



Ansible pour professionnels

Linux / Unix

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

variables

+sommaire

Variables

- variables utilisateur
- Ansible Facts
- variables personnalisées (custom facts)
- variables magiques
- chiffrement des variables avec ansible-vault





- Variable = emplacement pour une valeur qui est susceptible de changer
- Que met-on en variable ? ➔ **Tout ce qu'il est possible, autant que possible**
- Noms de variables :
 - uniquement des lettres, des chiffres et des traits de soulignement (underscore)
 - doivent commencer par une lettre ou un trait de soulignement.
- Des noms sont réservés (mots-clés Python ou mots-clés de playbooks)
- Portée des variables :
 - **globale** : les valeurs sont définies pour tous les hôtes (configuration Ansible, variables d'environnement, ligne de commande)
 - **play** : variables définies en début de play
 - **hôte/groupe** (variables définies par hôte ou groupe dans le fichier d'inventaire)
 - **tâche** (valeurs définies pour tous les hôtes dans le contexte d'une tâche, par exemple dans la section vars d'une tâche)

```
---  
- name: Démo variables  
  hosts: localhost  
  connection: local  
  vars:  
    var_1: world  
    var_2: Hello  
    var_3: "{{ var_2 }}" "{{ var_1 }}"  
  
  tasks:  
    - name: Affiche var_3  
      ansible.builtin.debug:  
        msg: "{{ var_3 }}"
```



ansible-playbook mon-playbook.yml

Exemple de playbook de base et différentes possibilités d'afficher des variables avec le module `ansible.builtin.debug` :

```
---
- name: Mon playbook
  hosts: localhost
  connection: local
  gather_facts: no
  vars:
    http_port: 80
    http_root: '/var/www/html'

  tasks:
    - name: Affiche une variable
      ansible.builtin.debug:
        msg: "{{ http_root }}"
```

```
---
- name: Mon playbook
  hosts: localhost
  connection: local
  gather_facts: no
  vars:
    http_port: 80
    http_root: '/var/www/html'

  tasks:
    - name: Affiche une variable
      ansible.builtin.debug:
        msg:
          - Fichiers = {{ http_root }}
          - "Port : {{ http_port }}"
```

```
---
- name: Mon playbook
  hosts: localhost
  connection: local
  gather_facts: no
  vars:
    http_port: 80
    http_root: '/var/www/html'

  tasks:
    - name: Affiche une variable
      ansible.builtin.debug:
        var: http_root, http_port
```

Les variables sont entourées de `{{ }}` quand elles sont référencées.

On utilise des `'''` quand `{{` en début de ligne, ou que la ligne contient `'`

(si une valeur après un `:` commence par `{`, YAML pensera que l'on déclare un dictionnaire)

Certains mots-clefs ne prennent pas de `{{ }}` (`var`, `when`)

Attention à l'auto-complétion dans VS Code / l'éditeur

var: séparer les variables par des `,` (non documenté)



```
ansible-playbook -vv mon-playbook.yml
```

```
---
- name: Mon playbook
  hosts: localhost
  connection: local
  gather_facts: no
  vars:
    http_port: 80
    http_root: '/var/www/html'

  tasks:
    - name: Affiche une variable
      ansible.builtin.debug:
        msg: "{{ http_root }}"
        verbosity: 2
```

- 'verbosity' permet un affichage conditionnel de la tâche, en fonction du nombre degré de verbosité avec lequel le playbook a été invoqué
- -vv = verbosity: 2
- -vvv = verbosity: 3
- 'debug' est exécuté sur le nœud de contrôle (ie celui d'où la commande ansible est lancée)



Type de donnée	Données/Structure	Affichage / référencement
Variables booléennes	<pre>vars: vrai: True faux: false</pre>	La casse n'a pas d'importance; la documentation se focalise sur true/false (par cohérence avec ansible-lint), mais les valeurs suivantes sont valides : "Truthy values" : True , true , 't' , 'yes' , 'y' , 'on' , '1' , 1 , 1.0 "Falsy values" : False , false , f , no , n , off , 0 , 0.0
Variables de type nombre	<pre>vars: num1: 5 num2: 2.55</pre>	entier ou flottants
Chaînes	<pre>vars: texte1: 'hello world' texte2: "\u4f60\u597d" =</pre>	Type 'AnsibleUnicode' Défini entre simple quote, sauf si on veut afficher des caractères UniCode
Liste / tableaux	<pre>vars: tableau1: [1,'2',5.0] tableau2: - 1 - '2' - 5.0</pre>	On référence les éléments d'un tableau avec un index - <code>ansible.builtin.debug</code> <code>msg:</code> - <code>"{{ tableau2[1] }}"</code> - <code>"{{ tableau2 first }}"</code> - <code>"{{ tableau2 last }}"</code>
Dictionnaire	<pre>vars: mydic: clef: valeur</pre>	Association de clefs et de valeurs

