+adlere
DIGITAL EXPERTISE

Ansible pour professionnels Linux / Unix

Johan Klotz - jklotz@adlere.fr

Le support du cours «Ansible pour professionnel 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

installation, fichier de configuration, inventaires

+sommaire

1. Installation
2. Fichier de configuration
3. Inventaires

INSTALLATION

Installation

- Au niveau du système
  - `dnf install ansible-core / apt install ansible-core`

- Au niveau d'un utilisateur
  - disposer de python3
  - créer un virtual env
    - `python3 -m venv <nom du virtual env>`
  - activer le virtual env
    - `. ./<virtual env>/bin/activate`
  - installer ansible-core ou ansible
    - `python3 -m pip install ansible-core` // ou ansible
    - `python3 -m pip install ansible-navigator`
    - […]
    - contrôler : `ansible --version`
  - `deactivate` pour sortir du virtual env

Installation depuis le repository git

- Pré-requis : un environnement Python carré
  - version appropriée de Python (avril 2024 : au moins Python 3.10 pour ansible-core 2.18)
  - Cohérence dans `alternatives --list` (RHEL, pour Debian : `update-alternatives`)
  - module pip
    - RHEL : `python3.11-pip`
    - attention, `pip` est lié à la version de Python

- Ensuite :

```
$ mkdir ansible-git
$ cd ansible-git
$ git clone https://github.com/ansible/ansible.git .
$ source ./hacking/env-setup
$ python3 -m pip install --user -r requirements.txt
$ ansible --version
```

FICHIER DE CONFIGURATION

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 1

Nœud 2

Nœud 3

Nœud 4

Nœud 5

nœud de contrôle / management

Configuration ansible

ansible configuration

- variables d'environnement

  https://docs.ansible.com/ansible/latest/reference_appendices/config.html

  ```
  ansible-config dump | grep -i XXXX
  ```

- mots-clés du playbook, variables de l'inventaire ou variables des playbooks

- `ansible.cfg`

- Remarque sur les variables en ligne de commande (`-e` | `--extra-vars`)
  - la documentation indique : `-e` n'est pas une option de configuration en ligne de commande, c'est une manière de transmettre des variables à un playbook.
  - les variables ont leurs propres règles de priorité

