Ansible

Marc Baudoin

Hybrix

IGPDE — 8 – 9 décembre 2025

Sommaire

Ici, c'est du fait maison, sans aucun artifice d'intelligence

Figure – https://www.uqac.ca/ressourcespedago/iag/

Figure – https://ai-label.org/

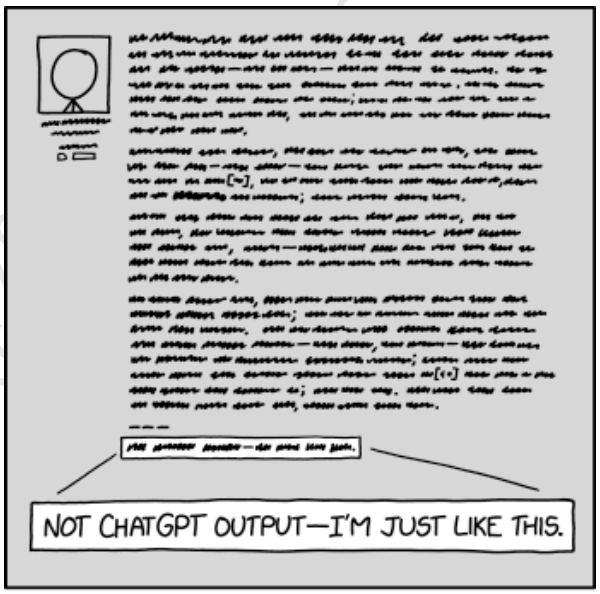
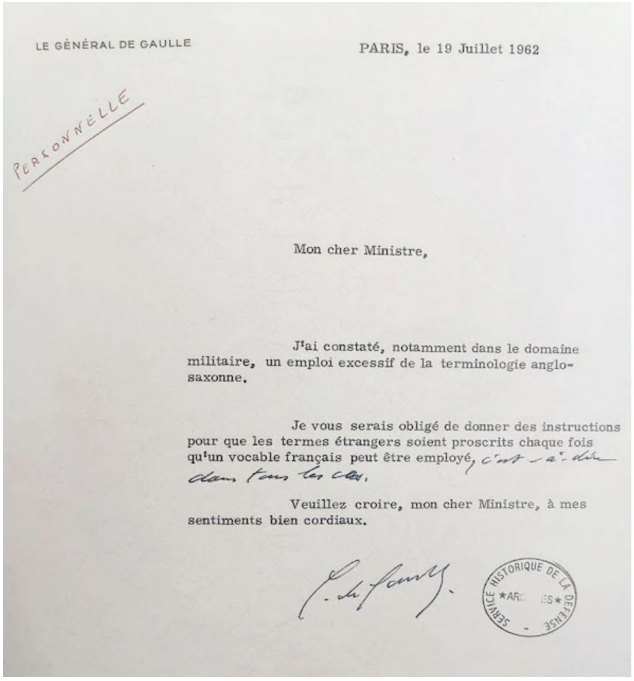


Figure – https://xkcd.com/3126/

Ici, on s'exprime en français (pas en franglais)

« La langue de la République est le français. »

https://fr.wikipedia.org/wiki/Article_2_de_la_Constitution_de_la_Cinquième_République_française



LE GÉNÉRAL DE GAULLE

PARIS, le 19 Juillet 1962

PERSONNELLE

Mon cher Ministre,

J'ai constaté, notamment dans le domaine militaire, un emploi excessif de la terminologie anglo-saxonne.

Je vous serais obligé de donner des instructions pour que les termes étrangers soient proscrits chaque fois qu'un vocable français peut être employé, dans tous les cas.

Veuillez croire, mon cher Ministre, à mes sentiments bien cordiaux.

SERVICE HISTORIQUE DE LA DÉFENSE

Figure – https://www.fix-dessinateur.com/

Sommaire

IaC

- Signifie : *infrastructure as code*.
- L'IaC consiste à gérer un système informatique (l'*infrastructure*) au moyen de fichiers de définition (le *code*) plutôt que grâce à de nombreuses opérations manuelles — idéalement regroupées dans des scripts afin d'atteindre un certain niveau d'automatisation — effectuées par des administrateurs.

`https://fr.wikipedia.org/wiki/Infrastructure_as_code`
`https://infrastructure-as-code.com/`

Gestion de configuration

Problématique

- Dans des temps pas si anciens, un gros centre informatique se composait au plus de quelques dizaines de serveurs, souvent hétérogènes.
- Aujourd'hui, un gros centre informatique se compose de plusieurs centaines de serveurs virtualisés, souvent organisés en groupes de serveurs identiques pour des raisons de redondance.
- Comment assurer la gestion de ces serveurs, de manière cohérente, simple et la plus automatisée possible ?

Gestion de configuration

De quoi s'agit-il ?

- La gestion de configuration a pour objectifs :
  - de décrire la configuration d'un système informatique ;
  - de l'appliquer de manière automatique.
- La configuration du système informatique est contenue dans un ensemble de fichiers appelé CMDB (*configuration management database*).
- Outre son utilité pour la gestion de configuration, la CMDB a également d'indéniables qualités documentaires.
- En corollaire, la gestion de configuration permet également :
  - de conserver l'historique des configurations successives (en mettant la CMDB sous contrôle de versions) ;
  - de détecter et de corriger les configurations non conformes.
- Attention : il est illusoire d'espérer utiliser de la gestion de configuration sans savoir au préalable réaliser manuellement les mêmes opérations.

https://fr.wikipedia.org/wiki/Gestion_de_configuration
https://fr.wikipedia.org/wiki/Configuration_management_database
https://cfgmgmtcamp.org/

Gestion de configuration

Idempotence

- Le concept d'*idempotence* est fondamental en gestion de configuration.
- En mathématiques, une application f est dite *idempotente* si et seulement si :

$$f \circ f = f \qquad \text{(le symbole } \circ \text{ se dit « rond », il s'agit de la composition)}$$

c'est-à-dire si et seulement si :

$$\forall x \; f(f(x)) = f(x)$$

- Une opération est donc idempotente si elle a le même effet lorsqu'on l'effectue une fois ou deux fois de suite et ainsi, par récurrence, plusieurs fois de suite.

https://fr.wikipedia.org/wiki/Composition_de_fonctions
https://fr.wikipedia.org/wiki/Idempotence

Gestion de configuration

Quels logiciels peut-on utiliser ?

Ansible
- pas d'agent sur les machines à gérer (mais il y a des prérequis)
- la communication avec les machines à gérer se fait au moyen de leur protocole natif (SSH, WinRM, REST…)
- écrit en Python

CFEngine
- modèle agent-serveur
- écrit en C
- utilise un langage de description complexe

Chef
- modèle agent-serveur
- écrit en Ruby

Puppet
- modèle agent-serveur (sauf Bolt)
- écrit en Ruby

Salt
- modèle agent-serveur
- écrit en Python

https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software

Ansible

Principe

- Ansible est simple et léger à mettre en œuvre.
- Ansible fonctionne sans agent sur les machines à gérer.
- Ansible utilise le protocole de communication habituel des machines à gérer.
- Ansible utilise des fichiers de description (la CMDB) au format YAML.
- Ansible a été créé par Michael DeHaan.

https://www.ansible.com/
https://docs.ansible.com/
https://forum.ansible.com/
https://fr.wikipedia.org/wiki/Ansible_(logiciel)
https://www.reddit.com/r/ansible/
https://stackoverflow.com/questions/tagged/ansible
https://www.youtube.com/@RedHatAnsible
https://www.youtube.com/@AnsibleAutomation
https://www.learnlinux.tv/getting-started-with-ansible/

Ansible

Catégories de machines

- Ansible distingue deux catégories de machines :
  - la machine de contrôle (*control node*), sur laquelle Ansible est installé (cette machine ne peut pas fonctionner sous Windows)
  - les machines à gérer (*managed nodes*), sur lesquelles l'installation d'un agent spécifique n'est pas nécessaire
- La machine de contrôle communique avec les machines à gérer en utilisant :

  SSH pour les machines à gérer fonctionnant sous UNIX
  WinRM pour les machines à gérer fonctionnant sous Windows

- Les actions sur les machines à gérer sont effectuées au moyen de :

  Python pour les machines à gérer fonctionnant sous UNIX
  PowerShell pour les machines à gérer fonctionnant sous Windows

https://docs.ansible.com/projects/ansible/latest/getting_started/basic_concepts.html#control-node
https://docs.ansible.com/projects/ansible/latest/getting_started/basic_concepts.html#managed-nodes

Ansible

Cycle de vie

- Ansible a été restructuré à partir du début de 2021 et, depuis, est composé de deux parties complémentaires :
  - `ansible-core`
    - contient les outils et les modules élémentaires
    - les trois versions les plus récentes (l'actuelle et les deux précédentes) bénéficient de correctifs (toutes les trois semaines)
  - `ansible`
    - contient de nombreuses collections gérées par la communauté
    - seule la version actuelle bénéficie de correctifs (toutes les trois semaines)
    - une nouvelle version majeure tous les six mois, peu après la sortie de la nouvelle version d'`ansible-core`
    - suit les règles de gestion sémantique de version

https://docs.ansible.com/projects/ansible/latest/reference_appendices/release_and_maintenance.html

https://forum.ansible.com/c/news/releases/18

https://semver.org/lang/fr/

Ansible

Cycle de vie

ansible-core 2.15.0	2023-05-15	ansible 8.0.0	2023-05-30
ansible-core 2.16.0	2023-11-06	ansible 9.0.0	2023-11-21
ansible-core 2.17.0	2024-05-20	ansible 10.0.0	2024-06-04
ansible-core 2.18.0	2024-11-04	ansible 11.0.0	2024-11-19
ansible-core 2.19.0	2025-07-21	ansible 12.0.0	2025-09-09
ansible-core 2.20.0	2025-11-04	ansible 13.0.0	2025-11-19
ansible-core 2.21.0	2026-05-18	ansible 14.0.0	2026-06-02