Déterminer le type d'une variable :

```
- ansible.built.in.debug:
  var: myvar | type_debug
```




Où définir des variables (1/2)

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

- En dehors d'un playbook

Emplacement	Exemple
Dans le fichier d'inventaire, par hôte ou par groupe	<pre>[web] node1 ansible_host=3.66.235.245 node2 ansible_host=18.159.111.141 [web:vars] http_port=8080</pre>
Dans les fichiers de variables d'hôtes et de groupes	<pre>group_vars/web host_vars/node1 host_vars/node2</pre>
Dans des fichiers de variables personnalisés, fournis en ligne de commande	<pre>ansible-playbook playbook.yml -e@2.yml -e@3.yml</pre>
Fournies directement en ligne de commande (-e ou --extra-vars)	<pre>ansible-playbook test.yml -e myvar1="hello" -e myvar2=123 ansible-playbook test.yml -e "myvar1=hello myvar2=123"</pre>

Fichier de variables

```
---
myvar: hello
semaine: {
  "lundi":1,
  "mardi":2}
weekend:
  - samedi
  - dimanche
...
```



Où définir des variables (2/2)

10

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

- Dans un playbook

Emplacement	Exemple	
Section 'vars' d'un playbook	<pre>--- - name: Playbook vars section hosts: localhost connection: local vars: vrm: 12.7.1 myvar: 'value'</pre>	La notation en liste reste valide mais cela pourrait changer : <pre>vars: - vrm: 12.7.1 - myvar: 'value'</pre>
Dans des fichiers de variables personnalisés, appelés depuis un playbook	<pre>--- - name: Playbook vars section hosts: localhost connection: local vars_files: ./vars/variables.yml</pre>	Sans précision de chemin de recherche, le fichier désigné sera spontanément aussi recherché dans le répertoire 'vars' du projet.
Placées dans un fichier et chargées dans un playbook par le module <code>include_vars</code>	<pre>- ansible.builtin.include_vars: file: vars.yml</pre>	<pre>- name: Include a variables file ansible.builtin.include_vars: variables_file.yml</pre>
Définies à la volée par l'utilisateur	<pre>- ansible.builtin.set_fact: my_var: "Hello" - ansible.builtin.debug: msg: "{{ my_var }}"</pre>	
En enregistrant la sortie d'un module	<pre>- ansible.builtin.command: uptime register: output - ansible.builtin.debug: var: output.stdout</pre>	



Type de donnée	Données/Structure	Affichage / référencement
Liste / tableau	<pre>vars: tableau1: [1,'2',5.0] tableau2: - 1 - '2' - 5.0</pre>	<pre>- ansible.builtin.debug msg: - "{{ tableau2[1] }}" - "{{ tableau2 first }}" - "{{ tableau2 last }}"</pre>
Dictionnaire	<pre>vars: system_settings: maxfree: 95 minfree: 20 blk_size: 4096</pre>	<p>Les dictionnaires sont des collections d'ensembles de données clé : valeur. Modifiables, indexés (par les clefs) et non ordonnés.</p> <pre>- ansible.builtin.debug: msg: - "{{ system_settings['blk_size'] }}" - "{{ system_settings.blk_size }}"</pre>
Liste de dictionnaires	<pre>vars: user_list: - name: john uid: 1510 - name: bob uid: 1511 vars: user_list: [{ 'name': 'john', 'uid': 1510 }, { 'name': 'bob', 'uid': 1511 }]</pre>	<pre>tasks: - name: Affichage de toutes les clefs des éléments ansible.builtin.debug: msg: "{{ item.name }}, {{ item.uid }}" loop: "{{ user_list }}"</pre>
Dictionnaire sur plusieurs niveaux	<pre>vars: bloc_cidr: production: cidr_prod: "172.31.0.0/16" developpement: cidr_dev: "10.0.0.0/24"</pre>	<pre>tasks: - name: Print CIDR block ansible.builtin.debug: var: bloc_cidr['production']['cidr_prod']</pre>