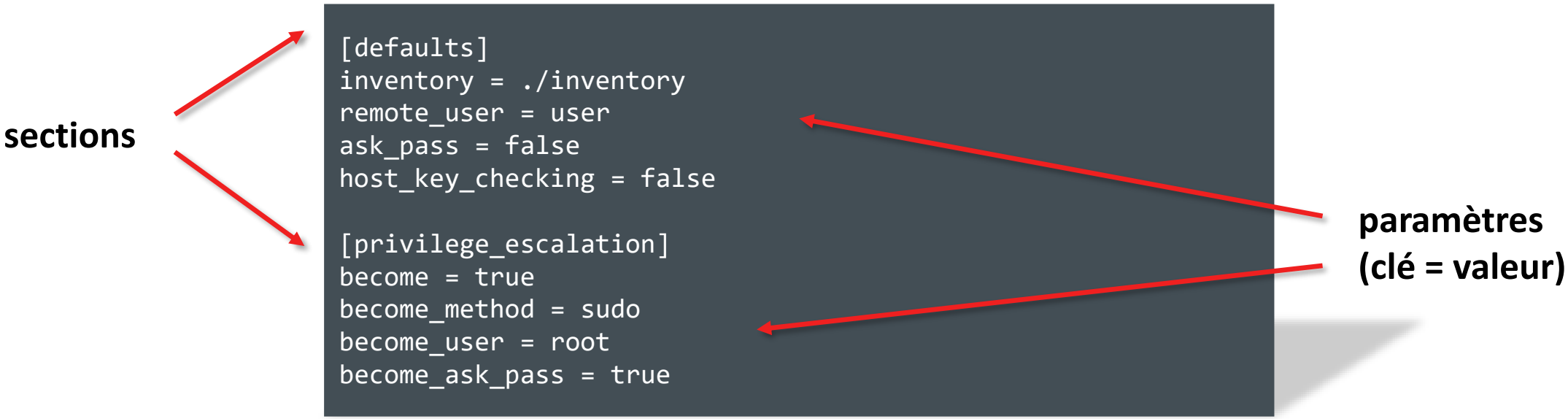
ansible.cfg

Ordre de recherche

- De la priorité la plus élevée à la moins élevée :
  - variable d'environnement `ANSIBLE_CONFIG`
  - `./ansible.cfg` (si le répertoire parent n'est pas accessible en écriture par tout le monde)
  - `~/.ansible.cfg`
  - `/etc/ansible/ansible.cfg`

```
$ ansible --version
ansible [core 2.13.6]
  config file = /home/admjkl/ansible/tmp_project/ansible.cfg
  configured module search path = ['/home/admjkl/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.9/site-packages/ansible
  ansible collection location = /home/admjkl/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.9.13 (main, Nov  9 2022, 13:16:24) [GCC 8.5.0 20210514 (Red Hat 8.5.0-15)]
  jinja version = 3.0.3
  libyaml = True
```

ansible.cfg



- deux sections particulièrement importantes :

  `[defaults]` = valeurs par défaut pour l'opération d'Ansible.

  `[privilege_escalation]` = comment Ansible effectue l'élévation de privilèges sur les hôtes gérés.

Mots-clefs ansible.cfg

Nom	Description	CLI
`inventory`	Spécifie le chemin vers le fichier d'inventaire. Note : il y a la variable d'environnement ANSIBLE_INVENTORY	`-i`
`host_key_checking`	Vérification de la clé publique de l'hôte SSH ; true par défaut (true / false)	
`private_key_file`	Chemin vers la clé privée SSH	`--private-key`
`ask_pass`	Indique s'il faut ou non demander un mot de passe SSH. Peut être "false" si l'authentification par clé publique SSH est utilisée.	`-k`
`deprecation_warnings`	Affiche ou non les informations d'obsolescence. True par défaut, sinon mettre False	

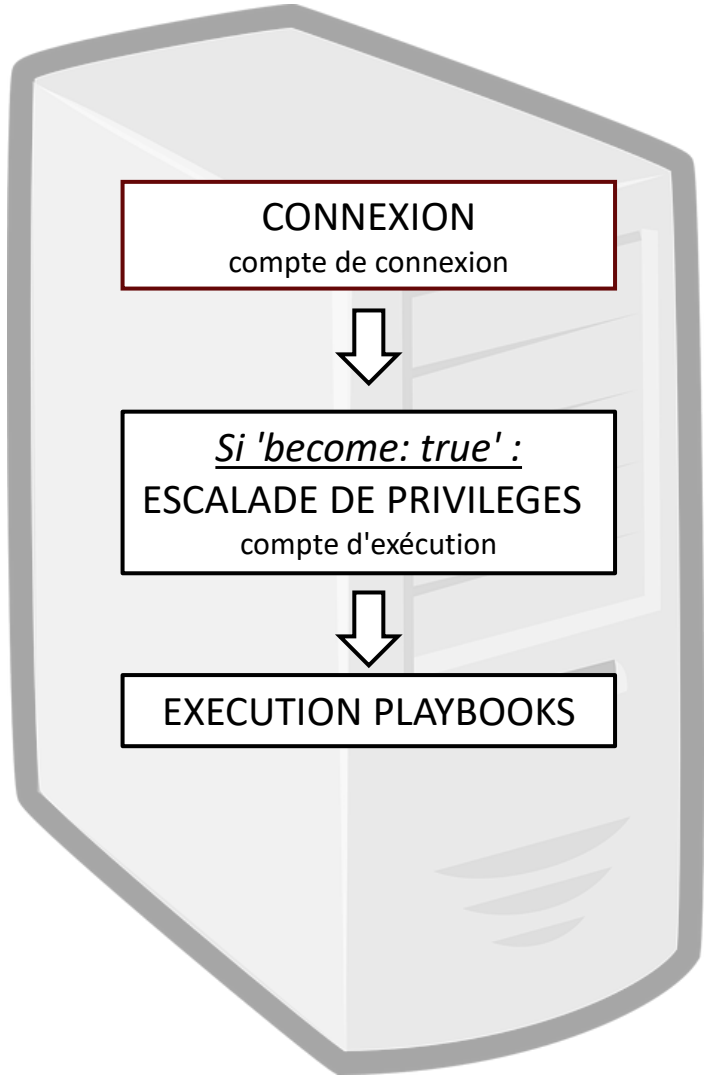
```
[defaults]
inventory = ./inventory.ini
host_key_checking = false
private_key_file = ~/.ssh/id_rsa
ask_pass = true
deprecation_warnings = False
```