Table – *Correspondances entre versions d'*`ansible-core` *et d'*`ansible`

https://docs.ansible.com/projects/ansible/devel/roadmap/ansible_core_roadmap_index.html

https://docs.ansible.com/projects/ansible/devel/roadmap/ansible_roadmap_index.html

https://docs.ansible.com/projects/ansible/latest/porting_guides/porting_guides.html

Ansible

Certification Red Hat

- Les plus intrépides peuvent tenter l'examen de certification EX294 (anciennement EX407) proposé par Red Hat :
  - l'inscription coûte 530 € HT
  - l'examen dure 4 h
  - la certification est valable pendant 3 ans

https://www.redhat.com/fr/services/training/ex294-red-hat-certified-engineer-rhce-exam-red-hat-enterprise-linux-9

Ansible

Interfaces Web

- Plusieurs interfaces Web permettent de faire fonctionner Ansible :

  Automation controller (anciennement *Ansible Tower*) est un logiciel commercialisé par Red Hat

  AWX est le logiciel libre parrainé par Red Hat dont certaines versions sont utilisées pour concevoir *Automation controller*

  Semaphore UI est un logiciel libre offrant une solution extérieure à Red Hat

- Attention, ces logiciels permettent d'exécuter Ansible — et ils offrent des possibilités intéressantes dont Ansible ne dispose pas — mais ils ne dispensent en aucun cas d'avoir à mettre au point les fichiers de la CMDB.

https://www.redhat.com/fr/technologies/management/ansible/automation-controller
https://docs.ansible.com/projects/awx/en/latest/
https://www.semui.co/
https://www.learnlinux.tv/complete-ansible-semaphore-tutorial-from-installation-to-automation/

Sommaire

Installation sur la machine de contrôle

BSD

- Paquet `py311-ansible` sur FreeBSD :

```
# pkg install py311-ansible
```

  On peut aussi utiliser les ports.

- Paquet `ansible`[1] sur NetBSD :

```
# pkgin install ansible
```

  On peut aussi utiliser pkgsrc.

- Paquet `ansible` sur OpenBSD :

```
# pkg_add ansible
```

  On peut aussi utiliser les ports.

1. https://cdn.netbsd.org/pub/pkgsrc/current/pkgsrc/sysutils/ansible/

Installation sur la machine de contrôle

Linux

- Paquet `ansible-core`[2] sur *Red Hat Enterprise Linux* 10 :

```
# dnf install ansible-core
```

- Paquet `ansible` sur *Red Hat Enterprise Linux* 8 et 9 (dans le dépôt EPEL[3]) et Fedora[4] :

```
# dnf install epel-release
# dnf install ansible
```

2. https://learn.redhat.com/t5/Automation-Management-Ansible/RHEL-10-and-Ansible/td-p/54804
3. https://docs.ansible.com/projects/ansible/latest/installation_guide/installation_distros.html#installing-ansible-from-epel
4. https://docs.ansible.com/projects/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-fedora-linux
   https://www.youtube.com/watch?v=9eHtelvVi6o

Installation sur la machine de contrôle

Linux

- Paquet `ansible` sur Debian[5] et Ubuntu[6] :

```
# apt install ansible
```

- Sur Ubuntu, il est possible d'installer une version d'Ansible plus récente que celle du dépôt Ubuntu en utilisant un dépôt PPA[7] (*Personal Package Archives*) :

```
$ sudo add-apt-repository -P ppa:ansible/ansible
$ sudo apt install ansible
```

5. https://docs.ansible.com/projects/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-debian
6. https://docs.ansible.com/projects/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu
7. https://launchpad.net/~ansible
   https://www.youtube.com/watch?v=pJjUkogMtpE

Sommaire

Fichiers de configuration

- Les fichiers de configuration d'Ansible se trouvent dans le répertoire `/etc/ansible` :

  `/etc/ansible/ansible.cfg` paramétrage
  `/etc/ansible/hosts` inventaire
  `/etc/ansible/roles` répertoire contenant les rôles

- Ces fichiers ne sont généralement pas utilisés parce qu'il est possible de placer des fichiers équivalents dans la CMDB, ce qui offre plus de souplesse.

Paramétrage d'Ansible

- Le fichier `/etc/ansible/ansible.cfg` contient les paramètres de fonctionnement d'Ansible.
- Il s'agit d'un fichier au format INI.
- Plus généralement, Ansible obtient son paramétrage, dans cet ordre, depuis le premier des fichiers suivants qui existe (les suivants, s'ils existent également, sont ignorés) :
  1. le fichier dont le chemin d'accès absolu est indiqué par la variable d'environnement `ANSIBLE_CONFIG`
  2. `./ansible.cfg`
  3. `~/.ansible.cfg`
  4. `/etc/ansible/ansible.cfg`

---

https://docs.ansible.com/projects/ansible/latest/installation_guide/intro_configuration.html

https://www.youtube.com/watch?v=7zT0j2S6I2w

https://www.youtube.com/watch?v=RM7dguUv21A

https://fr.wikipedia.org/wiki/Fichier_INI

La commande `ansible-config`

- La commande `ansible-config` permet d'afficher le paramétrage d'Ansible :

`init` affiche le paramétrage exhaustif avec des commentaires et la valeur par défaut de chaque paramètre :

```
$ ansible-config init > ansible.cfg
```

avec l'option `--disabled`, les paramètres sont commentés :

```
$ ansible-config init --disabled > ansible.cfg
```

`view` affiche le paramétrage actuel :

```
$ ansible-config view
```

---

https://docs.ansible.com/projects/ansible/latest/cli/ansible-config.html

Inventaire

Inventaire statique

- L'inventaire répertorie les machines gérées par Ansible :

hosts

```
[test]
test1.example.com
test2.example.com

[web]
www.example.com
www-dev.example.com
```

- Le format d'inventaire le plus utilisé est le format INI :
  - les groupes (dont les noms choisis arbitrement sont entre crochets) rassemblent les machines devant subir le même traitement
  - les machines peuvent être indiquées par nom (FQDN) ou adresse IP
- Le chemin d'accès de l'inventaire est indiqué par le paramètre de configuration `inventory` ou l'une des options `-i`, `--inventory` et `--inventory-file`.

https://docs.ansible.com/projects/ansible/latest/inventory_guide/intro_inventory.html

https://www.youtube.com/watch?v=qyFw0nR4eaw

https://fr.wikipedia.org/wiki/Fichier_INI

Inventaire

Inventaire statique

- Il est aussi possible d'indiquer, pour chaque machine, un alias (un nom symbolique) couplé à la variable `ansible_host`, dont la valeur indique le nom ou l'adresse IP de la machine :

hosts

```
[test]
test1   ansible_host=test1.example.com
test2   ansible_host=198.51.100.2
```

Inventaire

Inventaire dynamique

- Si le fichier d'inventaire est exécutable — on parle alors de *script d'inventaire* —, il est exécuté et sa sortie standard — qui doit être au format JSON (*JavaScript Object Notation*) — fournit l'inventaire :

```
{
        "test": {
                "hosts": [ "test1.example.com" , "test2.example.com" ]
        }
}
```

- Un script d'inventaire est censé accepter les options :

`--list` pour afficher tout l'inventaire

`--host <nom d'hôte>` pour n'afficher que les variables de l'hôte indiqué

https://docs.ansible.com/projects/ansible/latest/dev_guide/developing_inventory.html

https://www.json.org/

https://fr.wikipedia.org/wiki/JavaScript_Object_Notation

Inventaire

Répertoire d'inventaire

- Il est aussi possible d'indiquer un répertoire en tant qu'inventaire.
- Ce répertoire doit contenir des fichiers dont le contenu est :
  - soit un inventaire statique
  - soit un inventaire dynamique
- Dans ce cas, l'ensemble cumulé des fichiers contenus dans ce répertoire est utilisé comme inventaire.

https://docs.ansible.com/projects/ansible/latest/inventory_guide/intro_inventory.html#organizing-inventory-in-a-directory

Inventaire

`localhost` implicite

- Il est possible d'utiliser une machine appelée `localhost`, qui correspond à la machine de contrôle et qui est reconnue par Ansible sans avoir besoin de la définir dans l'inventaire.

https://docs.ansible.com/projects/ansible/latest/inventory/implicit_localhost.html

La commande `ansible-inventory`

- La commande `ansible-inventory` permet d'afficher l'inventaire (par défaut au format JSON) :

`--list` affiche l'inventaire complet :

```
$ ansible-inventory --list
[...]
```

`--graph` affiche l'inventaire sous la forme d'un arbre :

```
$ ansible-inventory --graph
@all:
  |--@ungrouped:
  |--@test:
  |  |--test1.example.com
  |  |--test2.example.com
```

https://docs.ansible.com/projects/ansible/latest/cli/ansible-inventory.html

Organisation des fichiers

- Ansible n'impose aucune structure particulière pour les fichiers de la CMDB mais il est courant de les organiser en ayant un répertoire par projet, contenant :
  - `ansible.cfg` configuration du projet
  - `hosts` inventaire du projet
  - `roles` rôles du projet
  - `*.yml` livrets du projet

```
<projet>/
├── ansible.cfg
├── hosts
├── roles/
├── livret-1.yml
├── livret-2.yml
└── ...
```

ansible.cfg

```
[defaults]
inventory = hosts
```


Sommaire

Communication entre machines

- Ansible utilise SSH pour communiquer avec les machines à gérer fonctionnant sous UNIX.
- Quel compte peut-on utiliser pour se connecter aux machines distantes ?
- Deux approches sont acceptables en matière de sécurité :
  - `root` avec authentification par clé
  - un compte non privilégié puis une élévation de privilèges (généralement effectuée au moyen de la commande `sudo`)

  La deuxième approche permet une meilleure traçabilité des actions de `root` sur les machines distantes lorsqu'on travaille directement en ligne de commande sur les machines à gérer mais elle est plus limitée dans le cadre de l'utilisation d'Ansible puisque la seule commande qu'il exécute est `python` et que `sudo` journalise uniquement la ligne de commande exécutée et pas le contenu du script que `python` interprète.

