+adlere
DIGITAL EXPERTISE

Ansible pour professionnels Linux / Unix

Johan Klotz - jklotz@adlere.fr

Le support du cours «Ansible pour professionnels Linux/Unix » est non contractuel ; il ne doit pas être redistribué et/ou reproduit en partie ou en totalité sans permission explicite et écrite de la société Adlere.

Red Hat, le logo Red Hat, OpenShift et Ansible sont des marques déposées ou commerciales de Red Hat, Inc ou ses filiales aux États-Unis et dans d'autre pays. Linux® est une marque déposée de Linus Torvalds aux États-Unis et dans d'autre pays.

UNIX ® est une marque déposée par « The Open Group » aux Etats-Unis et dans d’autres pays.
Wiindows® est une marque déposée de Microsoft Corporation aux États-Unis et dans d'autre pays.

Les autres marques citées sont déposées par leurs propriétaires respectifs.

+adlere
DIGITAL EXPERTISE

Mise en œuvre et utilisation

+sommaire

Architecture ansible

Configuration

```
[defaults]
inventory = ./inventory
host_key_checking = false

[privilege_escalation]
become = true
become_method = sudo
[…]
```

Inventaire

```
[web]
appsrv01.example.org
appsrv02.example.org

[db]
postgre01.example.org
postgre01.example.org
```

Playbook

```
---
- name: Installe et démarre Apache
  hosts: web
  become: yes

  tasks:
    - name: Installation Apache
      ansible.builtin.dnf:
          name: httpd
          state: latest
```

ansible-playbook

nœud de contrôle / management

Systèmes gérés :

Nœud 1

Nœud 2

Nœud 3

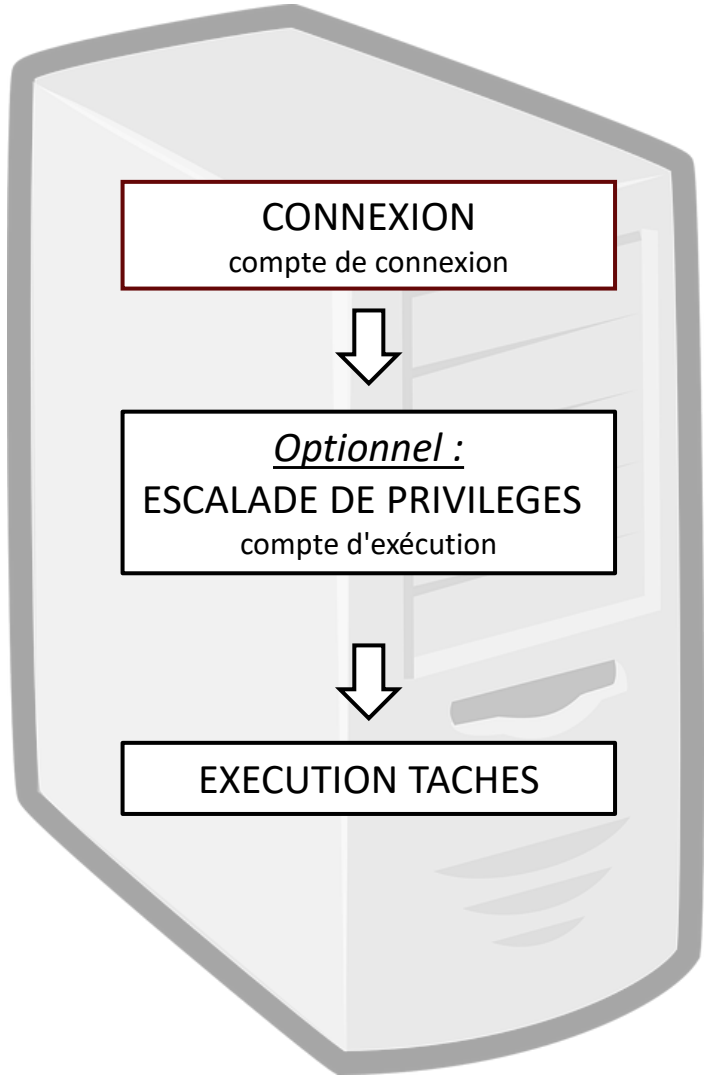
Nœud 4

Nœud 5

Mécanismes de connexion et d'escalade

ansible / ansible-playbook

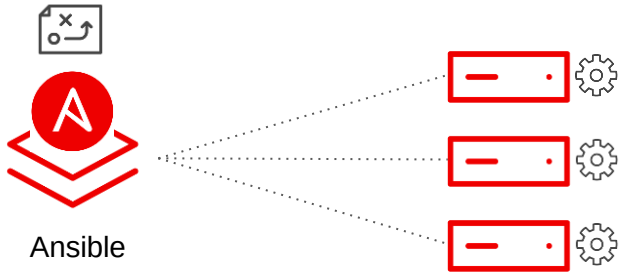
ssh
winrm
paramiko
[…]