Escalade de privilège

Mécanismes par défaut

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.

Utilisateur d'exécution des playbooks
Uniquement si on a un 'become: true' défini.
Par défaut : sudo

Répertoire d'exécution : ~/.ansible/tmp

Mots-clefs ansible.cfg

Nom	Description	CLI
remote_user	Le nom de l'utilisateur pour se connecter aux hôtes gérés. Si non spécifié, le nom de l'utilisateur actuel est utilisé.	-u user
become	Indique s'il faut ou non passer automatiquement à un utilisateur différent sur l'hôte géré (généralement root) après la connexion. Cela peut également être spécifié dans un script (play).	-b, --become
become_method	Comment passer à un autre utilisateur (sudo est la valeur par défaut, mais su est une option).	--become-method
become_user	L'utilisateur vers lequel passer sur l'hôte géré (par défaut = root).	--become-user
become_ask_pass	Indique s'il faut demander un mot de passe pour la méthode become_method ; par défaut = false.	--ask-become-pass, -K

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

Quel fichier de configuration ?

Déterminer le fichier utilisé

- `ansible --version`
- `ansible-config view -vv`

  ansible --version + affiche le contenu du fichier

- `ansible-inventory -host XXX -v`
- `ansible <PATTERN> -v --list-hosts`

```
$ ansible-inventory --host aah01 -v
Using /home/admjkl/ansible/playbooks/01_initial_basic_setup/ansible.cfg as config file
[…]
```

```
$ ansible infra -v --list-hosts
Using /home/admjkl/ansible/tmp_project/ansible.cfg as config file
  hosts (3):
    psql01
    ref0<<<<<<<<<<<<
    nfs01
```

Commande ansible-config | ansible-navigator config

Affiche les éléments de configuration

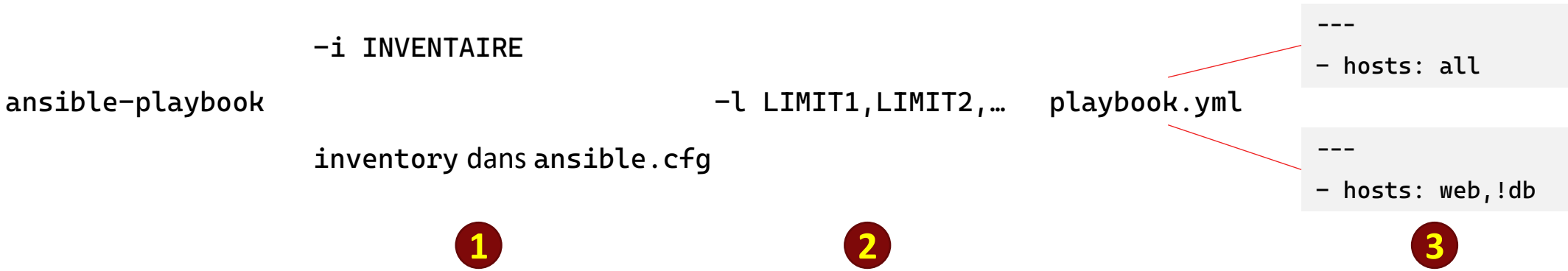
Name	Description
`list`	Liste les paramètres de configuration disponibles. Restriction de catégorie avec -t + mot-clef (all,base,become,cache,callback,cliconf,connection,httpapi,inventory,lookup,netconf,shell,vars) `ansible-navigator config list [-t xxx]`
`dump`	Affiche la ou les valeurs de tous les paramètres Ansible. Variante : `ansible-config dump --only-changed` `ansible-navigator config dump [--only-changed]`
`view`	Affiche le contenu du fichier *ansible.cfg* actuel. `ansible-navigator config view [-v | -vv]`