Clés SSH

- Création d'un couple de clés sur la machine de contrôle :

OpenSSH ⩾ 9.5p1

```
$ ssh-keygen
```

OpenSSH < 9.5p1

```
$ ssh-keygen -t ed25519
```

- Transfert de la clé publique depuis la machine de contrôle vers une machine à gérer :
  - en utilisant un compte utilisateur identique à celui de la machine de contrôle :

    ```
    $ ssh-copy-id test1.example.com
    ```

  - en utilisant un compte utilisateur différent de celui de la machine de contrôle :

    ```
    $ ssh-copy-id root@test1.example.com
    ```

Clés SSH

- Afin d'éviter d'avoir à saisir de manière répétée la phrase de passe de la clé privée, il est préférable de l'enregistrer en mémoire.
- Pour cela, il faut lancer l'agent SSH s'il ne l'est pas déjà :

```
$ eval `ssh-agent`
Agent pid 1664
```

```
$ eval $(ssh-agent)
Agent pid 1664
```

- Puis on peut mémoriser la clé privée et sa phrase de passe :

```
$ ssh-add
[...]
```


Tests de communication

- Le module `ansible.builtin.ping` permet de savoir si la communication avec les machines indiquées dans l'inventaire se fait correctement :

```
$ ansible -m ansible.builtin.ping all
test1.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
test2.example.com | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```


https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/ping_module.html

Tests de communication

- S'il est nécessaire de se connecter sous l'identité d'un utilisateur différent de celui qui exécute la commande `ansible`, il faut utiliser l'option `-u` suivie de l'identifiant de cet utilisateur :

```
$ ansible -m ansible.builtin.ping -u root all
[...]
```


Tests de communication

- Le module `ansible.builtin.setup` récupère et affiche de nombreuses informations factuelles (*facts*) stockées par Ansible dans des variables :

```
$ ansible -m ansible.builtin.setup all
test1.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "198.51.100.33"
        ],
        "ansible_all_ipv6_addresses": [
            "2001:db8:1234:5678:cafe:abba:d0d0:c0c0",
            "fe80::cafe:abba:d0d0:c0c0"
        ],
[...]
```

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

https://www.youtube.com/watch?v=dGjId7RSLek

Commande *ad hoc*

- Cette façon d'utiliser un module au moyen de la commande `ansible` constitue une commande *ad hoc* (locution latine signifiant : « pour cela »).
- Une commande *ad hoc* permet d'effectuer une action ponctuelle au moyen d'un des modules d'Ansible sans utiliser la CMDB.
- Par conséquent, une commande *ad hoc* n'a pas vocation à être réutilisable.

---

https://docs.ansible.com/projects/ansible/latest/command_guide/intro_adhoc.html
https://www.youtube.com/watch?v=-ytUl6sl9Bw
https://www.dictionnaire-academie.fr/article/A9A0527
https://fr.wiktionary.org/wiki/ad_hoc

Commande *ad hoc*

- La syntaxe d'une commande *ad hoc* utilise généralement les options `-m` et `-a` de la commande `ansible` :

```
$ ansible -m <module> -a <paramètres du module> <quoi>
```

- Les paramètres du module sont spécifiés au moyen d'une chaîne de caractères pouvant prendre deux formes :

```
$ ansible -m ansible.builtin.copy \
> -a $'src=toto dest=/tmp/toto owner=root group=root mode=\'444\'' \
> -bK test
```

```
$ ansible -m ansible.builtin.copy \
> -a '{"src": "toto", "dest": "/tmp/toto", "owner": "root", "group": "root", "mode": "444"}' \
> -bK test
```

L'option `-b` est équivalente au mot clé `become`.

Sommaire

YAML

- YAML signifie : *YAML Ain't Markup Language* (acronyme récursif).
- Il s'agit d'un format général de représentation de données.
- YAML est le format utilisé par Ansible pour représenter la CMDB.
- La commande `yamllint` (paquet `yamllint` sur toutes les distributions) permet de vérifier la syntaxe d'un fichier au format YAML.

---

https://yaml.org/
https://fr.wikipedia.org/wiki/YAML
https://yaml-multiline.info/
https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
https://github.com/adrienverge/yamllint
https://www.redhat.com/sysadmin/check-yaml-yamllint

Exemple

coquillettes.yml

```
---
recette: coquillettes au beurre
convives: 1
ingrédients:
  - ingrédient: coquillettes
    quantité: 70g
  - ingrédient: beurre
    quantité: 15g
étapes:
  - faire bouillir l'eau
  - faire cuire les coquillettes dedans
  - les égoutter
  - mettre du beurre
# en variante, ajouter du fromage
```

- Un fichier YAML a généralement une extension `.yml`.
- Un fichier YAML est censé commencer par une ligne contenant trois tirets. Ce n'est plus indispensable pour Ansible mais c'est une habitude qu'on a tendance à respecter.
- Les commentaires commencent par un croisillon # et se poursuivent jusqu'à la fin de la ligne.

Collections

- Un fichier YAML peut contenir deux types de structures de données (qu'on appelle *collections* en terminologie YAML) :

  des *séquences* contenant des éléments ordonnés :

```
- faire bouillir l'eau
- faire cuire les coquillettes dedans
- les égoutter
- mettre du beurre
```

  des *correspondances* (*mappings*) contenant des éléments (leur ordre est sans importance) composés d'une clé (unique) et d'une valeur :

```
recette: coquillettes au beurre
convives: 1
```

Style des collections

- Les collections peuvent s'écrire selon deux styles :

le *style en bloc* (*block style*)

```
- entrée
- plat
- dessert
```

```
manger: choucroute
boire: bière
```

le *style en flux* (*flow style*)

```
[ entrée , plat , dessert ]
```

```
{ manger: choucroute , boire: bière }
```


Imbrication des collections

- Les collections peuvent être imbriquées.
- L'indentation — généralement de deux espaces — est importante car elle traduit l'imbrication des collections.
- Il faut utiliser des espaces pour l'indentation, pas des tabulations (si l'on utilise `vi` ou Vim, la commande `:set list` permet de matérialiser visuellement les tabulations et les fins de ligne).
- Ici, par exemple, la valeur de l'élément de correspondance est une séquence dont chaque élément est une correspondance :

```
ingrédients:
  - ingrédient: coquillettes
    quantité: 70g
  - ingrédient: beurre
    quantité: 15g
```

- Toutes les combinaisons possibles sont autorisées.

Éditeurs de texte

Vim

- Le format YAML impose un respect très strict de l'alignement vertical des éléments de même niveau hiérarchique.
- L'option `cursorcolumn` (ou sa forme abrégée `cuc`) permet de matérialiser visuellement la ligne verticale sur laquelle se trouve le curseur, ce qui facilite l'alignement vertical des éléments. Ceci peut se faire :
  - ponctuellement, pour la session en cours :

```
:set cursorcolumn
```

```
:set cuc
```

  - de manière systématique, dans le fichier de configuration de Vim :

~/.vimrc
```
set cursorcolumn
```

~/.vimrc
```
set cuc
```

---

http://vimdoc.sourceforge.net/htmldoc/options.html#'cursorcolumn'

Éditeurs de texte

Les autres

- D'autres éditeurs de texte disposent d'un mécanisme de matérialisation de la ligne verticale sur laquelle se trouve le curseur.
- Il aura fallu attendre la version 6.0 (2021-12-15) de GNU nano pour qu'il gère — enfin — la coloration syntaxique de fichiers au format YAML.

Sommaire

Sommaire

Généralités

Les livrets

- Les livrets (*playbooks*) sont les fichiers élémentaires constituant la CMDB d'Ansible.
- Les livrets sont des fichiers au format YAML.

- D'après le Wiktionnaire anglophone, le mot *playbook* signifie :
  1. *A book containing the text of a play or plays.*
  2. *A book of games and amusements for children.*
  3. *(US, American football) A book of strategies (plays) for use in American football (and by extension other sports or disciplines).*
  4. *(US, figurative) A set of commonly employed tactics and strategies.*

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks.html
https://www.youtube.com/watch?v=h3EcXPGmR7g
https://en.wiktionary.org/wiki/playbook

Généralités

Exemple

principes.yml

```
---
- name: Premier livret tout simple
  hosts: test
  tasks:
    - name: Commande id
      ansible.builtin.command:
        cmd: id
    - name: Commande uname -a
      ansible.builtin.command:
        cmd: uname -a
```

- Structure générale :
  - `name` description du livret ou de la tâche
  - `hosts` groupe (ou machine) concerné dans l'inventaire
  - `tasks` liste des tâches (traitements) à effectuer
- Et le plus important :

  `ansible.builtin.command` module à utiliser suivi de ses paramètres

  Il existe de nombreux autres modules, chacun ayant une fonction spécifique.

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_intro.html

La commande ansible-playbook

- Pour faire en sorte d'appliquer le contenu d'un livret aux machines concernées, on utilise la commande ansible-playbook :

```
$ ansible-playbook -v principes.yml
```

- L'option -v permet d'afficher le détail des tâches. On peut augmenter le nombre de v (-vv, -vvv, -vvvv, -vvvvv) pour afficher encore plus d'informations.
- Il n'y a aucune raison d'exécuter la commande ansible-playbook sous l'identité du super-utilisateur (root) sur la machine de contrôle. La commande ansible-playbook doit toujours être exécutée sous l'identité d'un utilisateur non privilégié.
- La commande ansible-playbook dispose de nombreuses options. Pour en afficher la liste :

```
$ ansible-playbook -h
```

https://docs.ansible.com/projects/ansible/latest/cli/ansible-playbook.html

La commande ansible-playbook

- L'une des options de la commande ansible-playbook les plus utiles est l'option -C (ou --check) qui permet d'afficher ce qui se produirait en l'absence de cette option sans effectuer aucune modification sur les machines à gérer (*dry run*) :

```
$ ansible-playbook -vC livret.yml
```

- Cette option fonctionne correctement avec la majorité des modules mais certains, en raison de leur mode de fonctionnement, ne donneront pas un résultat fiable.

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_checkmode.html

Généralités

remote_user

principes-remote_user.yml