- 'set_fact' pour définir une variable pendant l'exécution du playbook.

```
---
- name: Démo set_fact
  hosts: localhost
  gather_facts: no

  tasks:
    - name: On définit des variables
      ansible.builtin.set_fact:
        var_str: 'hello world'
        var_int: 5
        var_dict: {'key_1': value1, 'key2': value2}
        var_list: [1,2,3,'quatre']

    - name: On affiche les variables
      ansible.builtin.debug:
        msg:
          - "{{ var_str }}"
          - "{{ var_int }}"
          - "{{ var_dict }}"
          - "{{ var_list }}"
```



Définition d'une variable avec 'register'

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

- La directive 'register:' stocke les informations résultantes d'un module dans une variable.
- La variable est de type dictionnaire, les clés disponibles dépendent du module appelé.
- En général, il y a au moins une clé '.changed'.
- Le module "command" a les clés '.stdout', '.stderr' et '.stdout_lines' disponibles.

```
---
- name: Playbook vars section
  hosts: localhost
  connection: local
  become: yes

  tasks:
    - name: Install dig
      ansible.builtin.dnf:
        name: bind-utils
        state: present
        register: output

    - ansible.builtin.debug:
        msg: "{{ output | type_debug }}"

    - ansible.builtin.debug:
        msg: "{{ output }}"
```




Capturer le résultat d'une commande

14

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

```
---
- hosts: localhost
  connection: local
  gather_facts: no

  tasks:
    - ansible.builtin.command:
      cmd: uptime
      register: output

    - ansible.builtin.debug:
      var: output
```

Afficher `output['stdout']` pour avoir juste la sortie de la commande.

Noter :

- xxx.failed (true / false)
- xxx.changed (true / false)
- rc (return code), spécifique à 'command:'
- stderr_lines ou stdout_lines

```
PLAY [localhost]
*****
TASK [ansible.builtin.command]
*****
changed: [localhost]

TASK [ansible.builtin.debug]
*****
ok: [localhost] => {
  "output": {
    "changed": true,
    "cmd": [
      "uptime"
    ],
    "delta": "0:00:00.022944",
    "end": "2024-01-15 08:13:17.500079",
    "failed": false,
    "msg": "",
    "rc": 0,
    "start": "2024-01-15 08:13:17.477135",
    "stderr": "",
    "stderr_lines": [],
    "stdout": " 08:13:17 up 119 days, 28 min,  1 user,  load average: 0.27, 0.08, 0.02",
    "stdout_lines": [
      " 08:13:17 up 119 days, 28 min,  1 user,  load average: 0.27, 0.08, 0.02"
    ]
  }
}
```



Priorité (de la plus élevée à la plus basse) :

1. Variables de la ligne de commande (`-e` | `--extra-vars`)
2. Variables définies au niveau d'une tâche
3. Variables définies au niveau d'un bloc
4. Variables de rôles `[role]/vars/main.yml` et d'une tâche `include_vars`
5. Variables définies par `set_fact`
6. Variables définies par `register`
7. Variables de play : `vars_files`, `vars_prompt`, `vars`
8. Facts de l'hôte
9. `host_vars` de playbook
10. `group_vars` de playbook
11. `host_vars`, `group_vars`, `vars` de l'inventaire
12. Variables par défaut de rôle (`roles/ ... /defaults/main.yml`)

```
---  
- hosts: all  
  remote_user: root  
  vars:  
    port: 443  
  roles:  
    - role: apache_install
```

```
roles:  
  - role: apache_install  
    vars:  
      port: 8443
```

appel de rôle avec redéfinition de variable

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_variables.html#understanding-variable-precedence



- attribuer des noms descriptifs et clairs aux variables, réfléchir au nommage.
- les inventaires devraient contenir un minimum possible de variables
 - problème de visibilité pour les mainteneurs de playbooks
 - ils ne devraient contenir que des variables vraiment spécifiques à un hôte ou un groupe
 - pas dans un fichier d'inventaire mais `group_vars` ou `host_vars`
 - valeurs par défaut des variables courantes communes → `group_vars/all`
 - variables spécifiques à un ensemble de machines → `group_vars/<nom_groupe>`
- Éviter `set_fact`, privilégier `vars_files`, ou l'inventaire en dernier recours
- Éviter les extra-vars (`-e`) autant que possible
 - pour des tests locaux ou quand maintenabilité / idempotence ne sont pas une préoccupation première
- Les rôles doivent fournir des valeurs par défaut les plus génériques possibles (dernier recours au cas où elles ne sont pas redéfinies par ailleurs)





