien sur notre volume /root/ansible
(Créez un fichier dans un des dossiers et vérifiez qu'il apparait bien dans l'autre)

Partie 3 : SSH

Toutes les machines sont donc accessibles via ssh.
Pour retrouver leurs IP , vous pouvez utiliser la commande (exemple pour la machine ansible)

```
docker container inspect ansible | grep IPAddress
```

```
            "SecondaryIPAddresses": null,
            "IPAddress": "",
                    "IPAddress": "172.28.0.7",
```

Vous pouvez donc accéder aux containers directement en ssh avec le compte root et le mot de passe password. Plus besoin de faire un docker container exec .(Exemple pour la machine ansible)

```
ssh root@172.28.0.7   (Le mot de passe est password en minuscules)
```

```
root@172.28.0.7's password:
Linux be1867ee17df 4.19.0-14-amd64 #1 SMP Debian 4.19.171-2 (2021-01-30) x86_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.
root@be1867ee17df:~#
```

EXERCICE : (Important pour la suite)
Relevez les différentes IP de vos containers, vous en aurez besoin un peu plus loin

Comme indiqué en début de TP, Ansible se sert donc aussi de SSH pour communiquer entre le nœud de contrôle (node ansible) et les autres machines du lab . Mais comment faire pour éviter de saisir le mot de passe à chaque connexion SSH ? Tout simplement en utilisant une authentification par clé. La démarche est la suivante :

1. Connectez-vous sur le container ansible (en ssh par exemple)
2. Générez une paire de clé (privée/publique) avec la commande :

```
ssh-keygen -t ecdsa
```

IMPORTANT : Validez les choix par défaut avec ENTREE , NE Saisissez pas de passphrase !!

Vous avez maintenant deux fichiers dans votre dossier /root/.ssh :
id_ecdsa (qui est votre clé privée), id_ecdsa.pub (votre clé publique)

3. Envoyez la clé publique sur les autres conteneurs avec la commande :

```
ssh-copy-id root@172.28.0.2   (Remplacez l'IP par celles de vos containers)
```

Répétez l'opération pour tous les containers (sauf ansible bien sûr) .
Le container ansible pourra ainsi communiquer avec les autres sans aucun mot de passe.
Vous pouvez tester directement avec :

```
ssh 172.28.0.2  ifconfig
```

Pas besoin de mot de passe, la commande **ifconfig** s'exécute directement sur le container distant.

Voilà notre lab est prêt à être utilisé. Pour repartir à zéro si le besoin s'en fait sentir, il suffira de démonter le stack (docker-compose down) et de le remonter (docker-compose up -d). Bien sûr il faudra régénérer les clés SSH et les recopier, mais ce n'est pas le plus compliqué (On pourrait aussi automatiser cela)

Partie 4 : ANSIBLE

Nous voilà donc arrivé au cœur de ce cours, nous n'allons pas voir ansible en détail (cela pourrait faire l'objet d'un module complet), mais simplement voir rapidement comment celui-ci fonctionne et quelles possibilités il offre.

ATTENTION : Pour la suite du TP, connectez-vous en SSH sur le container ansible.
Tous les fichiers sont à créer dans le dossier /etc/ansible de cette machine

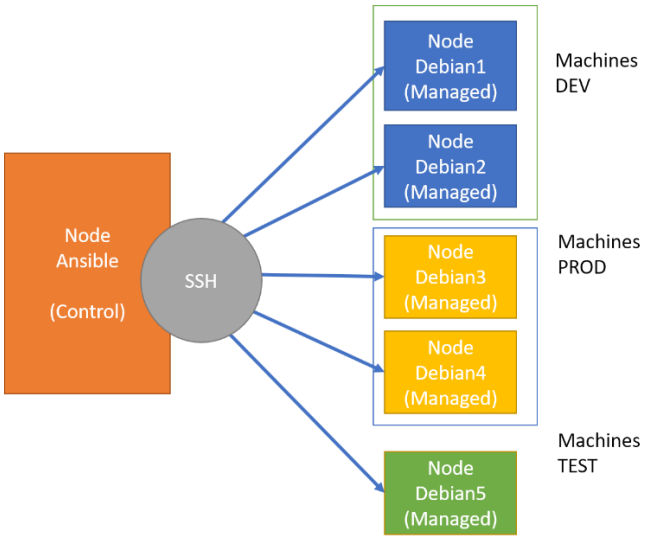
4a – ANSIBLE - INVENTORY

Premier élément important d'Ansible : l'INVENTORY qui comme son nom l'indique permet de dresser l'inventaire des nœuds (machines ou containers) sur lesquels seront effectuées les actions d'Ansible. Toutefois, nous le verrons juste après l'INVENTORY ne se limite pas à cela et permet aussi de stocker de nombreuses variables .