système cible

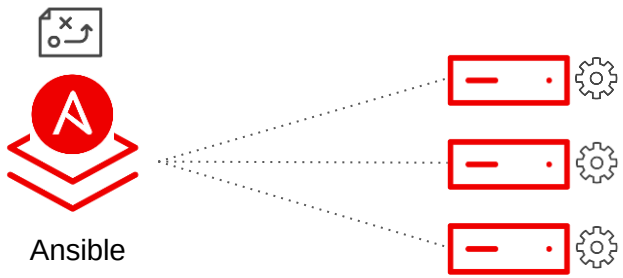
- Compte avec lequel Ansible se connecte sur le système cible. Par défaut, utlilisateur qui lance la commande. On peut lui définit un mot de passe ou une clef ssh.
  Compte par défaut d'exécution des playbooks.
- Changement d'utilisateur d'exécution des playbooks.
  Uniquement si on a un '`become: true`' défini.
  Par défaut : sudo
- Répertoire d'exécution : `~/.ansible/tmp`

Mécanismes de connexion Linux



- Protocole par défaut = ssh (OpenSSH supporté depuis Ansible 0.5, par défaut depuis 1.3)
- Utilise les mécanismes ssh standards (`~/.ssh/config`, port 22, …)
- Par défaut, utilisateur de connexion sur la cible = nom de l'utilisateur qui invoque la commande ansible
- La connexion se fait en utilisant les clefs ssh disponibles dans `~/.ssh` ou en demandant un mot de passe
- de nombreuses options existent pour personnaliser la méthodologie de connexion
  - peuvent se définir dans le fichier d'inventaire, dans le fichier de configuration ansible, dans des variables d'environnement
- Autres options : paramiko, local, winrm
- Plugins disponibles : `ansible-doc -l -t connection`
- Ne jamais lancer une automatisation sans s'assurer que la connexion est fonctionnelle (`ansible -m ping all`)

Mécanismes de connexion Windows



systèmes cibles Windows

- Protocole par défaut = WinRM
- ports 5985 (http) et 5986 (https)
- Configuration à faire côté serveur Windows
  - listener, protocole de connexion, chiffrage, certificats, …
  - https://docs.ansible.com/ansible/latest/os_guide/index.html
  - https://docs.ansible.com/ansible/latest/os_guide/windows_winrm.html
- Modules en PowerShell et non Python
- préfixés par `win_`
- s'assurer que la connexion est fonctionnelle : `ansible -m win_ping all`

- Setting up a Windows Host
  - Host Requirements
  - WinRM Setup
  - Windows SSH Setup
- Using Ansible and Windows
  - Use Cases
  - Path Formatting for Windows
  - Limitations
  - Developing Windows Modules
- Windows Remote Management
  - What is WinRM?
  - WinRM authentication options
  - Non-Administrator Accounts
  - WinRM Encryption
  - Inventory Options
  - IPv6 Addresses
  - HTTPS Certificate Validation
  - TLS 1.2 Support
  - WinRM limitations

Inventaire et fichier de configuration

Inventaire

- souvent un fichier texte avec une liste de cible et des informations de connexion

- On peut ajouter des variables spécifiques à chaque groupe ou chaque hôte.

```
[web]
web-dmz ansible_host=10.42.0.2
appserver01.exemple.org

[db]
postgres01.exemple.org
postgres01.exemple.org

[web:vars]
apache_listen_port=844,
apache_root_path=/www/html/

[all:vars]
ansible_user=automation
ansible_ssh_private_key_file=/home/automation/.ssh/automation_rsa
```

```
[defaults]
inventory = ./inventory
remote_user = user
ask_pass = false
host_key_checking = false

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = true
```

Fichier de configuration

- Fichier texte au format .ini, organisé en blocs
- Définit le fonctionnement et des paramètres par défaut d'Ansible
- Certains paramètres peuvent être redéfinis dans l'inventaire