- une variable non définie génère une erreur d'exécution
 - "The task includes an option with an undefined variable. "
- il est recommandé de fixer une valeur par défaut aux variables
 - | default('VALEUR')
- on peut spécifier qu'une variable est optionnelle
 - | default(omit)

```
---
- hosts: all
  gather_facts: false
  vars:
    user: "psmith"

  tasks:
    - ansible.builtin.user:
      name: "{{ user }}"
      comment: "{{ gecos | default('Human account') }}"
```

```
- name: Création de fichiers, mode optionnel
  ansible.builtin.file:
    dest: "{{ item.path }}"
    state: touch
    mode: "{{ item.mode | default(omit) }}"
  loop:
    - path: /tmp/foo
    - path: /tmp/bar
    - path: /tmp/baz
    mode: "0444"
```

- variables découvertes automatiquement par Ansible sur les hôtes gérés
 - tâche '[Gathering Facts]'
- contiennent des informations spécifiques à l'hôte
- découvertes par défaut
- désactiver :
 - `ansible.cfg: gathering = explicit` dans [defaults]
 - directive 'gather_facts: no' / activation avec 'gather_facts: yes'
- visualisation rapide d'une cible : `ansible -m setup target`

```
---  
- name: Test host pattern #1  
  hosts: localhost  
  connection: local  
  gather_facts: yes
```

```
{  
  "ansible_all_ipv4_addresses": [  
    "REDACTED IP ADDRESS"  
  ],  
  "ansible_all_ipv6_addresses": [  
    "REDACTED IPV6 ADDRESS"  
  ],  
  "ansible_apparmor": {  
    "status": "disabled"  
  },  
  "ansible_architecture": "x86_64",  
  "ansible_bios_date": "11/28/2013",  
  "ansible_bios_version": "4.1.5",  
  "ansible_cmdline": {  
    "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-862.14.4.el7.x86_64",  
    "console": "ttyS0,115200",  
    "no_timer_check": true,  
    "nofb": true,  
    "nomodeset": true,  
    "ro": true,  
    "root": "LABEL=cloudimg-rootfs",  
    "vga": "normal"  
  },  
  "ansible_date_time": {  
    "date": "2018-10-25",  
    "day": "25",  
    "epoch": "1540469324",  
    "hour": "12",
```




- `gather_facts = no` et collecte explicitement limitée à un sous-ensemble

```
---
- name: facts playbook
  hosts: localhost
  connection: local
  gather_facts: no

  tasks:
    - name: Collect a subset of facts
      ansible.builtin.setup:
        gather_subset:
          - all_ipv4_addresses
          - "!all"
          - "!min"
```

all, all_ipv4_addresses,
all_ipv6_addresses, apparmor,
architecture, caps,
chroot, cmdline, date_time,
default_ipv4, default_ipv6,
devices, distribution,
distribution_major_version,
distribution_release,
distribution_version, dns,
effective_group_ids,
effective_user_id, env, facter,
fips, hardware, interfaces,
is_chroot, iscsi, kernel, local,
lsb, machine, machine_id, mounts,
network, ohai, os_family, pkg_mgr,
platform, processor,
processor_cores, processor_count,
python, python_version,
real_user_id, selinux,
service_mgr,
ssh_host_key_dsa_public,
ssh_host_key_ecdsa_public,
ssh_host_key_ed25519_public,
ssh_host_key_rsa_public,
ssh_host_pub_keys, ssh_pub_keys,
system, system_capabilities,
system_capabilities_enforced,
user, user_dir, user_gecos,
user_gid, user_id, user_shell,
user_uid, virtual,
virtualization_role,
virtualization_type

`ansible -m setup all -a "filter=ansible_distribution"`

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html



Fact	Variable
Nom de la distribution Linux Alpine, Altlinux, Amazon, Archlinux, ClearLinux, Coreos, CentOS, Debian, Fedora, Gentoo, Mandriva, NA, OpenWrt, OracleLinux, RedHat, Slackware, SLES, SMGL, SUSE, Ubuntu, VmwareESX	<code>ansible_facts['distribution']</code>
Type d'OS AIX, Alpine, Altlinux, Archlinux, Darwin, Debian, FreeBSD, Gentoo, HP-UX, Mandrake, RedHat, SGML, Slackware, Solaris, Suse, Windows	<code>ansible_facts['os_family']</code>
Version majeure du système d'exploitation, par exemple 8 pour Red Hat Linux 8.6	<code>ansible_facts['distribution_major_version']</code>
Identifiant complet de version, par exemple "9.0", "8.8", ...	<code>ansible_facts['distribution_version']</code>
Identifiant unique de l'instance OS, généré à l'installation. <code>/etc/machine-id</code> ou <code>"Machine ID:"</code> de <code>hostnamectl</code>	<code>ansible_facts['machine_id']</code>