```
---
- name: Connexion sous l'identité d'un utilisateur spécifique
  hosts: test
  remote_user: root
  tasks:
    - name: Commande id
      ansible.builtin.command:
        cmd: id
    - name: Commande uname -a
      ansible.builtin.command:
        cmd: uname -a
```

`remote_user` identifiant à utiliser pour se connecter par SSH

Généralités

become

principes-become-1.yml

```
---
- name: Changement d'identité
  hosts: test
  remote_user: ansible
  become: true
  # become_user: root
  # become_method: sudo
  tasks:
    - name: Commande id
      ansible.builtin.command:
        cmd: id
    - name: Commande uname -a
      ansible.builtin.command:
        cmd: uname -a
```

`become` indique qu'il faut changer d'identité

`become_user` identifiant de l'utilisateur qu'il faut devenir (`root` par défaut)

`become_method` méthode à utiliser pour le changement d'identité (`sudo` par défaut)

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_privilege_escalation.html

Généralités

become

- L'option `-K` (ou `--ask-become-pass`) de la commande `ansible-playbook` permet la saisie du mot de passe destiné à `sudo` :

```
$ ansible-playbook -vK principes-become-1.yml
```


Généralités

become

- Il est possible de changer d'identité au niveau d'une seule tâche, en application du *principe de moindre privilège*.

principes-become-2.yml

```
---
- name: Changement d'identité
  hosts: test
  tasks:
    - name: Commande id normale
      ansible.builtin.command:
        cmd: id
    - name: Commande id privilégiée
      ansible.builtin.command:
        cmd: id
      become: true
    - name: Commande id normale
      ansible.builtin.command:
        cmd: id
```


https://fr.wikipedia.org/wiki/Principe_de_moindre_privilège

Sommaire

Les variables

Principe

- La documentation d'Ansible recense pas moins de 22 façons différentes de définir des variables.
- Pour rester simple, les façons les plus utilisées de définir des variables sont (dans l'ordre de priorité) :
  1. sur la ligne de commande
  2. dans le livret
  3. par Ansible
  4. dans l'inventaire

`https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_variables.html`

Les variables

Variables définies sur la ligne de commande

```
$ ansible-playbook -v -e var=toto variables-lignecom.yml
```

```
$ ansible-playbook -v --extra-vars var=toto variables-lignecom.yml
```

variables-lignecom.yml

```
---
- name: Variables définies sur la ligne de commande
  hosts: test
  tasks:
    - name: Affichage de la variable var
      ansible.builtin.debug:
        msg: 'La variable var contient : {{ var }}'
```


Les variables

Variables définies dans le livret et variables définies par Ansible

variables-livret-ansible.yml

```
---
- name: Variables définies dans le livret et variables définies par Ansible
  hosts: test
  become: true
  vars:
    fichier: gopher
    version: 1.0
  tasks:
    - name: Copie du fichier de configuration
      ansible.builtin.copy:
        src: conf-{{ fichier }}-{{ version }}-{{ ansible_os_family }}.conf
        dest: /etc/{{ fichier }}.conf
        owner: root
        group: root
        mode: '444'
```

Les variables

Variables définies dans l'inventaire

/etc/ansible/hosts

```
[test]
test1.example.com var=test1
test2.example.com var=test2
```

variables-inventaire.yml

```
---
- name: Variables définies dans l'inventaire
  hosts: test
  tasks:
    - name: Affichage de la variable var
      ansible.builtin.debug:
        msg: 'La variable var contient : {{ var }}'
```

/etc/ansible/hosts

```
[test]
test1.example.com
test2.example.com

[test:vars]
ansible_user=ansible
ansible_become_pass=toto
```


Les variables

Guillemets

- Toute valeur d'un élément de collection YAML (séquence ou correspondance) qui commence par une variable doit être placée entre guillemets :

variables-guillemets.yml

```
---
- name: Variables et guillemets
  hosts: test
  vars:
    srv: mr-fabulous
    clt: blues-brothers
  tasks:
    - name: Test avec des guillemets
      ansible.builtin.debug:
        msg: "{{ item }}.example.com"
      loop:
        - "{{ srv }}"
        - "{{ clt }}"
```


https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_variables.html#when-to-quote-variables-a-yaml-gotcha

Les variables

Filtres

- Les *filtres* permettent d'effectuer des traitements sur le contenu des variables lors de leur utilisation (le contenu des variables n'est pas modifié) :

variables-filtres.yml

```
---
- name: Variables et filtres
  hosts: test
  vars:
    test: GRAND
  tasks:
    - name: Sans filtre
      ansible.builtin.debug:
        msg: "{{ test }}"
    - name: Avec filtre
      ansible.builtin.debug:
        msg: "{{ test | lower }}"
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_filters.html

https://jinja.palletsprojects.com/en/3.1.x/templates/#builtin-filters

Sommaire

Les conditions

- Une tâche peut n'être exécutée que si une condition est vérifiée.
- La condition est indiquée par l'instruction when.

Attention

La syntaxe de l'instruction when est inhabituelle :

- les variables ne doivent pas être utilisées entre des paires d'accolades
- les chaînes de caractères doivent être entourées par des apostrophes

when.yml

```
---
- name: Conditions
  hosts: test
  become: true
  tasks:
    - name: Paquet apache2 sur debianoïde
      ansible.builtin.package:
        name: apache2
        state: present
      when: ansible_os_family == 'Debian'
    - name: Paquet httpd sur redhatoïde
      ansible.builtin.package:
        name: httpd
        state: present
      when: ansible_os_family == 'RedHat'
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_conditionals.html

https://jinja.palletsprojects.com/en/3.1.x/templates/#builtin-tests

https://www.youtube.com/watch?v=x-sQpE1S9kQ

Sommaire

Les boucles

Principe

- Il est possible d'exécuter plusieurs fois le même module avec certains paramètres différents au moyen d'une *boucle*.
- Dans la boucle, la variable `item` prendra successivement les valeurs de la séquence suivant l'instruction `loop`.
- Cette instruction `loop` a été ajoutée dans la version 2.5 d'Ansible, sortie en mars 2018. Auparavant, on utilisait — et on peut toujours utiliser aujourd'hui — une dizaine de syntaxes `with_<type>`, avec un `<type>` différent pour chaque type de boucle possible.

---

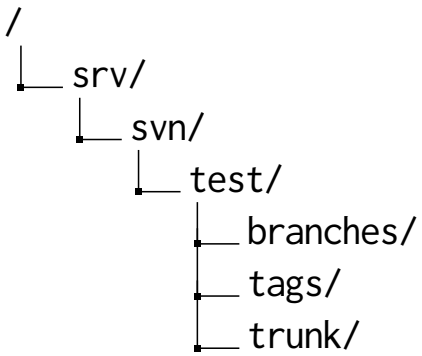
https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_loops.html

Les boucles

Exemple de boucle simple

boucle-simple.yml

```
---
- name: Boucle simple
  hosts: test
  become: true
  vars:
    projet_svn: test
  tasks:
    - name: Répertoires standards SVN
      ansible.builtin.file:
        path: /srv/svn/{{ projet_svn }}/{{ item }}
        state: directory
        owner: root
        group: root
        mode: '755'
      loop: [ branches, tags, trunk ]
```



Les boucles

Exemple de boucle sur une correspondance

boucle-correspondance.yml

```
---
- name: Boucle sur une correspondance
  hosts: test
  become: true
  vars:
    crt: /etc/pki/tls/certs/{{ inventory_hostname }}-crt.pem
    key: /etc/pki/tls/private/{{ inventory_hostname }}-key.pem
  tasks:
    - name: Droits d'accès sur la clé privée et le certificat TLS
      ansible.builtin.file: path={{ item.path }} mode={{ item.mode }}
      loop:
        - { path: "{{ crt }}" , mode: '444' } # noqa: yaml[commas]
        - { path: "{{ key }}" , mode: '400' } # noqa: yaml[commas]
```


Sommaire

Les modèles

Principe

- Les modèles (*templates*) sont des fichiers situés sur la machine de contrôle dans lesquels Ansible effectue diverses transformations avant de transférer le résultat sur les machines à gérer.
- Ansible utilise à cet effet une bibliothèque pour le langage de programmation Python appelée Jinja.

---

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_templating.html
https://palletsprojects.com/p/jinja/
https://fr.wikipedia.org/wiki/Jinja_(moteur_de_template)
https://jpmens.net/2020/09/29/using-ansible-managed/
https://www.youtube.com/watch?v=vjxVko4Y56E

Les modèles

Exemple de modèle

nginx-vhost.conf.j2

```
# {{ ansible_managed }}

server
{
        listen 80 ;
        listen [::]:80 ;

        server_name {{ inventory_hostname }} ;

        root /srv/www/{{ inventory_hostname }} ;
        index index.xhtml ;

        access_log      /var/log/nginx/{{ inventory_hostname }}-access_log ;
        error_log       /var/log/nginx/{{ inventory_hostname }}-error_log ;
}
```

Les modèles

Exemple de livret

template1.yml

```
---
- name: Modèle 1
  hosts: test
  become: true
  tasks:
    - name: Configuration hôte virtuel
      ansible.builtin.template:
        src: nginx-vhost.conf.j2
        dest: /etc/nginx/conf.d/{{ inventory_hostname }}.conf
        owner: root
        group: root
        mode: '444'
```


Les modèles

Exemple de livret

template2.yml

```
---
- name: Modèle 2
  hosts: test
  become: true
  vars:
    ethers:
      - { fqdn: test1.example.com, ethernet: 11:00:aa:11:22:33 }
      - { fqdn: test2.example.com, ethernet: 9c:93:e4:44:55:66 }
      - { fqdn: test3.example.com, ethernet: e0:cb:1d:77:88:99 }
  tasks:
    - name: Configuration DHCP
      ansible.builtin.template:
        src: dhcpd.conf.j2
        dest: /srv/dhcp/dhcpd.conf
        owner: root
        group: root
        mode: '444'
```

Les modèles

Exemple de modèle et résultat

dhcpd.conf.j2

```
# {{ ansible_managed }}
{% for host in ethers %}

host {{ host.fqdn }}
{
        option host-name "{{ host.fqdn }}" ;
        fixed-address {{ host.fqdn }} ;
        hardware ethernet {{ host.ethernet }} ;
}
{% endfor %}
```

/srv/dhcp/dhcpd.conf

```
# Ansible managed

host test1.example.com
{
        option host-name "test1.example.com" ;
        fixed-address test1.example.com ;
        hardware ethernet 11:00:aa:11:22:33 ;
}

host test2.example.com
{
        option host-name "test2.example.com" ;
        fixed-address test2.example.com ;
        hardware ethernet 9c:93:e4:44:55:66 ;
}

host test3.example.com
{
        option host-name "test3.example.com" ;
        fixed-address test3.example.com ;
        hardware ethernet e0:cb:1d:77:88:99 ;
}
```


Sommaire

Les gestionnaires

Principe

- Les gestionnaires (*handlers*) permettent de déclencher certaines actions après la fin du traitement des tâches.
- Une tâche peut déclencher un ou plusieurs gestionnaires.
- Chaque gestionnaire est associé à une ou plusieurs tâches.
- Un gestionnaire n'est déclenché que si au moins l'une de ses tâches associées a effectué une action (c'est-à-dire s'il n'y a pas eu idempotence).
- Les gestionnaires sont exécutés dans l'ordre dans lequel ils sont définis dans la section `handlers`.
- Le même gestionnaire peut être associé à plusieurs tâches mais il ne sera exécuté qu'une seule fois.