L'INVENTORY peut être décrit sous plusieurs formes :

Un fichier à plat (/etc/ansible/hosts)
Une structure de fichiers au format yaml.
Une structure de fichiers au format json.

Pour notre exercice, nous choisirons la première option (vous en avez un exemple dans /etc/ansible). Dans ce fichier, nous allons décrire l'organisation de notre lab tel que nous l'avons vu sur le schéma suivant :



Le fichier va donc répertorier deux types d'instances : les machines (hosts) et les groupes de machines (groups). Une machine peut bien sur appartenir à plusieurs groupes, il est même possible de faire des sous-groupes. Notez qu'il existe un groupe implicite qui désigne toutes les machines : **all**

Pour continuer, placez-vous dans le dossier **/etc/ansible** et créez un fichier nommé : **inventory.ini** avec le contenu suivant :

```
[all:vars]
ansible_python_interpreter=/usr/bin/python3
ansible_ssh_users=root
ansible_ssh_port=22

[test]
172.28.0.6

[dev]
172.28.0.2
172.28.0.5

[prod]
172.28.0.3
172.28.0.4
```

Remplacez les IP par celles de vos machines !!!

[all :vars] regroupe les variables applicables à TOUTES les machines.
Les variables utilisées dans notre cas :

ansible_python_interpreter=/usr/bin/python3
indique a ansible qu'il faut utiliser python3 sur les machines

ansible_ssh_users=root et **ansible_ssh_port=22**
indiquent à ansible qu'il faut utiliser le port 22 pour SSH et se connecter avec l'utilisateur root

Notez que ces variables peuvent être surchargées au niveau des groupes et même des machines.
On pourrait par exemple , redéclarer les variables et changer leurs valeurs pour le groupe test sans affecter les autres machines en créant une section [test :vars].

[test] [dev] [prod] permettent de regrouper les machines par groupe et donc d'effectuer des actions groupées en fonction des besoins.

Notez que si nous avions mis un place en environnement de résolution des noms (DNS) , nous aurions pu remplacer les IP par les noms de machines.

Notre fichier est prêt, il nous reste maintenant à indiquer à ansible ou il se trouve.
Pour cela éditer le fichier **/etc/ansible/ansible.cfg** et modifier la ligne inventory

```
inventory      = /etc/ansible/inventory.ini
```

N'oubliez d'enlever le # devant la ligne pour prendre en compte notre personnalisation.

Testons maintenant notre inventaire avec notre première commande ansible :

```
ansible all -m ping --one-line
```

```
172.28.0.6 | SUCCESS => {"changed": false,"ping": "pong"}
172.28.0.3 | SUCCESS => {"changed": false,"ping": "pong"}
172.28.0.2 | SUCCESS => {"changed": false,"ping": "pong"}
172.28.0.5 | SUCCESS => {"changed": false,"ping": "pong"}
172.28.0.4 | SUCCESS => {"changed": false,"ping": "pong"}
```

Et voilà ansible vous à réaliser un superbe ping de toutes vos machines avec une seule commande.
c'est magique !!! et ça ne s'arrête bien sûr pas là.

4b – ANSIBLE – LIGNE DE COMMANDES

Même si ce n'est pas l'usage le plus courant d'ansible (On utilise plutôt des Playbook), la ligne de commandes ansible reste un bon moyen de tester les possibilités du produit, de vérifier le passage des variables etc …

La syntaxe de base est assez simple. Regardons de plus près notre première commande :

```
ansible all -m ping --one-line
```

ansible : la commande de base
--one-line : Force le résultat de chaque nœud sur une seule ligne.
all : Désigne TOUTES les machines
On peut adresser une commande à une machine ou un groupe de machines en particulier. Essayez :

```
ansible test -m ping --one-line
ansible dev -m ping --one-line
ansible prod -m ping --one-line
ansible 172.28.0.5 -m ping --one-line
```