Attention : le nom d'utilisation n'est pas le nom d'affichage du module 'setup' : remplacer 'ansible_XXX' par `ansible_facts['XXX']`



Fact

Variable

Nom d'hôte court

`ansible_facts['hostname']`

Nom de domaine complet (Fully Qualified Domain Name, FQDN)

`ansible_facts['fqdn']`

Adresse IPv4 principale (basée sur le routage)

`ansible_facts['default_ipv4']['address']`

Mémoire du système

`ansible_facts['memtotal_mb']`

Nombre de processeurs

`ansible_facts['processor_count']`

Liste des noms de toutes les interfaces réseau

`ansible_facts['interfaces']`

Taille de la partition de disque /dev/vda1

`ansible_facts['devices']['vda']['partitions']['vda1']['size']`

Liste des serveurs DNS

`ansible_facts['dns']['nameservers']`

Version du noyau en cours d'exécution

`ansible_facts['kernel']`



```
---
- hosts: localhost
  gather_facts: true
  connection: local
  check_mode: yes

  tasks:
    - ansible.builtin.debug:
      var: ansible_facts['date_time']

    - ansible.builtin.debug:
      var: ansible_facts['date_time']['date']
```

```
ok: [localhost] => {
  "ansible_facts['date_time']": {
    "date": "2024-12-30",
    "day": "30",
    "epoch": "1735553367",
    "epoch_int": "1735553367",
    "hour": "11",
    "iso8601": "2024-12-30T10:09:27Z",
    "iso8601_basic": "20241230T110927194419",
    "iso8601_basic_short": "20241230T110927",
    "iso8601_micro": "2024-12-30T10:09:27.194419Z",
    "minute": "09",
    "month": "12",
    "second": "27",
    "time": "11:09:27",
    "tz": "CET",
    "tz_dst": "CEST",
    "tz_offset": "+0100",
    "weekday": "Monday",
    "weekday_number": "1",
    "weeknumber": "53",
    "year": "2024"
  }
}
```



- Lorsqu'une valeur de variable est un dictionnaire, il existe deux syntaxes pour récupérer la valeur :

<code>ansible_facts['default_ipv4']['address']</code>	<code>ansible_facts.default_ipv4.address</code>
<code>ansible_facts['dns']['nameservers']</code>	<code>ansible_facts.dns.nameservers</code>

- la syntaxe `['...']` est la méthode recommandée.
- avant Ansible 2.5, les faits étaient injectés en tant que variables individuelles préfixées par la chaîne `ansible_` au lieu de faire partie de la variable `ansible_facts`.
 - par exemple, `ansible_distribution` est devenu `ansible_facts['distribution']`
 - on peut désactiver l'ancien système de nommage en définissant le paramètre `inject_facts_as_vars = false` dans la section `[defaults]` de `ansible.cfg`
 - valeur par défaut = `true`



Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

Ancienne forme

`ansible_hostname``ansible_fqdn``ansible_default_ipv4['address']``ansible_interfaces``ansible_devices['vda']['partitions']['vda1']['size']``ansible_dns['nameservers']``ansible_kernel`

Forme ansible_facts

`ansible_facts['hostname']``ansible_facts['fqdn']``ansible_facts['default_ipv4']['address']``ansible_facts['interfaces']``ansible_facts['devices']['vda']['partitions']['vda1']['size']``ansible_facts['dns']['nameservers']``ansible_facts['kernel']`



- Il existe un certain nombre de modules ansible pour collecter des facts
- D'abord rechercher avant de se lancer dans un appel à 'shell' plus ou moins complexe

```
---  
- name: Collecte de facts  
  hosts: all  
  gather_facts: no  
  
  tasks:  
    - ansible.builtin.package_facts:  
  
    - name: Print the package facts  
      ansible.builtin.debug:  
        var: item['version']  
        loop: "{{ ansible_facts['packages']['python3'] }}"
```