`init`	Pour générer un fichier sample.cfg :
	`ansible-config init --disabled > ansible.cfg`
	`ansible-navigator config init --disabled > ./ansible.cfg`

INVENTAIRES

Définir la cible d'un playbook

- Une combinaison entre les instructions de la ligne de commande et la directive hosts du playbook est effectuée pour déterminer les cibles d'un playbook



Généralités inventaires (1/2)


```
dbserver01

[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[infra:children]
app1srv
web

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

- fichier texte au format INI ou YAML (= inventaire explicite)
- liste des hôtes qui peuvent être ciblés par Ansible
  - un hôte apparaissant dans plusieurs inventaires ne sera pas traité plusieurs fois
- les hôtes peuvent appartenir ou non à des groupes
  - toujours 2 groupes par défaut : `all` et `ungrouped`
- les hôtes peuvent appartenir à plusieurs groupes
- les groupes peuvent être composés d'autres groupes
- statiques ou dynamiques

Généralités inventaires (2/2)


```
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[app2srv]
appserver1 ansible_host=10.42.0.3
appserver2 ansible_host=10.42.0.4


[web]
node-[1:30]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[infra:children]
app1srv
web

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

Peut contenir des variables :

- par groupe = appelées group_vars
- par hôte = appelées host_vars

/etc/ansible/hosts = inventaire par défaut, ne pas utiliser

Généralement remplacé / défini :

- par la variable ANSIBLE_INVENTORY
- **ou** par l'option **-i** des différentes commandes Ansible
- ou **`./ansible.cfg`**
- ou depuis **`~/.ansible.cfg`**

Inventaires: format INI ou yaml

Les deux format sont valides

- TOML, JSON également pris en charge

```
mister-brown ansible_host=10.0.101.100
blue.example.com
192.168.100.1

[web]
alpha.example.org
beta.example.org ansible_host=192.168.200.122
192.168.1.10
192.168.1.15
srv[001:006]

[web:vars]
http_port=80
https_port=443

[db]
db01.intranet.local
db02.intranet.local
db-[99:101].example.com
```

Les noms des nœuds ne correspondent pas nécessairement aux noms réels du réseau

Les plages sont prises en charge.

Obligatoire en YAML.

```
---
ungrouped:
  hosts:
    mister-brown:
      ansible_host: 10.0.101.100
    blue.example.com:
    192.168.100.1:

web:
  hosts:
    alpha.example.org:
    beta.example.org:
      ansible_host: 192.168.200.122
    192.168.1.10:
    192.168.1.15:
    srv[001:006]:
  vars:
    http_port: 80
    https_port: 443

db:
  hosts:
    db01.intranet.local:
    db02.intranet.local:
db-[99:101].example.com:
```

Inventaire

Inventaires statiques : répertoires group_vars et host_vars

```
$ tree
.
├── ansible.cfg
├── group_vars
│   ├── aap2db
│   │   ├── file1.yml
│   │   └── file2.yml
│   ├── all
│   └── dev.yml
├── host_vars
│   ├── nfs01.yml
│   └── psql01
│       ├── file1.yml
│       └── file2
├── inventory
└── playbook.yml
```

- Inventaire implicite : structure de répertoire relative au répertoire du projet
- `group_vars` et `host_vars` = répertoires de recherche pour variables :
  - de groupe : group_vars
  - de systèmes : host_vars
- peuvent contenir des variables:
  - dans des fichiers ayant le même nom que le groupe ou l'hôte concerné
    - `/project/group_vars/asia.yml` => variables du groupe asia
  - dans des fichiers de nom quelconque, rangés dans un sous-répertoire de même nom que le groupe ou hôte concerné
    - `/project/host_vars/db01.exemple.org/db_vars` => variables spécifiques au bsystème db01.exemple.org

Inventaire

où définir les variables d'inventaire ?