```
[...]
  tasks:
    - name: tâche 1
      ansible.builtin.module:
        [...]
      notify:
        - g2
    - name: tâche 2
      ansible.builtin.module:
        [...]
      notify:
        - g1
    - name: tâche 3
      ansible.builtin.module:
        [...]
      notify:
        - g2
  handlers:
    - name: g1
      ansible.builtin.module:
        [...]
    - name: g2
      ansible.builtin.module:
        [...]
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_handlers.html

Les gestionnaires

Exemples

`handlers-listen.yml` (syntaxe moderne)

```
---
- name: Gestionnaire
  hosts: test
  become: true
  tasks:
    - name: Configuration syslog
      ansible.builtin.copy:
        src: openldap.conf
        dest: /etc/rsyslog.d/openldap.conf
        owner: root
        group: root
        mode: '444'
      notify:
        - restart rsyslog
  handlers:
    - name: Redémarrage de rsyslog
      ansible.builtin.service:
        name: rsyslog
        state: restarted
      listen: restart rsyslog
```

`handlers-name.yml` (syntaxe historique)

```
---
- name: Gestionnaire
  hosts: test
  become: true
  tasks:
    - name: Configuration syslog
      ansible.builtin.copy:
        src: openldap.conf
        dest: /etc/rsyslog.d/openldap.conf
        owner: root
        group: root
        mode: '444'
      notify:
        - restart rsyslog
  handlers:
    - name: restart rsyslog # noqa: name[casing]
      ansible.builtin.service:
        name: rsyslog
        state: restarted
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_handlers.html#naming-handlers

Sommaire

Principe

- Les livrets peuvent vite atteindre une taille importante et comporter entre eux des parties redondantes. Aussi est-il possible d'organiser la CMDB sous forme de *rôles*.
- L'objectif des rôles est d'éliminer les redondances dans la CMDB en la rendant modulaire, de faciliter la réutilisation du code et de le rendre plus générique.
- Un rôle est simplement le résultat de la décomposition des différentes parties d'un livret mettant en œuvre une fonction précise en organisant ces parties dans un ensemble de répertoires et de fichiers.

`https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_reuse_roles.html`

`https://www.youtube.com/watch?v=kjQWzRX9tB0`

Structure

```
roles/
└── <nom du rôle>/
    ├── defaults/
    │   └── main.yml
    ├── files/
    ├── handlers/
    │   └── main.yml
    ├── meta/
    │   └── main.yml
    ├── tasks/
    │   └── main.yml
    ├── templates/
    └── vars/
        └── main.yml
```

- Le nom d'un rôle (et donc le répertoire correspondant) ne peut contenir que des caractères alphanumériques bas de casse et des tirets bas et doit commencer par une lettre.
- Les répertoires `defaults`, `handlers`, `meta`, `tasks` et `vars` peuvent contenir un fichier `main.yml`, qui sera automatiquement pris en compte.
- Les fichiers utilisés par le module `ansible.builtin.copy` sont recherchés dans le répertoire `files`.
- Les fichiers utilisés par le module `ansible.builtin.template` sont recherchés dans le répertoire `templates`.

Utilisation

- Lorqu'on utilise un ou plusieurs rôles, la commande `ansible-playbook` est exécutée avec en argument un livret faisant la liaison entre l'inventaire et le ou les rôles à appliquer aux machines.
- Ce livret, dont le nom est indifférent, ne fait pas partie du rôle et doit donc être placé à l'extérieur du répertoire `roles`.

livret-roles.yml

```yaml
---
- name: Rôles
  hosts: test
  become: true
  roles:
    - nginx
    - postgresql
```

livret-roles-vars.yml

```yaml
---
- name: Rôle avec variables
  hosts: test
  become: true
  roles:
    - role: dns
      vars:
        dns_zones_primary:
          - example.com
          - example.net
```

Exemple

roles/nginx/tasks/main.yml (1/3)

```
---

- name: Installation de nginx
  ansible.builtin.package:
    name: nginx
    state: present
  notify:
    - nginx_enable_start
```


Exemple

roles/nginx/tasks/main.yml (2/3)

```
- name: Création de la clé privée et du certificat TLS
  ansible.builtin.command: openssl req -newkey rsa:2048
                           -nodes -x509 -days 365
                           -subj /CN={{ inventory_hostname }}
                           -keyout {{ nginx_key }}
                           -out {{ nginx_crt }}
  args:
    creates: "{{ nginx_crt }}"
```

Exemple

roles/nginx/tasks/main.yml (3/3)

```
- name: Droits d'accès sur la clé privée et le certificat TLS
  ansible.builtin.file: path={{ item.path }} mode={{ item.mode }}
  loop:
    - { path: "{{ nginx_crt }}", mode: '444' }
    - { path: "{{ nginx_key }}", mode: '400' }

- name: Hôte virtuel TLS
  ansible.builtin.template:
    src: nginx-vhost-tls.conf.j2
    dest: /etc/nginx/conf.d/{{ inventory_hostname }}-tls.conf
    owner: root
    group: root
    mode: '444'
```


Exemple

roles/nginx/vars/main.yml

```
---

nginx_crt: /etc/pki/tls/certs/{{ inventory_hostname }}-crt.pem
nginx_key: /etc/pki/tls/private/{{ inventory_hostname }}-key.pem
```

roles/nginx/handlers/main.yml

```
---

- name: nginx_enable_start # noqa: name[casing]
  ansible.builtin.service:
    name: nginx
    enabled: true
    state: started
```

Utiliser d'autres fichiers

- Les rôles peuvent faire appel à d'autres fichiers que `main.yml`, en les référençant explicitement depuis le fichier `tasks/main.yml` :

roles/apache/tasks/main.yml

```
---

- name: Inclusion des variables
  ansible.builtin.include_vars: "{{ ansible_os_family }}.yml"

- name: Import des tâches d'installation
  ansible.builtin.import_tasks: installation.yml
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_reuse.html

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/include_vars_module.html

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/import_tasks_module.html

Utiliser d'autres fichiers

roles/apache/vars/Debian.yml

```
---

apache_paquet: apache2
```

roles/apache/vars/RedHat.yml

```
---

apache_paquet: httpd
```

roles/apache/tasks/installation.yml

```
---

- name: Installation d'Apache
  ansible.builtin.package:
    name: "{{ apache_paquet }}"
    state: present
```

Alternance entre tâches et rôles

- Si l'on a besoin d'un livret dans lequel il faut alterner entre tâches et rôles, le mot clé `roles` ne convient pas.
- On peut alors utiliser le module `ansible.builtin.include_role` ou le module `ansible.builtin.import_role`.

livret-roles.yml

```
---
- name: Rôles
  hosts: test
  become: true
  roles:
    - nginx
    - postgresql
```


https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_reuse_roles.html#using-roles
https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_reuse.html#reusing-files-and-roles

Le module `ansible.builtin.include_role`

module-ansible.builtin.include_role.yml

```
---
- name: Module ansible.builtin.include_role
  hosts: test
  tasks:
    - name: Avant
      ansible.builtin.debug:
        msg: avant
    - name: Rôle
      ansible.builtin.include_role:
        name: role
      loop: [ 1 , 2 , 3 ]
    - name: Après
      ansible.builtin.debug:
        msg: après
```

- Le module `ansible.builtin.include_role` inclut un rôle (de manière dynamique).

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/include_role_module.html

Le module ansible.builtin.import_role

module-ansible.builtin.import_role.yml

```
---
- name: Module ansible.builtin.import_role
  hosts: test
  tasks:
    - name: Avant
      ansible.builtin.debug:
        msg: avant
    - name: Rôle
      ansible.builtin.import_role:
        name: role
    - name: Après
      ansible.builtin.debug:
        msg: après
```

- Le module ansible.builtin.import_role importe un rôle (de manière statique).