- `ansible-doc -l | grep fact`

```
ansible.builtin.gather_facts  
ansible.builtin.package_facts  
ansible.builtin.service_facts  
ansible.builtin.setup  
ansible.posix.rhel_facts  
ansible.windows.setup  
community.general.listen_ports_facts  
community.general.snmp_facts  
community.general.usb_facts  
community.general.zfs_facts  
community.general.zpool_facts  
community.windows.win_disk_facts  
community.windows.win_listen_ports_  
community.windows.win_product_facts  
containers.podman.podman_container_info  
redhat.rhel_system_roles.firewall_lib_facts  
redhat.rhel_system_roles.podman_container_info  
redhat.rhel_system_roles.selinux_modules_facts  
[ ... ]
```



- Injection de données customisée dans les facts Ansible en créant des fichiers `.fact` dans `/etc/ansible/facts.d`
- format `.ini` ou `.json`, ou exécutables produisant une sortie JSON
- stockés par le module `setup` dans la clef `ansible_facts['ansible_local']`
- accessible en ad-hoc avec `ansible SYSTEM -m setup -a "filter=ansible_local"`
- organisés en fonction du nom du fichier qui les a définis

```
[packages]
web_package = httpd
db_package = mariadb-server

[users]
user1 = joe
user2 = jane
```

`/etc/ansible/facts.d/custom.fact`



```
"ansible_local": {
  "custom": {
    "packages": {
      "db_package": "mariadb-server",
      "web_package": "httpd"
    },
    "users": {
      "user1": "joe",
      "user2": "jane"
    }
  }
}
```

Sortie de `'ansible -m setup SYSTEM -a"filter=ansible_local"'`

Dans un playbook :

```
- name: Display
  debug:
    var: ansible_facts['ansible_local']
```



- Variables automatiquement positionnées par ansible
 - **hostvars**
 - variables des systèmes gérés, accessibles depuis tout autre système cible du playbook
 - proviennent du **gather_fact** et des inventaires `host_vars` et `group_vars`
 - exemple: `hostvars['nom_systeme']['variable']`
 - **group_names**
 - les groupes auquel le système appartient
 - **groups**
 - tous les groupes de l'inventaire et leurs systèmes
 - **inventory_hostname | inventory_hostname_short**
 - le nom sous lequel le système est connu dans l'inventaire
pas nécessairement le nom réseau remonté par `ansible_facts['hostname']`
 - **play_hosts**
 - tous les hôtes sur lesquels le play en cours sera executé
 - **ansible_version**

```
- name: Unreachable hosts
  ansible.builtin.debug:
    var: ansible_play_hosts_all|difference(ansible_play_hosts)
```

```
"ansible_version": {
  "full": "2.16.6",
  "major": 2,
  "minor": 16,
  "revision": 6,
  "string": "2.16.6"
}
```

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

[Généralités](#)[Variables utilisateur](#)[Ansible Facts](#)[Custom Facts](#)[Magic Vars](#)[ansible-vault](#)

- `ansible_play_hosts`
 - list des hôtes encore actifs du play (ie les machines de l'inventaire qui sont encore joignables à ce stade)
- `ansible_playbook_python`
 - path to the python executable used to invoke the Ansible command line tool.
- `inventory_dir`
 - pathname of the directory holding Ansible's inventory host file
- `inventory_file`
 - pathname and the filename pointing to the Ansible's inventory host file.
- `playbook_dir`
 - contains the playbook base directory.
- `role_path`
 - returns the current role's pathname (since 1.8). This will only work inside a role.
- `ansible_check_mode`
 - a boolean magic variable which will be set to True if you run Ansible with `--check`.

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

- fonctionnalité pour chiffrer des données sensibles (AES256, clef symétrique, AES128 pour les versions plus anciennes)
- les données chiffrées peuvent être contenues dans le playbook ou dans un fichier externe
- un fichier chiffré peut être désigné sur la ligne de commande ou dans le playbook
- commande 'ansible-vault' pour mettre en œuvre la fonctionnalité

ansible-vault <option>

create	crée un nouveau fichier chiffré
decrypt	déchiffre un fichier
edit	édition d'un fichier chiffré
view	visualiser le contenu du fichier chiffré
encrypt	chiffre un fichier
rekey	change le chiffrement d'un fichier (ré-écriture complète)
encrypt_string	chiffrement d'une chaîne

