

# Ansible

- [L'essentiels d'ansible](#)

# L'essentiels d'ansible

## Introduction

Dans cette procédure nous allons apprendre les bases d'ansible.

Nous suivons le cours linkedin suivant : <https://www.linkedin.com/learning/l-essentiel-d-ansible>

De Samir Lakhdari.

## Pratique

### Installation

Dans un premier temps nous allons créer un user ansible sur l'ensemble des serveurs, et le placer en sudo user.

```
cat > /etc/sudoers.d/ansible
```

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

Création de clé SSH pour se connecter sans mot de passe sur l'ensemble des serveurs.

```
ssh-keygen
```

Si nous utilisation la résolution de nom dans le fichier d'inventaire de ansible, les hôtes doivent être déclaré dans le fichiers hosts de la machine /etc/hosts.

```
192.168.14.131 control
```

```
192.168.14.132 web1
```

```
192.168.14.133 web2
```

```
192.168.14.134 haproxy
```

Si FQDN déclarer dans le fichiers hosts alors vous pouvez mettre le FQDN de l'host.

Permet de copier la clé SSH vers un serveur distant

```
ssh-copy-id @IP ou FQDN
```

## Extra packet nécessaire pour ansible

```
yum install -y python3  
yum install -y epel-release
```

## Mise à jour de Python avant l'installation de ansible

```
pip3 install --upgrade pip
```

## Update de Python 3.6 vers 3.8 car déprécié pour ansible et remplacement de l'alias lors de l'utilisation de la commande

```
dnf module enable python38  
dnf install python38  
alias python3=python3.8  
alternatives --set python3 /usr/bin/python3.8
```

## Installation de ansible avec python3

```
pip3 install ansible
```

## Vérification de l'installation

```
ansible --version
```

## Permet de ping la machine local

```
ansible localhost -m ping
```

Permet de tester la connexion au hôte définis dans le fichier d'inventaire de ansible. L'argument ALL à la place de WEB, fait référence à l'ensemble des machines présentes dans les inventaires.

```
ansible web -m ping -i hosts
```

## Créer un fichier vide sur l'ensemble des hosts dans un repertoire spécifique.

```
ansible all -m command -a "touch /root/fic1" -i hosts
```

- all : Envoie la commande sur l'ensemble des hosts
- -m command : Stipule le module, ici une commande
- -a : correspond à l'argument du module, ici la commande ssh pour créer un fichier
- -i hosts : par rapport au fichier d'inventaire "hosts"

Si l'utilisateur de l'hosts n'a pas les droits utilisateurs, alors dans la commande ansible nous pouvons ajouter l'argument -b ou --become pour effectuer une élévation de privilège.

## Gestion des hôtes

Il existe différentes méthodes pour cibler une ou plusieurs machines, soit en appliquant le nom du groupe, le nom de la machine ou en utilisant des syntaxes génériques. Il est également possible d'exclure des machines.

### Ping sur le groupe web et ha

```
ansible web:ha -m ping -i hosts
```

### Ping sur plusieurs machines.

```
ansible web1:web2:haproxy -m ping -i hosts
```

### Ping plusieurs machines avec une syntaxe commune.

```
ansible 'web*' -m ping -i hosts
```

### Exclure la machine web1 avec le symbole "!".

```
ansible 'all:!web1' -m ping -i hosts
```

### Ping uniquement les machines communes à plusieurs groupes.

```
ansible 'web:&ha' -m ping -i hosts
```

### Ping la première machine de la liste du groupe, les groupes étant des tableaux à une seule colonne dans la configuration.

```
ansible 'web[0]' -m ping -i hosts
```

Cela va ping la première machine déclarée dans le groupe web.

### Nous pouvons également utiliser une plage, de la ligne X à la ligne Y.

```
ansible 'web[0:1]' -m ping -i hosts
```

## Utilisation des balises

Une commande ad hoc est un moyen d'exécuter rapidement une tâche Ansible, une tâche que vous n'avez pas besoin d'enregistrer pour la ré-exécuter ultérieurement. Ces commandes sont utiles pour des tests ponctuels et faire des changements rapides. Vous pouvez utiliser une

commande ad hoc pour vous assurer qu'une certaine ligne existe dans un fichier de configuration.

Mais dès le moment où vous avez beaucoup de commandes à exécuter, vous devez les ancrer dans un fichier descriptif qu'on va appeler « le playbook ». Un playbook est un fichier yml, avec en fait un ensemble de plays, donc de jeux d'action.

Notre playbook avec plusieurs petites tâches.

```
---
- name: play1
  hosts: all
  tasks:
    - name: create
      file:
        dest: /root/fic2
        state: touch
      tags:
        - create

- name: play2
  hosts: all:!haproxy
  tags:
    - delete
  tasks:
    - name: delete
      file:
        dest: /root/fic2
        state: absent

- name: play3
  hosts: haproxy
  tasks:
    - name: delete
      file:
        dest: /root/fic2
        state: absent
      tags:
        - delete
```

**Pour l'exécuter, il faut effectuer la commande ci-dessous.**

```
ansible-playbook -i hosts 02_03.yml
```

02\_03.yml étant le nom du fichier playbook.

**Si nous exécutons la même commande alors l'argument supplémentaire tag, nous allons ciblé uniquement les tâches dans le playbook associé avec le tag en question.**

```
ansible-playbook -i hosts 02_03.yml --tags create
```

**A l'inverse, nous pouvons utiliser l'option skip-tags pour outrepasser les tâches avec un tag spécifique.**

```
ansible-playbook -i hosts 02_03.yml --skip-tags create
```

**Nous pouvons également lister les tags des différentes tâches sans que cela n'exécute les tâches avec la commande suivante.**

```
ansible-playbook -i hosts 02_03.yml --list-tags
```

**Executer une tâche sur un hôte local.**

Pour se faire nous allons utiliser un playbook différent :

```
---
- name: play1
  hosts: localhost
  connection: local
  tasks:
    - name: creation via connexion locale
      file:
        dest: /root/Chapitre_02/fic3
        state: touch
```

Dans la section hosts, il est bien spécifié que l'hôte correspond à lui même. Le chemins d'accès doit exister pour que le fichier soit créée.