---

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/import_role_module.html

Sommaire

Principe

- Une *collection* permet de regrouper des fichiers (livrets, rôles, etc.) pour Ansible.

```
<espace de nommage>/
└── <nom de la collection>/
    ├── docs/
    ├── galaxy.yml
    ├── plugins/
    ├── README.md
    ├── roles/
    │   ├── role1/
    │   ├── role2/
    │   └── .../
    ├── playbooks/
    └── tests/
```

---

https://docs.ansible.com/projects/ansible/latest/dev_guide/developing_collections.html

https://docs.ansible.com/projects/ansible/latest/collections_guide/

Sommaire

Principe

- Galaxy est un site Web permettant le téléchargement et le télédépôt de rôles et de collections réalisés par la communauté.

https://galaxy.ansible.com/
https://docs.ansible.com/projects/ansible/latest/galaxy/user_guide.html

La commande `ansible-galaxy`

Principe

- La commande `ansible-galaxy` permet la gestion (recherche, téléchargement, télédépôt, etc.) des rôles et des collections depuis Galaxy.
- Elle s'utilise suivie d'un type, d'une action et, au besoin, d'options :

```
$ ansible-galaxy [type] <action> [options]
```

- Le type peut être :

  `collection` pour gérer les collections
  `role` pour gérer les rôles

  En l'absence de type, `role` est utilisé implicitement.

https://docs.ansible.com/projects/ansible/latest/cli/ansible-galaxy.html

La commande ansible-galaxy

Gestion des rôles

`init` crée le squelette d'un nouveau rôle dont le nom est en argument :

```
$ ansible-galaxy role init nom_role
```

`search` effectue une recherche parmi les rôles disponibles :

```
$ ansible-galaxy role search nginx
```

`info` affiche les informations du rôle en argument :

```
$ ansible-galaxy role info bennojoy.nginx
```


La commande ansible-galaxy

Gestion des rôles

`install` télécharge et installe les rôles en arguments :

```
$ ansible-galaxy role install bennojoy.nginx
```

Les rôles sont installés dans le premier répertoire accessible en écriture dans ceux spécifiés par le paramètre `roles_path`. Il est possible d'indiquer explicitement le répertoire d'installation au moyen de l'option `-p` :

```
$ ansible-galaxy role install -p [...] bennojoy.nginx
```

La commande ansible-galaxy

Gestion des rôles

`list` affiche la liste des rôles installés :

```
$ ansible-galaxy role list
```

L'option -p permet d'indiquer le répertoire contenant les rôles.

`remove` supprime les rôles en arguments :

```
$ ansible-galaxy role remove bennojoy.nginx
```

L'option -p permet d'indiquer le répertoire contenant les rôles.

La commande ansible-galaxy

Gestion des collections

`init` crée le squelette d'une nouvelle collection dont le nom qualifié est en argument :

```
$ ansible-galaxy collection init espace_de_nommage.nom_collection
```

`install` télécharge et installe les collections en arguments :

```
$ ansible-galaxy collection install icinga.icinga
```

Les collections sont installées dans le premier répertoire accessible en écriture dans ceux spécifiés par le paramètre `collections_paths`. Il est possible d'indiquer explicitement le répertoire d'installation au moyen de l'option -p :

```
$ ansible-galaxy collection install -p [...] icinga.icinga
```

Sommaire

Sommaire

Les étiquettes

Principe

- Les étiquettes (*tags*) sont des chaînes de caractères arbitraires associées à des structures d'Ansible — principalement des tâches ou des rôles — et permettant leur exécution sélective.
- Lors d'une exécution sélective au moyen d'étiquettes, les gestionnaires associés aux tâches exécutées sont déclenchés.

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_tags.html

https://www.youtube.com/watch?v=mPi6SwC4SJ8

Les étiquettes

Exemples

tags.yml (1/3)

```
---
- name: Étiquettes
  hosts: test
  become: true
  tasks:
    - name: Install. nginx
      ansible.builtin.package:
        name: nginx
        state: present
      tags:
        - installation
```

tags.yml (2/3)

```
    - name: Config. nginx
      ansible.builtin.copy:
        src: nginx.conf
        dest: /etc/nginx/nginx.conf
        owner: root
        group: root
        mode: '444'
      tags:
        - configuration
```

Les étiquettes

Exemples

tags.yml (3/3)

```
    - name: Restart nginx
      ansible.builtin.service:
        name: nginx
        state: restarted
      tags:
        - service
```

tags-role.yml

```
---
- name: Étiquettes avec rôle
  hosts: test
  become: true
  roles:
    - role: nginx
      tags:
        - nginx_configuration
```


Les étiquettes

Utilisation

- La commande `ansible-playbook` dispose d' options permettant de sélectionner les tâches à exécuter — ou à ne pas exécuter — en fonction des étiquettes indiquées :
  - L'option `-t` (ou `--tags`) permet d'exécuter les tâches correspondant aux étiquettes indiquées (les autres ne le sont pas) :

```
$ ansible-playbook -v -t installation tags.yml
```

```
$ ansible-playbook -v -t configuration,service tags.yml
```

  - L'option `--skip-tags` permet de ne pas exécuter les tâches correspondant aux étiquettes indiquées (les autres le sont) :

```
$ ansible-playbook -v --skip-tags installation tags.yml
```

Les étiquettes

Étiquettes spéciales

- Certaines étiquettes ont une signification spéciale :

`always` fait en sorte que les tâches correspondantes soient toujours exécutées, sauf si l'étiquette est explicitement ignorée :

```
$ ansible-playbook -v --skip-tags always tags.yml
```

`never` fait en sorte que les tâches correspondantes ne soient jamais exécutées, sauf si l'étiquette est explicitement sélectionnée :

```
$ ansible-playbook -v -t never tags.yml
```


Sommaire

Enregistrer le résultat d'une tâche dans une variable

- L'instruction `register` permet d'enregistrer le résultat d'une tâche dans une variable afin de le réutiliser dans une autre tâche :

register.yml

```
---
- name: Enregistrement du résultat
  hosts: test
  tasks:
    - name: Fichier temporaire
      ansible.builtin.tempfile:
      register: temp
    - name: Droits d'accès
      ansible.builtin.file:
        path: "{{ temp.path }}"
        mode: '644'
```


https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_variables.html#registering-variables

Enregistrer le résultat d'une tâche dans une variable

- Le contenu exact de la variable créée par l'instruction `register` varie selon le module auquel elle s'applique.
- Le module `debug` permet d'en afficher le contenu.

register-debug.yml

```
---
- name: Enregistrement puis affichage
  hosts: test
  tasks:
    - name: Fichier temporaire
      ansible.builtin.tempfile:
      register: temp
    - name: Variable temp
      ansible.builtin.debug:
        var: temp
```

```
$ ansible-playbook -v register-debug.yml
[...]
TASK [Variable temp] ************************
ok: [test1.example.com] => {
    "temp": {
        "changed": true,
        "failed": false,
        "gid": 3351,
        "group": "beatles",
        "mode": "0600",
        "owner": "lennon",
        "path": "/tmp/ansible.zq8ba2xm",
        "size": 0,
        "state": "file",
        "uid": 1664
    }
}
[...]
```

Enregistrer le résultat d'une tâche dans une variable

- Utilisée avec une boucle, l'instruction `register` enregistre le résultat de toutes les tâches de la boucle dans la variable indiquée :

register-loop.yml

```
---
- name: Enregistrement et boucle
  hosts: test
  tasks:
    - name: Fichier temporaire
      ansible.builtin.tempfile:
      loop: [ 1, 2, 3 ]
      register: temp
    - name: Droits d'accès
      ansible.builtin.file:
        path: "{{ item.path }}"
        mode: '644'
      loop: "{{ temp.results }}"
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_loops.html#registering-variables-with-a-loop

Enregistrer le résultat d'une tâche dans une variable

- Certains modules génèrent directement un ensemble de données sans avoir besoin d'une boucle mais une boucle est nécessaire afin de pouvoir traiter ces données.

register-menage.yml

```
---
- name: Ménage de la salle de cours
  hosts: test
  become: true
  tasks:
    - name: Détection des fichiers LDIF
      ansible.builtin.find:
        paths: [ /home/user, /root, /tmp ]
        patterns: '*.ldif'
        recurse: true
      register: ldif
    - name: Suppression des fichiers LDIF
      ansible.builtin.file:
        path: "{{ item.path }}"
        state: absent
      loop: "{{ ldif.files }}"
```

Sommaire

Gestion des erreurs

- En cas d'erreur lors de l'application d'une tâche, le traitement s'arrête sur la machine correspondante (mais il continue sur les autres) en application du principe de précaution.
- Plusieurs approches sont possible pour éviter cela.

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_error_handling.html

ignore_errors

- Il est possible d'ignorer sciemment les erreurs lorsqu'on sait qu'une erreur dans une tâche ne compromet pas la suite de l'exécution.

ignore_errors.yml

```
---
- name: Ignorer les erreurs
  hosts: test
  become: true
  tasks:
    - name: Installation d'un paquet inexistant
      ansible.builtin.package:
        name: glub
        state: present
      ignore_errors: true
    - name: Mais on continue quand même
      ansible.builtin.debug:
        msg: nananère
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_error_handling.html#ignoring-failed-commands

Les blocs

Principe

- Les blocs permettent de grouper des tâches et de gérer les erreurs :
  - `block` tâches principales
  - `rescue` (facultatif) tâches à exécuter en cas d'erreur dans la partie `block`
  - `always` (facultatif) tâches à exécuter dans tous les cas

block.yml

```
---
- name: Blocs
  hosts: test
  become: true
  tasks:
    - name: Test de bloc
      block:
        - name: Installation d'un paquet inexistant
          ansible.builtin.package:
            name: glub
            state: present
      rescue:
        - name: Rescue
          ansible.builtin.debug:
            msg: 'cette partie est exécutée en cas d''erreur'
      always:
        - name: Always
          ansible.builtin.debug:
            msg: 'cette partie est toujours exécutée'
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_blocks.html

Les blocs

Message d'erreur personnalisé

- On peut utiliser un bloc afin d'afficher un message d'erreur personnalisé en cas d'échec de l'une des tâches du bloc.
- À cet effet, le module `ansible.builtin.fail` peut être utilisé dans la partie `rescue`. Ce module provoque toujours un échec donc Ansible arrêtera le traitement (après la partie `always`, si elle existe).
- Le module `ansible.builtin.debug` peut également être utilisé à la place du module `ansible.builtin.fail` si l'on souhaite que le traitement se poursuive.

rescue-msg_err_perso.yml