- dans l'inventaire, dans une section `[all:vars]`
  - s'applique à chaque groupe/hôte
- dans l'inventaire, au niveau de l'hôte
- dans l'inventaire, dans une section `[<nom_groupe>:vars]`

- dans une structure de répertoires
  - le répertoire group_vars : contient les variables spécifiques à chaque groupe
  - le répertoire `host_vars` contient les variables spécifiques à chaque hôte

```
$ tree
.
├── ansible.cfg
├── group_vars
│   ├── aap2db
│   └── all
├── host_vars
│   ├── nfs01.yml
│   └── psql01
│       ├── file1.yml
│       └── file2
├── inventory
└── playbook.yml
```

- Les variables sont stockées au format "`clé: valeur`"
  - pas comme "variable = valeur" dans le fichier d'inventaire.

```
$ cat ./host_vars/psql01/file1.yml
---
variable: valeur
```

Arborescence par environnements


```
.
├── ansible.cfg
├── environments/          # Répertoire parent pour les différents inventaires
│   ├── 000_cross_env_vars
│   ├── dev/               # Fichiers de l'environnement dev
│   │   ├── group_vars/    # group_vars de dev
│   │   │   ├── all
│   │   │   │   ├── 000_cross_env_vars → ../../../000_cross_env_vars
│   │   │   │   └── env_specific
│   │   │   ├── db
│   │   │   └── web
│   │   └── hosts          # Uniquement les systèmes de l'environnement dev
│   │
│   ├── prod/              # Fichiers de l'environnement prod
│   │   ├── group_vars/    # group_vars de prod
│   │   │   ├── all
│   │   │   │   ├── 000_cross_env_vars → ../../../000_cross_env_vars
│   │   │   │   └── env_specific
│   │   │   ├── db
│   │   │   └── web
│   │   └── hosts          # Uniquement les systems de l'environnement prod
│   │
│   └── stage/             # Fichiers de l'environnement stage
│       ├── group_vars/    # group_vars de stage
│       │   ├── all
│       │   │   ├── 000_cross_env_vars → ../../../000_cross_env_vars
│       │   │   └── env_specific
│       │   ├── db
│       │   └── web
│       └── hosts          # Uniquement les systems de l'environnement stage
│
└── playbook.yml
```

- Plus complexe mais valide : un système d'arborescence par environnement (dev, prod, stage), et des liens symboliques vers un fichier de variables communes, à basse précédence.

- Inventaire par défaut :

```
[defaults]
inventory = ./environments/dev
```

Inventaires

ansible-inventory

ansible-inventory --graph
ansible-inventory --graph --vars
ansible-inventory --host XXX
ansible-inventory --list [--yaml | toml]
(toml si le package python toml est installé)
ansible-navigator inventory my_inventory --list -m stdout

```
$ ansible-inventory --graph
@all:
  |--@aap2:
  |  |--aac01
  |  |--aah01
  |  |--psql01
  |--@ctl:
  |  |--aac01
  |--@hub:
  |  |--aah01
  |--@infra:
  |  |--nfs01
  |  |--psql01
  |  |--ref0
  |--@test:
  |  |--localhost
  |  |--mini01
  |  |--testrepo2
  |--@ungrouped:
```

ansible-playbook --list-hosts

Liste des systèmes concernés par un playbook :

```
$ ansible-playbook selinux.yml --list-hosts

playbook: selinux.yml

  play #1 (all): Touch a file and sets permissions      TAGS: []
    pattern: ['all']
    hosts (13):
      aac01
      ref1
      rsyslog01.intra.ks2i.net
      eda01
      awx01
```

```
$ ansible infra -v --list-hosts
Using /home/admjkl/ansible/tmp_project/ansible.cfg as config
file
  hosts (3):
    psql01
    ref0
    nfs01
```