-m ping : Désigne le module **ping** (pas la commande ping !)
Ansible est fourni avec des modules . Chaque module est spécialisé sur un type d'action et peut accepter différents arguments. Quelques modules à découvrir :

command (pour envoyer des commandes système simples)
shell (pour exécuter une commande dans un shell distant)
apt (pour utiliser le gestionnaire de paquets de Debian et donc installer les applications)
service (pour démarrer, arrêter , recharger un service)
copy (pour copier des fichiers)
file (pour créer et gérer des fichiers et des dossiers)
fetch (pour récupérer des fichiers depuis un nœud vers la machine ansible)

Tous ces modules sont en fait des programmes en python qui sont envoyés (via scp) et exécutés sur les machines (avec l'interpréteur python). Chaque module à ses propres particularités et ses options, mais la documentation d'Ansible est très complète .
Elle est même accessible directement avec la commande **ansible-doc**
Exemple pour le module ping :

```
ansible-doc ping
```

Attention, toutefois certains modules ne peuvent pas être utilisés directement en ligne de commandes, ils nécessitent obligatoirement le passage par un Playbook .

4c – ANSIBLE – PLAYBOOK

Les PLAYBOOK sont le cœur d'Ansible. C'est en effet grâce à ces fichiers et à leur exécution via la commande **ansible-playbook** que nous allons pouvoir automatiser toutes les actions que nous voulons réaliser sur nos machines.

Un playbook est écrit au format **yaml** et va permettre de décrire toutes les actions que nous voulons réaliser sur une machine ou un groupe de machines. Voici par exemple un playbook qui va nous permettre d'installer php sur nos machines du groupe **dev** :

Contenu de **playbook1.yml**

```
- name: playbook1
  hosts: dev
  remote_user: root
  tasks:
  - name: install php
    apt:
      name: php
      state: latest
      update_cache: yes
```

Pour l'exécuter :

```
ansible-playbook playbook1.yml
```

Le résultat :

```
root@be1867ee17df:/etc/ansible# ansible-playbook playbook.yml

PLAY [playbook] ************************************************************************************************************************************************

TASK [Gathering Facts] *****************************************************************************************************************************************
ok: [172.28.0.2]
ok: [172.28.0.5]

TASK [install php] *********************************************************************************************************************************************
changed: [172.28.0.5]
changed: [172.28.0.2]

PLAY RECAP *****************************************************************************************************************************************************
172.28.0.2                 : ok=2    changed=1    unreachable=0    failed=0
172.28.0.5                 : ok=2    changed=1    unreachable=0    failed=0
```

On voit bien ici, que la tache a bien été réalisée sur les deux machines du groupe DEV.
Nous pouvons vérifier la version installée avec la commande :

```
ansible dev -m shell -a "php -v"
```

qui nous affiche donc la version de php installée sur les deux machines du groupe dev :

```
root@be1867ee17df:/etc/ansible# ansible dev -m shell -a "php -v"
172.28.0.5 | CHANGED | rc=0 >>
PHP 7.3.27-1~deb10u1 (cli) (built: Feb 13 2021 16:31:40) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.27, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.3.27-1~deb10u1, Copyright (c) 1999-2018, by Zend Technologies

172.28.0.2 | CHANGED | rc=0 >>
PHP 7.3.27-1~deb10u1 (cli) (built: Feb 13 2021 16:31:40) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.27, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.3.27-1~deb10u1, Copyright (c) 1999-2018, by Zend Technologies
```

Pour mieux comprendre l'écriture d'un playbook, nous allons en écrire un nouveau ensemble qui mettra en oeuvre le module File qui permet la manipulation des dossiers et des fichiers.
Commencez par créer un nouveau fichier : **playbook2.yml**

Le playbook commence comme ceci :

```
- name: Playbook2                    - le nom du playbook
  hosts: dev                         - la cible = une machine ou un groupe – ici les machines dev
  remote_user: root                  - L'utilisateur distant qui exécutera les scripts
  tasks:                             - Le début des tâches à effectuer par le playbook
```

ATTENTION à bien indenter les lignes avec des espaces (pas de tabulation) et à respecter les espaces

Déclarons une première tâche qui va juste tester la connexion avec les machines (module ping)

```
  tasks                              - Le début des tâches à effectuer par le playbook
   - name: test de connexion         - Le nom de la tâche
     ping:                           - le module utilisé
                                     - rien d'autre pour le module ping, il n'y a pas d'arguments
```