```
---
- name: Messages d'erreur personnalisés
  hosts: test
  tasks:
    - name: Qui ne tente rien...
      block:
        - name: On peut toujours essayer...
          ansible.builtin.file:
            path: /couic
            state: touch
        - name: Ne sera pas exécuté
          ansible.builtin.debug:
            msg: fin du bloc
      rescue:
        - name: Message d'erreur personnalisé
          ansible.builtin.fail:
            msg: c'est là qu'est l'os
    - name: Ne sera pas exécuté
      ansible.builtin.debug:
        msg: fin du livret
```

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/fail_module.html

Sommaire

Ansible vault

Principe

- Ansible vault est une fonctionnalité d'Ansible permettant de chiffrer les fichiers dont le contenu doit rester confidentiel.
- Ces fichiers doivent être gérés au moyen de la commande `ansible-vault`.
- Pour utiliser ensuite des fichiers chiffrés, la commande `ansible-playbook` doit être appelée avec l'une des options `--ask-vault-password` ou `--ask-vault-pass` (qu'on peut — enfin — abréger en `-J` à partir de la version 2.16.0 d'`ansible-core`) :

```
$ ansible-playbook -vJ vault.yml
Vault password:
[...]
```

https://docs.ansible.com/projects/ansible/latest/vault_guide/

Ansible vault

La commande `ansible-vault`

- La commande `ansible-vault` permet de gérer des fichiers chiffrés. Elle s'utilise selon la syntaxe suivante :

```
$ ansible-vault <action> [options] fichier.yml
```

- Certaines des actions de la commande `ansible-vault` exécutent un éditeur de texte. Celui-ci peut être personnalisé au moyen de la variable d'environnement `EDITOR`.

https://docs.ansible.com/projects/ansible/latest/cli/ansible-vault.html

Ansible vault

La commande `ansible-vault`

`create` création d'un nouveau fichier chiffré :

```
$ ansible-vault create vault.yml
New Vault password:
Confirm New Vault password:
```

`encrypt` chiffrement d'un fichier existant :

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


Ansible vault

La commande `ansible-vault`

`edit` édition d'un fichier chiffré :

```
$ ansible-vault edit vault.yml
Vault password:
```

`view` affichage d'un fichier chiffré :

```
$ ansible-vault view vault.yml
Vault password:
```

Sommaire

La commande ansible-lint

- La commande ansible-lint permet de vérifier le respect d'un certain nombre de bonnes pratiques dans des fichiers YAML pour Ansible.
- Elle s'utilise avec un ou plusieurs arguments, dont chacun peut être :
  - un livret, auquel cas les rôles utilisés par ce livret sont également vérifiés
  - un nom de rôle, auquel cas les tâches et les gestionnaires sont vérifiés
- Installation :
  - paquet ansible-lint sur toutes les distributions Linux
- Exemple :

```
$ ansible-lint livret.yml
[...]
name[casing]: All names should start with an uppercase letter. (warning)
livret.yml:2
[...]
```

https://docs.ansible.com/projects/lint/
https://www.redhat.com/sysadmin/ansible-lint-YAML
https://www.youtube.com/watch?v=8-M66pCzVTQ

Sommaire

Sommaire

Organisation de la CMDB

Principe

- Lorsque les machines à gérer sont nombreuses et la CMDB volumineuse, les rôles permettent de la rendre plus modulaire mais ils peuvent ne pas suffire.
- En cas d'utilisation de nombreuses variables spécifiques à certaines machines à gérer ou à certains groupes de machines, l'inventaire peut être complété par des répertoires `host_vars` et `group_vars`.
- Les rôles utilisés dans des conditions différentes peuvent être complétés par des répertoires `files`, `tasks`, `templates` et `vars` dans le répertoire contenant les livrets qui les utilisent.

```
<CMDB>/
├── ansible.cfg
├── inventaire.d
│   ├── group_vars/
│   └── host_vars/
├── livret/
│   ├── files/
│   ├── group_vars/
│   ├── host_vars/
│   ├── livret.yml
│   ├── tasks/
│   ├── templates/
│   └── vars/
└── roles/
```

https://docs.ansible.com/projects/ansible/latest/tips_tricks/sample_setup.html

Organisation de la CMDB

Répertoires `host_vars` et `group_vars`

- Des répertoires `host_vars` et `group_vars` peuvent exister dans deux répertoires (dans l'ordre de priorité) :
  1. celui contenant le livret
  2. celui contenant l'inventaire
- Ces répertoires contiennent un fichier (au format YAML, d'extension `.yml` ou `.yaml` ou sans extension) par machine à gérer ou par groupe.

```
<CMDB>/
├── inventaire.d/
│   ├── inventaire
│   ├── group_vars/
│   │   ├── groupe1.yml
│   │   └── ...
│   └── host_vars/
│       ├── machine1.example.com.yml
│       └── ...
└── livret-1/
    ├── group_vars/
    └── host_vars/
```

https://docs.ansible.com/projects/ansible/latest/inventory_guide/intro_inventory.html#organizing-host-and-group-variables

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

Organisation de la CMDB

Répertoires `files`, `tasks`, `templates` et `vars`

- Lorsqu'ils existent, Ansible utilise les répertoires `files`, `templates` et `vars` dans le répertoire contenant le livret (pas le répertoire courant) comme les répertoires de mêmes noms dans les rôles.
- Le fichier `vars/main.yml` n'est pas pris en compte automatiquement, il faut utiliser le module `ansible.builtin.include_vars`.
- Dans un livret faisant appel à un rôle :
  - les fichiers du rôle sont prioritaires sur ceux du répertoire contenant le livret
  - les fichiers du répertoire `tasks` sont utilisables seulement depuis le rôle

```
<CMDB>/
├── ansible.cfg
├── livret/
│   ├── files/
│   ├── livret.yml
│   ├── tasks/
│   ├── templates/
│   └── vars/
└── roles/
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbook_pathing.html#resolving-local-relative-paths

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

Sommaire

Les greffons (*plugins*)

Principe

- Ansible dispose d'une petite quinzaine de types de *greffons* (*plugins*), qui réalisent différents types de traitements.
- Les greffons, comme les modules, sont organisés sous forme de collections, même s'il y a beaucoup moins de greffons que de modules.

https://docs.ansible.com/projects/ansible/latest/module_plugin_guide/index.html
https://docs.ansible.com/projects/ansible/latest/collections/all_plugins.html

Les greffons (*plugins*)

Greffons de filtrage (*filter plugins*)

- Les *greffons de filtrage* (*filter plugins*) sont des filtres, spécifiques à Ansible, utilisables dans des expressions Jinja :

filter-ansible.builtin.human_readable.yml

```
---
- name: Greffon de filtrage ansible.builtin.human_readable
  hosts: test
  tasks:
    - name: Affichage brut
      ansible.builtin.debug:
        msg: 'point de montage {{ item.mount }} : {{ item.size_available }}'
      loop: "{{ ansible_mounts }}"
    - name: Affichage avec filtrage
      ansible.builtin.debug:
        msg: 'point de montage {{ item.mount }} : {{ item.size_available | ansible.builtin.human_readable }}'
      loop: "{{ ansible_mounts }}"
```


https://docs.ansible.com/projects/ansible/latest/plugins/filter.html
https://docs.ansible.com/projects/ansible/latest/collections/index_filter.html

Les greffons (*plugins*)

Greffons d'inventaire (*inventory plugins*)

- Les *greffons d'inventaire* (*inventory plugins*) permettent de générer un inventaire dynamique en interrogeant une source de données externe :

inventory-community.general.virtualbox.yml

```
---
plugin: community.general.virtualbox
running_only: true
groups:
  test: "'Ansible-' in inventory_hostname"
```

```
$ ansible-inventory -i inventory-community.general.virtualbox.yml --list
```

https://docs.ansible.com/projects/ansible/latest/plugins/inventory.html
https://docs.ansible.com/projects/ansible/latest/collections/index_inventory.html

Les greffons (*plugins*)

Greffons de consultation (*lookup plugins*)

- Les *greffons de consultation* (*lookup plugins*) permettent de consulter différentes sources de données dans des expressions Jinja :

lookup-ansible.builtin.file.yml

```
---
- name: Greffon de consultation ansible.builtin.file
  hosts: test
  become: true
  tasks:
    - name: Configuration de sshd
      ansible.builtin.blockinfile:
        path: /etc/ssh/sshd_config
        block: "{{ lookup ( 'ansible.builtin.file' , 'etc_ssh_sshd_config' ) }}" # noqa: jinja[spacing]
      notify:
        - sshd_reload
  handlers:
    - name: Rechargement de sshd
      ansible.builtin.service:
        name: sshd
        state: reloaded
      listen: sshd_reload
```

https://docs.ansible.com/projects/ansible/latest/plugins/lookup.html
https://docs.ansible.com/projects/ansible/latest/collections/index_lookup.html

Les greffons (*plugins*)

Greffons de test (*test plugins*)

- Les *greffons de test* (*test plugins*) sont des tests, spécifiques à Ansible, utilisables dans des expressions Jinja :

test-ansible.utils.ip_address.yml

```
---
- name: Greffon de test ansible.utils.ip_address
  hosts: test
  tasks:
    - name: Affichage
      ansible.builtin.debug:
        msg: "{{ item is ansible.utils.ip_address }}"
      loop:
        - 2001:db8:1234:5678:cafe:abba:d0d0:c0c0
        - 2001:db8:1234:5678:cafe:abba:d0d0:z0z0
        - 198.51.100.33
        - 198.51.100.1664
```

https://docs.ansible.com/projects/ansible/latest/plugins/test.html
https://docs.ansible.com/projects/ansible/latest/collections/index_test.html

Sommaire

Exécuter des tâches sur une autre machine

Principe

- Il peut s'avérer utile d'exécuter certaines tâches sur une autre machine (souvent, la machine de contrôle mais il peut s'agir de n'importe quelle machine).