Playbooks

- fichiers textes au format YAML
- en théorie commence par --- et clos par …
- liste d'éléments, déclarés par un '-'
- chaque élément est un dictionnaire
  - clef: valeur
- on peut mettre des lignes vides pour lisibilité
- Best practice : un clef/valeur ligne
- Il existe une notation courte, déconseillée :

```
- name: Installation de tree
  dnf: name=tree state=latest
  become: true
```

```
---
- name: Installe et démarre Apache
  hosts: web

  tasks:
    - name: Installation Apache
      ansible.builtin.dnf:
        name: httpd
        state: latest
      become: yes

- name: Installe et démarre une une base de données
  hosts: dbservers

  tasks:
    - name: Installation MariaDB
      ansible.builtin.dnf:
        name: mariadb
        state: latest
      become: yes
```

Playbooks

- '#' est le caractère pour les commentaires
- ': ' (2 points + espace) est utilisé pour les mappings
- Literal block scalar : |

```
-  texte: |
     Les longs sanglots
     des violons
     de l'automne.
```

```
Les longs sanglots\ndes violons\nde l'automne.
```

- Folded block scalar : >

```
-  texte: >
     Les longs sanglots
     des violons
     de l'automne.
```

```
Les longs sanglots des violons de l'automne.
```

https://yaml-multiline.info/

Formatage de playbook

Les éléments de même niveau hiérarchique (les modules utilisés par exemple) doivent avoir la même indentation.

Les éléments doivent être plus indentés que ceux dont ils dépendent.

```yaml
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      ansible.builtin.yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      ansible.builtin.template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      ansible.builtin.service:
        name: httpd
        state: started
```

Erreurs de formatage courantes

- Indentation avec de la tabulation au lieu d'espaces
- Indentation sans cohérence (ie des objets de mêmes niveaux n'ont pas le même degré d'indentation)
- Utilisation d'un module qui n'est pas installé ou pas dans un chemin de recherche ('Module not found')
- Mauvaise utilisation / absence d'utilisation des " ou '
- Fautes de frappes dans les mots-clefs

- `ansible-playbook […] --syntax-check`
- `ansible-navigator […] --syntax-check`

```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    -name: httpd package is present
      ansible.builtin.yum:
        name: httpd
        state: latest

    - name :latest index.html file is present
      ansible.builtin.template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      ansible.builtin.service:
          name: httpd
          state: started
          register: var
```

Présentation d'un playbook (1/2)

Un playbook (playbook.yml)

```yaml
---
- name: Installe et démarre Apache
  hosts: web

  tasks:
    - name: Installation Apache
      ansible.builtin.dnf:
        name: httpd
        state: latest
      become: yes

- name: Installe et démarre une une base de données
  hosts: dbservers

  tasks:
    - name: Installation MariaDB
      ansible.builtin.dnf:
        name: mariadb
        state: latest
      become: yes
```

2 plays

`yamllint` ou `ansible-lint` pour vérifier la syntaxe d'un playbook

Playbook = fichier texte ascii au format YAML

Présentation d'un playbook (2/2)

play

```
---
- name: Installe et démarre Apache
  hosts: web

  tasks:
    - name: Installation Apache
      ansible.builtin.dnf:
        name: httpd
        state: latest
      become: yes

    - name: Démarre httpd
      ansible.builtin.service:
        name: httpd
        state: started
```

en-tête de play

tâches ➔ modules

paramètres de module

https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#play

La commande ansible-playbook

- `ansible-playbook -i inventaire [-l sous-ensemble] playbook.yml`

Option de connexion	Objet
`-u utilisateur`	Spécifie le compte de connexion
`-k | --ask-pass`	Affiche le prompt pour demander le mot de passe de connexion
`--private-key XXX`	Spécifie la clef ssh de connexion
Options d'escalade de privilège	
`-b`	
`--become-user XXX`	Spécifie le compte d'escalade de privilèges
`-K | --ask-pass`	Éventuel mot de passe nécessaire pour procéder à l'escalade de privilèges

Codes couleurs lors de l'exécution


```
PLAY [PLAYBOOK DEPLOIEMENT APACHE] *********************************************************************

TASK [Gathering Facts] *********************************************************************************
ok: [lab1node2]

TASK [Installe Apache si RHEL] *************************************************************************
ok: [lab1node2]

TASK [Installe Apache si Debian] ***********************************************************************
skipping: [lab1node2]

TASK [Configure le service web] ************************************************************************
changed: [lab1node2] => {
    "msg": "Configuration effectuée  ... "
}

TASK [Redémarre service] *******************************************************************************
fatal: [lab1node2]: FAILED! => {"changed": false, "msg": "Failed as requested from task"}

PLAY RECAP *********************************************************************************************
lab1node2                  : ok=3    changed=1    unreachable=0    failed=1    skipped=1    rescued=0    ignored=0
```