Variables de connexion utiles

Le nom peut différer par rapport au nom utilisé dans le fichier de configuration

Exemples de variables de connexion disponibles

Variable de l'inventaire	Objectif
`ansible_host=xx.xx.xx.xx`	Adresse réseau / IP de l'hôte
`ansible_password = PASSWORD`	Mot de passe à utiliser (`ansible_ssh_pass` fonctionne également)
`ansible_port = 222`	Port de connexion SSH
`ansible_user = ansible`	Nom d'utilisateur pour la connexion avant Ansible 2.0, c'était `ansible_ssh_user`
`ansible_ssh_private_key_file=~/ansible/keys`	Fichier de clé privée à utiliser
`ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q SYSADM@IP_REBOND" -o ForwardAgent=yes -o StrictHostKeyChecking=no '`	Arguments additionnels à la ligne de commande ssh
`ansible_ssh_extra_args='-o StrictHostKeyChecking=no'`	Arguments ajoutés à la ligne de commande SSH
`connection=local`	Généralement utilisé pour travailler avec 'localhost', pour contourner la pile SSH/réseau.

➔ ne pas confondre ansible_host et ansible_hostname

Variables de connexion utiles

Escalade de privilèges

Variables décrivant l'escalade de privilèges dans l'inventaire

Variable de l'inventaire	Objectif
`ansible_become`	Indique s'il faut procéder à une escalade de privilèges yes/no, true/false
`ansible_become_method`	Outil d'escalade de privilèges (su ou sudo, sudo par défaut)
`ansible_become_user`	le nom de l'utilisateur avec lequel on travaille une fois l'escalade de privilèges effectuée
`ansible_become_password`	le mot de passe associé à cet utilisateur
`ansible_become_exe`	le binaire d'escalade de privilèges (spécifique à la méthode sélectionnée)
`ansible_become_flags`	les options à passer à ce binaire

Host patterns

Informations supplémentaires

- La cible d'une automatisation peut être désignée de plusieurs façons
  - sur la ligne de commande ansible-playbook avec l'option -l
    - `ansible-playbook playbook.yml -l dev`
  - sur la ligne de commande ansible
    - `ansible dev -m XXXX`
- on appelle 'host pattern' ce motif de désignation
  - chaîne de caractère qui désigne des groupes et/ou systèmes définis dans l'inventaire

- Cette cible est filtrée / combinée avec la directive hosts du playbook
  - il peut en résulter une situation où aucune cible valide n'est déterminée
  - exemple : '`-l prod`' mais directive '`hosts: dev`' dans le playbook
  - `ansible-playbook --list-hosts` permet de voir qels hôtes sont utilisés par le playbook
  - un playbook peut enchaîner des plays avec des cibles hosts différentes

```
---
- hosts: all
```

Host patterns

Adresser des hôtes : exemples

- La liste des hôtes réellement adressée est définie par
  - le contenu de l'inventaire
  - et les directives de ligne de commande
  - et les directive de play des playbooks

```
---
- name: Debug module sample
  hosts: web
  gather_facts: yes

  tasks:
  […]
```

- Légère différence de syntaxe entre `ansible-playbook` | `ansible-navigator` et `ansible`

ansible	ansible-playbook
ansible all --list-hosts ansible psql01 -m ping ansible all psql01 -m ping ansible all -i 10.0.4.70, -m ping	ansible-playbook debug.yml ansible-playbook -l psql01,ref0 debug.yml ansible-playbook -l 'all:!groupe2' debug.yml ansible-playbook -l '192*' debug.yml ansible-playbook -l 'groupe1:&groupe2' debug.yml ansible-playbook -i 10.0.4.70, debug.yml ansible-playbook -i localhost, --connection=local site.yml

Bonus sur les inventaires

- On peut fournir plusieurs inventaires à la fois

  `ansible -m ping all -i  hors-prod -i prod`

- On peut les regrouper dans un répertoire (`ansible -i ./inventory_folder -m ping all`)

```
inventory_folder/
  proxmox.yml             # appel au plugin Proxmox
  dynamic-inventory.py    # script d'inventaire dynamique
  static-inventory        # inventaire statique (ini | yaml)
  group_vars/
    all.yml
```