Enregistrez le fichier et réalisez un premier test :

```
ansible-playbook playbook2.yml
```

Le résultat *(s'il n'y a pas d'erreurs dans le fichier yaml)*

```
PLAY [Playbook2] ***************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [172.28.0.2]
ok: [172.28.0.5]

TASK [test de connexion] *******************************************************************************************************************************
ok: [172.28.0.2]
ok: [172.28.0.5]

PLAY RECAP *********************************************************************************************************************************************
172.28.0.2                 : ok=2    changed=0    unreachable=0    failed=0
172.28.0.5                 : ok=2    changed=0    unreachable=0    failed=0
```

Nous retrouvons les informations de notre playbook :

```
PLAY [Playbook2]                     - Le nom de notre Playbook
TASK [test de connexion]             - Le nom de la tâche effectuée
```

et aussi

```
TASK [Gathering Facts]               - La collecte des informations de la machine distante – tâche Ansible par défaut
PLAY RECAP                           - Le récap des tâches effectuées  avec les réussites et les échecs
```

Complétons maintenant notre playbook et essayons de créer un dossier sur les machines dev

```
  tasks                              - Le début des tâches à effectuer par le playbook
   - name: test de connexion         - Le nom de la tâche
     ping:                           - le module utilisé
                                     - rien d'autre pour le module ping, il n'y a pas d'arguments


   - name: création d'un dossier     - Nom de la tâche
     file:                           - Module
      path: /exo-ansible             - Nom du dossier (voir Documentation du module File)
      state: directory               - Type d'éléments à créer(ici un dossier)
      owner: root                    - Propriétaire du dossier
      group: root                    - Groupe propriétaire du dossier
      mode: 0755                     - Permissions du dossier
```

Exécutez de nouveau le playbook.

```
TASK [creation d'un dossier] ***************************************************************************************************************************
changed: [172.28.0.2]
changed: [172.28.0.5]

PLAY RECAP *********************************************************************************************************************************************
172.28.0.2                 : ok=3    changed=1    unreachable=0    failed=0
172.28.0.5                 : ok=3    changed=1    unreachable=0    failed=0
```

Ici, Ansible a effectué une modification sur les machines distantes (ce qui n'est pas le cas avec ping), il nous indique donc que le changement a été effectué (changed)

Si vous relancez de nouveau le playbook.

```
TASK [creation d'un dossier] ***************************************************************************************************************
ok: [172.28.0.5]
ok: [172.28.0.2]

PLAY RECAP *********************************************************************************************************************************
172.28.0.2                 : ok=3    changed=0    unreachable=0    failed=0
172.28.0.5                 : ok=3    changed=0    unreachable=0    failed=0
```

Ansible n'a pas effectué de changement, car le dossier existe déjà sur les machines cibles. Il nous signale juste qu'il a effectué correctement la tâche, mais qu'il n'a rien changé sur les machines.

Nous pouvons le vérifier par nous-même avec cette commande ssh :

ssh 172.28.0.2 "ls -l / | grep exo" ou ansible dev -m shell -a "ls -l / | grep exo"

```
drwxr-xr-x   2 root 4096 Mar 11 16:10 exo-ansible
```

On voit bien ci-dessus, le dossier **exo-ansible** avec les droits en **755** (rwxr-wr-x)
et **root** comme propriétaire.

Complétons maintenant notre playbook pour créer un fichier dans le dossier exo-ansible

```
tasks                                 - Le début des tâches à effectuer par le playbook
 - name: test de connexion            - Le nom de la tâche
   ping:                              - le module utilisé
                                      - rien d'autre pour le module ping, il n'y a pas d'arguments


 - name: creation d'un dossier        - Nom de la tâche
   file:                              - Module
    path: /exo-ansible                - Nom du dossier (voir Documentation du module File)
    state: directory                  - Etat de l'action (ici création de dossier)
    owner: root                       - Propriétaire du dossier
    group: root                       - Groupe propriétaire du dossier
    mode: 0755                        - Permissions du dossier

 - name: creation d'un fichier        - Nom de la tâche
   file:                              - Module
     path: /exo-ansible/test.txt      - Chemin et Nom du fichier à créer
     state: touch                     - Etat de l'action (Ici création de fichier)
     mode: 0600                       - Permissions sur le fichier
```

Exécutons de nouveau le playbook et allons vérifier le résultat sur les machines

```
TASK [creation d'un fichier] ***************************************************************************************************************
changed: [172.28.0.2]
changed: [172.28.0.5]

PLAY RECAP *********************************************************************************************************************************
172.28.0.2                 : ok=4    changed=1    unreachable=0    failed=0
172.28.0.5                 : ok=4    changed=1    unreachable=0    failed=0
```

ssh 172.28.0.2 ls -l /exo-ansible ou ansible dev -m shell -a "ls -l /exo-ansible"

```
-rw------- 1 root 0 Mar 11 16:55 test.txt
```

Le fichier est bien créé dans le bon dossier et avec les bons droits .