Tâche exécutée, mais aucun changement
Tâche exécutée, a effectué un changement
Tâche non exécutée (probablement à cause d'une condition)
Une erreur a été générée

- unreachable : impossible de se connecter
- rescued : nombre de blocs 'rescue:' dans lequel un playbook est rentré
- ignored : +1 à chaque fois qu'une tâche échoue mais qu'elle disposait du mot-clef 'ignore_errors: true'
  - mention '... ignoring' en bleu lors de l'exécution

Commandes ad-hoc

- on peut invoquer directement des modules depuis la ligne de commande
- avec la commande `ansible`, pas `ansible-playbook`
- pour des tâches simples en une ligne, ne nécessitant pas un playbook

- Syntaxe générale :

```
ansible [-i inventory] <TARGET> -m <module> [-a "param1=value1 'param2=value2 avec espaces'"]
```

Exemples

`ansible -m ping <TARGET>`	test complet d'une connexion + version de python
`ansible -m setup  <TARGET>`	gather facts `ansible -m setup teacher -a "filter=ansible_distribution"`
`ansible -m command <TARGET> -a "whoami"`	à remplacer par la commande de son choix
`ansible <TARGET> -m command -a /usr/bin/hostname`	
`ansible <TARGET> -m command -a /usr/bin/hostname -o`	sur une seule ligne
`ansible <TARGET> -a /usr/bin/hostname`	`-m command` présumé par défaut

Modules utiles avec les commandes ad-hoc

Catégorie	Module	Commentaires / exemples
Modules de fichiers	`copy`	copie d'un fichier local vers la cible (`ansible -m copy all -a "src=file dest=/tmp/file"`)
	`fetch`	récupère un fichier distant en local `ansible target01 -m fetch -u automation -a "src=/etc/passwd dest=passwd flat=yes"`
	`file`	positionne des permissions et propriétés sur des fichiers
	`lineinfile`	s'assure (ou pas) de la présence d'une ligne en particulier dans un fichier `ansible all -m lineinfile -a "dest=/etc/group regexp='^(users:x:100:)(.*)' line='\1ldapusername,\2' state=present backrefs=yes"`
	`synchronize`	synchronization de contenu avec rsync
Gestion logicielle	`package`	gestion des logiciels en fonction du gestionnaire natif détecté
	`yum / apt / dnf / gem / pip`	Gestionnaires spécifiques à un OS ou format donné.

Modules utiles avec les commandes ad-hoc

Catégorie	Module	Exemples / commentaires
Système	`service` (`systemd` disponible)	`ansible webservers -m service -a "name=httpd state=enabled state=started"`
	`reboot`	redémarrage du système distant
	`user / group`	ajout, effacement, modifications sur des utilisateurs / groupes `ansible all -m user -a "name=newbie uid=4000 state=present"`
	`setup`	informations système `ansible target01 -m setup -u automation -a 'filter=ansible_distribution*'`
	`authorized_key`	Dépôt d'une clef ssh pour un utilisateur donné `ansible TARGET -m authorized_key -a "user=root key='ssh-rsa AAAA ... XXX == root@hostname'"`
Réseau	`get_url`	téléchargement de fichiers par HTTP, HTTPS, ou FTP
	`nmcli`	gestion du réseau
	`uri`	interactions avec les services web
	`firewalld`	gestion du firewall

Modules utiles avec les commandes ad-hoc

Module	Caractéristiques
`command`	Python obligatoire sur la cible; comme c'est lui qui gère la commande, les variables d'environnement et les fonctions de stream (redirections, enchaînements) ne fonctionnent pas.
`shell`	Fait appel à l'environnement utilisateur (donc si problème avec celui-ci, le module peut ne pas fonctionner); enchaînements et redirections disponibles. `/bin/sh` est le shell par défaut.
`raw`	Similaire au module shell, utilise le shell défini pour l'utilisateur; ne nécessite pas python du tout.

Commandes ad-hoc : -t pour le tree callback plugin

- option `-t <répertoire>` fait gérer la sortie d'une commande ad-hoc par le plugin de sortie (callback plugin) tree
- la sortie de chaque host est stockée dans un fichier à son nom, dans le répertoire donné en argument de `-t` :

```
$ ansible -m ping all -t ./output
proxmox | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
docker01 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
[...]
$ ls -R output
output:
aac01  aah01  ansible01  awx01  docker01  eda01  pods01  proxmox  psql01  psql02  ref0  ref1  rsyslog01.intra.ks2i.net
]$ cat ./output/proxmox | jq .
{
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```


Merci