- On peut mettre des numéros (eg `01-proxmox.yml`, `02-static-inventory`, ...) pour forcer un ordre particulier, ensuite :

  `ansible-playbook.yml -i /chemin/vers/inventory`

INVENTAIRES DYNAMIQUES À PARTIR D'UN PLUGIN

Inventaires

Informations supplémentaires

- L'option -i peut référencer un plugin
  - `ansible-inventory --graph  -i $(/bin/cat /etc/hosts| /bin/cut -f1 -d' ' |/bin/grep -P '^[a-z]'|/bin/xargs|/bin/sed 's# #,#g')`
  - `ansible-config dump | grep -i inventory`
  - `ansible-doc -t inventory <PLUGIN_NAME>`

- On peut configurer des plugins (sources) d'inventaires dans ansible.cfg
- Cf par défaut : `ansible-config dump | grep -i inventory_enabled`

```
[inventory]
enable_plugins = ini, advanced_host_list, host_list, virtualbox, yaml, constructed, nmap, community.general.proxmox
```

Plugin d'inventaire

https://www.redhat.com/sysadmin/ansible-plugin-inventory-files

- Obligatoirement un programme Python respectant une structure précise
- Placé dans un emplacement connu

  `ansible-config dump|grep DEFAULT_INVENTORY_PLUGIN_PATH`
- Désigné dans un fichier .yml, donné en entrée des commandes ansible

  `ansible -m ping -i nmap_inventory_plugin.yaml all`
- Activé dans `ansible.cfg`

```
[inventory]
enable_plugins = ini, advanced_host_list, host_list, virtualbox, yaml, constructed, nmap
```

désignation plugin

paramètres plugin

```
---
plugin: nmap
address: 192.168.1.0/24
strict: False
ipv4: yes
ports: no
sudo: true
groups:
  appliance: "'Amazon' in hostname"
  regular: "'host' in hostname"
```

nmap_inventory.yaml

- liste des plugins d'inventaire disponibles :

  `ansible-doc -l -t inventory`

  Documentation : `ansible-doc -t inventory <NOM-PLUGIN>`
- plugin nmap (community.general; nmap doit être installé) :
  - `ansible-inventory -i nmap_inventory.yaml --list`
- host_list plugin :
  - `ansible -i $(/bin/cat /etc/hosts| /bin/cut -f1 -d' '|/bin/grep -P '^[a-z1]'|/bin/xargs|/bin/sed 's# #,#g') -m ping all`

INVENTAIRES DYNAMIQUES À PARTIR D'UN SCRIPT

Inventaire dynamiques à partir d'un script

- RH recommande plutôt de faire des plugins d'inventaire
- le script ar convention accepte les arguments `--list` et `--host <hostname>`
- **`--list`**
  - le script doit produire en sortie des données encodées en JSON contenant tous les groupes et les hôtes à gérer.
- **`--host <hostname>`**
  - le script doit renvoyer un hachage ou un dictionnaire au format JSON contenant les variables de l'hôte (peut être vide) pour le nom d'hôte spécifié.
- Développement de sources d'inventaire dynamiques :
  https://docs.ansible.com/ansible/latest/dev_guide/developing_inventory.html

https://www.redhat.com/sysadmin/ansible-dynamic-inventories
https://linux.goffinet.org/ansible/comprendre-inventaire-ansible/

Exemple :

```
$ ansible-inventory --inventory
scripts/nmap_inventory.py --list
{
    "_meta": {
        "hostvars": {
            "dmaf5.home": {
                "ip": [
                    "192.168.1.26",
                    "192.168.1.25"
                ]
            },
            "macmini2": {
                "ip": [
                    "192.168.1.16"
                ]
            },
            "raspberrypi": {
                "ip": [
                    "192.168.1.11"
                ]
            }
        }
    },
    "all": {
        "children": [
            "ungrouped"
        ]
    },
    "ungrouped": {
        "hosts": [
            "dmaf5.home",
            "macmini2",
            "raspberrypi"
        ]
    }
}
```


Merci