



# **Ansible pour professionnels**

## **Linux / Unix**

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

**variables**



# +sommaire

## Variables

- déclaration et utilisation
- Ansible Facts
- variables personnalisées (custom facts)
- variables magiques





## GÉNÉRALITÉS



# Playbook de base

Mode local uniquement

mon-playbook.yml

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

  tasks:
  - name: Affiche un message
    ansible.builtin.debug:
      msg: "Message"
```

- Pour une exécution / développement en local, tester des tâches ou des syntaxes ou les opérations sur des variables
- Se mettre dans un répertoire de travail dédié, enrichir inventaire et `ansible.cfg` au fur et à mesure des besoins

**ansible-playbook mon-playbook.yml**



Différentes possibilités d'afficher des variables avec le module `ansible.builtin.debug` :

`ansible-playbook 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 }}"
```

```
---
- 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 `:`  
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 `,`



```
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
- Malgré ce que l'affichage peut laisser croire, 'debug' est exécuté sur le nœud de contrôle (ie celui d'où la commande ansible est lancée)



```
---
- 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 }}"
```

- 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)

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  <pre>ansible.builtin.debug   msg:     - tableau2[1]     - tableau2   first     - tableau2   last</pre>

Déterminer le type d'une variable :

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



# Types de variables

Comment accéder à certaines variables

11

Type de donnée	Données/Structure	Affichage / référencement
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>{{ system_settings['blk_size'] }}</pre> <pre>{{ 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: Message     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_cidr: "10.0.0.0/24"</pre>	<pre>tasks:   - name: Print CIDR block     ansible.builtin.debug:       var: bloc_cidr['production']['cidr_prod']</pre>



# Où définir des variables (1/2)

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)

## 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>	
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>	





# Variables ansible

## Types de variables / en action

indenté sur

ansible.builtin.debug

14

```
---
- hosts: csr1
  gather_facts: false
  hosts: localhost
  connection: local
  vars_files:
    - "./vars.yml"
    - "vars2.yml"
  vars:
    var_int: 4
    var_str: 'a string text'
    var_list1: [1, 3, 5, 7, 9, 11]
    var_list:
      - city1
      - city3
      - city5
      - city7
      - city9
    var_dic1:
      name: John
      family: DOE
      age: 43

  tasks:
    - name: include vars
      ansible.builtin.include_vars: var3.yml
```

implicite : dans ./vars

int  
strings  
lists  
dictionnaires

charge le fichier de variables

```
- name: Affiche des variables
  ansible.builtin.debug:
    msg:
      - "var_int : {{ var_int }}"
      - "var_str: {{ var_str }}"
      - "var_list1: {{ var_list1 }}"
      - "var_list2[1]: {{ var_list2[1] }}"
      - "var_dic1: {{ var_dic1 }}"
      - "var_dic1.name: {{ var_dic1.name }}"
      - "var_dic1.name: {{ var_dic1['name'] }}"
```

- Itération sur une variable de type liste :

```
- name: print vars
  ansible.builtin.debug:
    msg: "{{ item }}"
  loop:
    - "{{ var_list }}"
```



Définition d'une variable avec 'set\_fact:'

- 'set\_fact' pour définir une variable pendant l'exécution du playbook.

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

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

    - name: Displays the variables
      ansible.builtin.debug:
        msg:
          - "{{ var_str }}"
          - "{{ var_int }}"
          - "{{ var_dict }}"
          - "{{ var_list }}"
```



## Définition d'une variable avec 'register:'

- 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 }}"
```



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](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.
- Valeurs par défaut des variables courantes communes → `group_vars/all`
- Variables spécifiques à un ensemble de machines → `group_vars`
- Définir les variables de groupe et d'hôte dans les répertoires `group_vars` et `host_vars`
  - de préférence **PAS** dans le fichier d'inventaire.
- Si des rôles sont utilisés, définir les variables de rôle par défaut dans `roles/your_role/defaults/main.yml`
- Lors de l'appel de rôles, transmettre les variables que l'on souhaite remplacer en tant que paramètres,  
pour faciliter la lecture des tâches.

```
roles:  
- role: apache_install  
  vars:  
    http_port: 8080
```





- seules des variables vraiment spécifiques à un hôte ou un groupe devraient être définies dans l'inventaire
- les inventaires devraient contenir un minimum possible de variables
  - problème de visibilité pour les mainteneurs de playbooks
- 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 bien quand maintenabilité ou 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)





## ANSIBLE FACTS

- 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](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  <a href="https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html">https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html</a>	<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>



Fact	Variable
Nom d'hôte court	<code>ansible_facts['hostname']</code>
Nom de domaine complet (Fully Qualified Domain Name, FQDN)	<code>ansible_facts['fqdn']</code>
Adresse IPv4 principale (basée sur le routage)	<code>ansible_facts['default_ipv4']['address']</code>
Mémoire du système	<code>ansible_facts['memtotal_mb']</code>
Nombre de processeurs	<code>ansible_facts['processor_count']</code>
Liste des noms de toutes les interfaces réseau	<code>ansible_facts['interfaces']</code>
Taille de la partition de disque /dev/vda1	<code>ansible_facts['devices']['vda']['partitions']['vda1']['size']</code>
Liste des serveurs DNS	<code>ansible_facts['dns']['nameservers']</code>
Version du noyau en cours d'exécution	<code>ansible_facts['kernel']</code>



- 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`



Ancienne forme	Forme ansible_facts
<code>ansible_hostname</code>	<code>ansible_facts['hostname']</code>
<code>ansible_fqdn</code>	<code>ansible_facts['fqdn']</code>
<code>ansible_default_ipv4['address']</code>	<code>ansible_facts['default_ipv4']['address']</code>
<code>ansible_interfaces</code>	<code>ansible_facts['interfaces']</code>
<code>ansible_devices['vda']['partitions']['vda1']['size']</code>	<code>ansible_facts['devices']['vda']['partitions']['vda1']['size']</code>
<code>ansible_dns['nameservers']</code>	<code>ansible_facts['dns']['nameservers']</code>
<code>ansible_kernel</code>	<code>ansible_facts['kernel']</code>

- 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
[ ... ]
```



VARIABLES PERSONNALISÉES ("CUSTOM 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 variable `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 MAGIQUES



- Variables automatiquement positionnées par ansible
  - **hostvars**
    - variables des systems gérés, accessibles depuis tout autre système cible du playbook
    - proviennent des inventaires `host_vars` et `group_vars`
    - exemple: `hostvars['nom_système']['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 exécuté

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

[https://docs.ansible.com/ansible/latest/reference\\_appendices/special\\_variables.html](https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html)





**Merci**