--ask-vault-pass, --vault-password-file --vault-id



Chiffrer un fichier entier

30

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

1

```
Lorem ipsum dolor sit amet,  
consectetur adipiscing  
elit. Nullam commodo, nisl  
sit amet fringilla  
dignissim, augue dui  
ultrices ipsum, et dapibus  
massa lorem quis nulla.  
Nunc ac mi eget elit  
commodo interdum et eget  
neque.
```

4

```
$ ansible-vault decrypt  
lorem.txt  
Vault password:  
Decryption successful
```

2

```
$ ansible-vault encrypt lorem.txt  
New Vault password:  
Confirm New Vault password:  
Encryption successful
```

3

```
$ANSIBLE_VAULT;1.1;AES256  
3762346639393365333263616636306533653  
6646136636361343931646465393066313139  
6263363134393030386264623338613766383  
438303662613066350a663466396635313766  
3737333537626361643562373532323265303  
6636366313834666237356265616436396238  
663036623732646138  
[...]
```



Exemple #1 : variable dans un playbook (1/2)

31

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

- On commence par générer une version chiffrée de la valeur à protéger

```
$ ansible-vault encrypt_string 'My_Passw0rd' --name 'db_passwd'
```

```
New Vault password: <password>
```

```
Confirm New Vault password: <password>
```

```
Encryption successful
```

```
db_passwd: !vault |
```

```
    $ANSIBLE_VAULT;1.1;AES256
```

```
    33653831326366656138613535366634623434616461363363616165633862643934346230663539  
    3132653166333863656334663336613665643666386366620a663233343537656439383433323334  
    61393764343165353264646566363638623330616333313139363635336531396232613563346131  
    6462303030333237370a363763666530636161363661316462353066373334373632353462366631  
    6238
```

- copier le résultat précédent à partir du nom de la variable dans un playbook
- inclure toute la sortie depuis le nom de la variable (ici db_passwd) jusqu'à la fin de la donnée chiffrée



Exemple #1 : variable dans un playbook (2/2)

32

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

```
---
- name: Test variable vault
  hosts: localhost
  connection: local
  gather_facts: no
  vars:
    db_passwd: !vault |
      $ANSIBLE_VAULT;1.1;AES256
      33653831326366656138613535366634623434616461363363616165633862643934346230663539
      3132653166333863656334663336613665643666386366620a663233343537656439383433323334
      61393764343165353264646566363638623330616333313139363635336531396232613563346131
      6462303030333237370a363763666530636161363661316462353066373334373632353462366631
      6238

  tasks:
    - ansible.builtin.debug:
        msg: "{{ db_passwd }}"
```

```
$ ansible-playbook --ask-vault-password ./vault_demo.yml
```

```
Vault password:
```

```
PLAY [Vault usage demo] *****
TASK [Display secret value2] *****
ok: [localhost] => {
    "msg": "My_Passw0rd"
}
```

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



Example #2: variable dans un fichier dédié

33

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

1. fichier `datas.yml` :
2. chiffrer avec : `$ ansible-vault encrypt datas.yml`
3. `datas.yml` devient un fichier chiffré

```
---  
db_passwd: 'passw0rd'
```

```
---  
- hosts: localhost  
  connection: local  
  gather_facts: no  
  
  tasks:  
    - name: Affiche donnée chiffrée  
      ansible.builtin.debug:  
        var: db_passwd
```

```
---  
- hosts: localhost  
  connection: local  
  gather_facts: no  
  vars_files:  
    - datas.yml  
  
  tasks:  
    - name: Affiche donnée chiffrée  
      ansible.builtin.debug:  
        var: db_passwd
```

```
---  
- hosts: localhost  
  connection: local  
  gather_facts: no  
  
  tasks:  
    - name: Charge un fichier de données  
      ansible.builtin.include_vars: datas.yml  
  
    - name: Affiche donnée chiffrée  
      ansible.builtin.debug:  
        var: db_passwd
```

```
ansible-playbook vault.yml \  
--ask-vault-password -e@datas.yml
```

```
ansible-playbook -l localhost vault.yml --ask-vault-password
```



Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

- Afficher le prompt sur la ligne de commande :

```
ansible-playbook [...] --ask-vault-pass ou --ask-vault-password  
ou encore : export ANSIBLE_ASK_VAULT_PASS=False|True
```