Pour terminer et voir si vous avez compris, créez un nouveau fichier **playbook3.yml** qui aura comme seule tâche la suppression du fichier test.txt sur la machine dev1 (172.28.0.2 dans mon cas)
Pour cela le module file n'a besoin que de deux arguments :
path pour indiquer le nom du fichier
state avec la valeur **absent** pour indiquer que le fichier doit être absent (donc supprimé s'il existe)

Correction rapide pour ceux qui n'auraient pas tout compris :

Contenu du fichier **playbook3.yml**

```
- name: Playbook3
  hosts: 172.28.0.2
  remote_user: root
  tasks:
  - name: suppression sur dev1 (172.28.0.2)
    file:
      path: /exo-ansible/test.txt
      state: absent
```

Exécution :

```
root@be1867ee17df:/etc/ansible# ansible-playbook playbook3.yml

PLAY [Playbook3] ***************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
ok: [172.28.0.2]

TASK [suppression sur dev1 (172.28.0.2)] ***************************************************************************************************************
changed: [172.28.0.2]

PLAY RECAP *********************************************************************************************************************************************
172.28.0.2                 : ok=2    changed=1    unreachable=0    failed=0
```

Et vérification avec ssh :

```
root@be1867ee17df:/etc/ansible# ssh 172.28.0.2 "ls -l /exo-ansible"
total 0
root@be1867ee17df:/etc/ansible#
```

Plus aucun fichier dans le dossier /exo-ansible de la machine dev1 (172.28.0.2)

Partie 5 : DEPLOIEMENT

Pour terminer , je vous propose les petits challenges suivant :

CHALLENGE 1 :
Créez un playbook (**playbook4.yml)** qui permettra de déployer **apache2** et **php** sur machines de **PROD** de notre Lab. Faites en sorte que les deux serveurs affichent la configuration de php (phpinfo) lorsque l'on se connecte sur leurs URL respectives . **IP:8084 pour prod1, IP:8085 pour prod2**

CHALLENGE 2 :
En complément du challenge1 (qui doit donc avoir été exécuté correctement), créez un playbook (playbook5.yml) qui déploie **symfony** sur les serveurs de prod.
Objectifs . Les URLs 172.16.50.ID:8084/jeu et 172.16.50.ID:8085/jeu doivent afficher la page d'accueil de symfony.

Pour vous aider, vous trouverez une procédure d'installation de symfony, dans la partie 8 de ce PDF: http://195.83.128.21/~pgommery/IMG/pdf/s4-seance00-special_vps.pdf

ATTENTION: Il faut adaptez la procédure à votre configuration (pas de php8 par exemple)
CONSEIL : Utilisez la machine debian-test pour tester l'installation en manuel avant de faire le playbook

N'hésitez pas non plus à consulter la documentation : https://docs.ansible.com/

Partie 6 : CONCLUSION

J'espère que cette toute petite introduction à Ansible vous aura donné envie d'en savoir plus.
Si l'on ajoute les rôles et les plugins que nous n'avons pas vus ici (ainsi que tous les modules disponibles) ,
Imaginez les possibilités qui s'offrent à vous si vous aviez à gérer un parc de centaines de machines
(que ce soient des containers, des VPS ou de vraies machines physiques)

Il existe de très bon tuto sur Internet, par exemple ceux de xavki sur Youtube .
La plupart des vidéos sont gratuites, mais il vous faudra payer 0,99 € par mois pour voir la globalité.
Xavier Pestel touche à tous les domaines du DevOps, vous y trouverez donc des vidéos sur Ansible, Docker ,
Linux mais aussi AWS (Amazon Web Services), Kubernetes, Terraform . Bref une mine d'or à ce tarif-là.
Les Vidéos sont très pédagogiques avec beaucoup d'exemples pratiques.