- En mettant la clef dans un fichier :

```
ansible-playbook [...] --vault-password-file=./mot_de_passe.txt
```

ajouter quand même le fichier dans un éventuel .gitignore

une variable d'environnement ANSIBLE_VAULT_PASSWORD_FILE disponible





Prévenir l'apparition d'un secret dans un log

35

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

- **no_log: true** pour que le secret ne soit ni loggué, ni affiché sur la console en cas de plantage d'une tâche
- s'applique à toutes les informations d'une tâche

```
- name: Avec no_log
  ansible.builtin.debug:
    msg: "Le mot de passe est {{password }}"
  no_log: true
```

```
PLAY [Exemple no_log] *****

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

TASK [Sans no_log] *****
ok: [localhost] => {
  "msg": "Le mot de passe est Passw0rd=456"
}

TASK [Avec no_log] *****
ok: [localhost]
```

```
ansible-playbook no_log.yml
```

```
PLAY [Exemple no_log]
*****

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

TASK [Sans no_log]
*****
ok: [localhost] => {
  "msg": "Le mot de passe est Passw0rd=456"
}

TASK [Avec no_log]
*****
ok: [localhost] => {"censored": "the output has been hidden
due to the fact that 'no_log: true' was specified for this
result"}
```

```
ansible-playbook no_log.yml -v
```


- avant Ansible 2.4, une clef par playbook
- chaque fichier/variable devait utiliser la même clef
- vault IDs : identifiant de mot de passe ou fichier de mot de passe spécifique
 - doivent être définis dans ansible.cfg

```
[defaults]  
vault_identity_list = dev@./password_file1 , prod@./password_file2
```

```
ansible-vault encrypt --encrypt-vault-id dev file_to_encrypt
```

```
ansible-vault encrypt_string 'secret_data' --name 'db_passwd' --encrypt-vault-id dev
```

- peuvent être renseignés à la volée au déchiffrement

```
ansible-playbook playbook.yml --vault-id dev@prompt --vault-id prod@./password_file2
```

- dans le même fichier, on peut avoir différentes variables chiffrées avec des clefs différentes



Variables chiffrées et vault-ids dans un playbook

37

Généralités

Variables utilisateur

Ansible Facts

Custom Facts

Magic Vars

ansible-vault

```
--
- name: Vault usage demo
  hosts: all
  gather_facts: false
  vars:
    - db_passwd: !vault |
      $ANSIBLE_VAULT;1.2;AES256;dev
      37303463336139323362623634303737316534326665353037326237656432316164633636356234
      6335633166386662633162303936613661643766343762330a316437336430303934623665303164
      62333961353930656165656638313536383561363135343338323937666664613339313266353032
      6430613966623963320a326636313663336332303837376339356236323232373335323633306562
      61623163623133643634383035633231613366623837656130343936616430323537303332636636
      6433336562656230373863653232616562306338323634346265
    - prod_password: !vault |
      $ANSIBLE_VAULT;1.2;AES256;prod
      66353934323061316130363130366663303130633338343132393466303433306161613234306330
      3533396331646164396630663239623062386666643430370a666531323162626464306165396166
      31633235656265636164326232613365613830363665643234386164646261356639313363366264
      3561333330613231310a303336313935366564303736376536643439356535323438633266316261
      3636

  tasks:
    - name: Display secret value
      debug:
        msg: "{{ db_passwd }}"
      when: db_passwd is defined

    - name: Display secret value 2
      debug:
        msg: "{{ prod_password }}"
      when: prod_password is defined
```



Merci





Variable	Description
awx_job_id	l'ID du job en cours
awx_job_launch_type	Comment le job a été lancé : manual, relaunch, host callback, scheduled, dependency, workflow, sync (synchro de projet), scm (synchro inventaire SCM)
awx_job_template_id	ID du template dont le job est instancié
awx_job_template_name	Nom du template dont le job est instancié
awx_execution_node	nom du noeud d'exécution qui a lancé le job
awx_project_revision	ID du commit du projet dans le SCM (aka scm_revision dans les propriétés du job)
awx_project_scm_branch	branche du projet dans le SCM
awx_job_scm_branch	branche du playbook dans le SCM
awx_user_email	Pas disponible en cas de lancement par callback ou schedule.
awx_user_first_name	Pas disponible en cas de lancement par callback ou schedule.
awx_user_id	Pas disponible en cas de lancement par callback ou schedule.
awx_user_last_name	Pas disponible en cas de lancement par callback ou schedule.
awx_user_name	Pas disponible en cas de lancement par callback ou schedule.
awx_schedule_id	si applicable, l'ID du schedule qui a lancé le job
awx_schedule_name	
awx_workflow_job_id	
awx_workflow_job_name	
awx_inventory_id	
awx_inventory_name	