Exécuter des tâches sur une autre machine

delegate_to

- Le mot clé `delegate_to` permet d'exécuter une tâche sur la machine indiquée :

delegate_to.yml

```
---
- name: Réplication
  hosts: postgresql_secours
  become: true
  tasks:
    - name: Fichier pg_hba.conf
      community.postgresql.postgresql_pg_hba:
        dest: /var/lib/pgsql/18/data/pg_hba.conf
        contype: host
        databases: replication
        users: repl
        address: "{{ item }}/{{ '128' if item is ansible.utils.ipv6_address else '32' }}"
        method: scram-sha-256
      when: item is not ansible.utils.subnet_of 'fe80::/10'
      loop: "{{ [ ansible_all_ipv4_addresses , ansible_all_ipv6_addresses ] | ansible.builtin.flatten }}"
      delegate_to: postgresql_principal
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_delegation.html

Exécuter des tâches sur une autre machine

local_action

- Le mot clé local_action est équivalent à delegate_to: localhost :

local_action.yml

```
---
- name: Mot clé local_action
  hosts: test
  become: true
  tasks:
# [...]
    - name: Téléchargement du fichier sha256sums.txt
      become: false
      run_once: true
      local_action:
        module: ansible.builtin.get_url
        url: https://www.example.com/sha256sums.txt
        dest: /tmp/sha256sums.txt
        mode: '600'
```

https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_delegation.html

https://docs.ansible.com/projects/ansible/latest/inventory/implicit_localhost.html

Exécuter des tâches sur une autre machine

localhost implicite

- Il est bien entendu toujours possible d'effectuer des tâches directement sur la machine localhost implicite :

localhost.yml

```
---

- name: Machines à gérer
  hosts: test
  tasks:
    - name: Affichage
      ansible.builtin.debug:
        var: ansible_system

- name: Machine de contrôle
  hosts: localhost
  tasks:
    - name: Affichage
      ansible.builtin.debug:
        var: ansible_system
```

https://docs.ansible.com/projects/ansible/latest/inventory/implicit_localhost.html

Sommaire

Informations factuelles personnalisées

- Il est possible d'enrichir les informations factuelles (*facts*) par des informations factuelles personnalisées (*custom facts*).
- Les informations factuelles personnalisées figurent dans des fichiers présents sur les machines à gérer :
  - dans le répertoire `/etc/ansible/facts.d`
  - ou dans un autre répertoire, spécifié dans la CMDB par le paramètre `fact_path`
- Les fichiers conternant les informations factuelles personnalisées doivent avoir l'extension `.fact` et :
  - ne pas être exécutables et dans l'un des formats INI ou JSON
  - ou être exécutables et afficher sur leur sortie standard des données au format JSON
- Les fichiers contenant les informations factuelles personnalisées doivent être accessibles — en lecture et, s'il y a lieu, en exécution — par l'utilisateur sous l'identité duquel fonctionne Ansible sur les machines à gérer.

`https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_vars_facts.html#adding-custom-facts`

Informations factuelles personnalisées

/etc/ansible/facts.d/ini.fact

```
[sgbd]
logiciel = PostgreSQL
version = 17.5
```

/etc/ansible/facts.d/json.fact

```
{
        "logiciel": "OpenLDAP" ,
        "version": "2.6.10"
}
```

/etc/ansible/facts.d/exec.fact

```
#! /bin/sh

cat << FIN
{
        "uname": "$(uname)"
}
FIN
```


Informations factuelles personnalisées

- Les variables contenant les informations factuelles se trouvent dans la correspondance ansible_local :

custom_facts.yml

```
---
- name: Infos factuelles perso
  hosts: test
  # fact_path: /[...]
  tasks:
    - name: Affichage
      ansible.builtin.debug:
        var: ansible_local
```

```
$ ansible-playbook -v custom_facts.yml
[...]
TASK [Affichage] ****************************
ok: [test1.example.com] => {
    "ansible_local": {
        "exec": {
            "uname": "NetBSD"
        },
        "ini": {
            "sgbd": {
                "logiciel": "PostgreSQL",
                "version": "17.5"
            }
        },
        "json": {
            "logiciel": "OpenLDAP",
            "version": "2.6.10"
        }
    }
}
[...]
```

Le module ansible.builtin.set_fact

- Le module ansible.builtin.set_fact permet de définir une ou plusieurs variables en cours d'exécution, généralement à partir de données provenant d'une origine extérieure ou d'un traitement quelconque.
- En pratique, les variables sont évaluées lors de leur utilisation et pas lors de leur définition et il est donc possible de les définir de façon habituelle dans la très grande majorité des cas.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/set_fact_module.html

Le module ansible.builtin.set_fact

module-ansible.builtin.set_fact.yml

```
---
- name: Définition de variables
  hosts: test
  tasks:
    - name: Lecture fichier CSV
      community.general.read_csv:
        path: /tmp/set_fact.csv
        fieldnames: [ un , deux ]
        delimiter: ','
      register: csv
    - name: Variables
      ansible.builtin.set_fact:
        un: "{{ csv.list.0.un }}"
        deux: "{{ csv.list.0.deux }}"
```

module-ansible.builtin.set_fact-ou_pas.yml

```
---
- name: Définition de variables
  hosts: test
  vars:
    un: "{{ csv.list.0.un }}"
    deux: "{{ csv.list.0.deux }}"
  tasks:
    - name: Lecture fichier CSV
      community.general.read_csv:
        path: /tmp/set_fact.csv
        fieldnames: [ un , deux ]
        delimiter: ','
      register: csv
    - name: Affichage de la variable un
      ansible.builtin.debug:
        var: un
    - name: Affichage de la variable deux
      ansible.builtin.debug:
        var: deux
```

Sommaire

Actions asynchrones

- En temps normal, lorsqu'une tâche est en cours d'exécution sur une machine, Ansible attend qu'elle se finisse avant de poursuivre.
- Une *action asynchrone* est une tâche pour laquelle il est demandé à Ansible de ne pas attendre qu'elle se finisse avant de poursuivre.

`https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_async.html`

Actions asynchrones

async-poll.yml

```
---
- name: Limitation de durée
  hosts: test
  tasks:
    - name: On fonce droit dans le mur en klaxonnant
      ansible.builtin.shell:
        cmd: for i in `seq 30` ; do date >> /tmp/async ; sleep 1 ; done
      async: 10
      poll: 1
```

- Le paramètre `async` indique la durée maximale (exprimée en secondes) de la tâche. Si elle ne se finit pas avant, elle est arrêtée de force à l'issue de cette durée, avec un état d'échec.
- Le paramètre `poll` indique la fréquence (exprimée en secondes) à laquelle vérifier l'état de la tâche.

Actions asynchrones

- Si le paramètre `poll` n'est pas utilisé, Ansible utilise la valeur du paramètre `poll_interval` du fichier de paramétrage ansible.cfg :

ansible.cfg

```
[defaults]
poll_interval = 1
```

La valeur par défaut du paramètre `poll_interval` est 15.

https://docs.ansible.com/projects/ansible/latest/reference_appendices/config.html#default-poll-interval

Actions asynchrones

- Lorsque le paramètre `poll` a pour valeur 0, la tâche est exécutée en arrière-plan et Ansible poursuit sans attendre qu'elle se finisse.

async-poll_0.yml

```
---
- name: Action asynchrone
  hosts: test
  tasks:
    - name: Tâche asynchrone
      ansible.builtin.shell:
        cmd: for i in `seq 10` ; do date >> /tmp/async ; sleep 1 ; done
      async: 30
      poll: 0
      register: asynchrone
    - name: Affichage
      ansible.builtin.debug:
        var: asynchrone
# ici, il faudrait faire quelque chose d'intelligent
    - name: État d'avancement
      ansible.builtin.async_status:
        jid: "{{ asynchrone.ansible_job_id }}"
        # mode: status
      register: etat
      until: etat.finished
      delay: 1
      retries: 50
    - name: Ménage
      ansible.builtin.async_status:
        jid: "{{ asynchrone.ansible_job_id }}"
        mode: cleanup
```


Actions asynchrones

- Le module `ansible.builtin.async_status` permet de gérer les tâches asynchrones.
- Son paramètre `jid` (*job identifier*) indique l'identifiant de la tâche concernée.
- Son paramètre `mode` peut prendre les valeurs :
  - `status` pour obtenir l'état de la tâche (c'est la valeur par défaut en l'absence du paramètre `mode`)
  - `cleanup` pour supprimer le fichier temporaire contenant l'état de la tâche dans le répertoire `~/.ansible_async` sur les machines à gérer

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/async_status_module.html

Sommaire

YAML

Ancres et alias

- Le format YAML permet de réutiliser des données au sein d'un document grâce à des *ancres* (*anchors*) et des *alias* (*aliases*) :
  - une ancre consiste en une esperluette & immédiatement suivie d'un identifiant arbitraire et permet d'identifier les données qui la suivent ;
  - un alias consiste en un astérisque * immédiatement suivi d'un identifiant déjà défini au moyen d'une ancre et permet de réutiliser les données identifiées par l'ancre.

YAML-ancre-alias.yml

```
---
- name: YAML - ancre et alias
  hosts: test
  tasks:
    - name: Ficher de test
      ansible.builtin.copy:
        content: |
          1. ligne une
          2. ligne deux
        dest: &TEST /tmp/test
    - name: Ajout d'une ligne
      ansible.builtin.lineinfile:
        path: *TEST
        line: 3. ligne trois
        regexp: ^3\.
```


https://docs.ansible.com/projects/ansible/latest/playbook_guide/playbooks_advanced_syntax.html#yaml-anchors-and-aliases-sharing-variable-values

https://yaml.org/spec/1.2.2/#example-node-for-sammy-sosa-appears-twice-in-this-document

https://yaml.org/spec/1.2.2/#anchors-and-aliases

Sommaire

Les modules

- Les *modules* sont les composants d'Ansible qui effectuent les actions sur les machines distantes.
- Ansible propose de très nombreux modules.
- Depuis la version 2.9 d'Ansible, les modules ont été réorganisés sous forme de *collections*. La collection `ansible.builtin` regroupe les modules les plus utilisés. Il est désormais recommandé d'écrire les noms de modules dans la CMDB au moyen de leur FQCN (*fully qualified collection name*).
- Il n'est pas utile détailler tous les modules ici — la documentation d'Ansible est très claire et il est préférable de toujours s'y reporter pour avoir une description à jour des modules — mais nous allons en survoler quelques uns.

https://docs.ansible.com/projects/ansible/latest/module_plugin_guide/index.html
https://docs.ansible.com/projects/ansible/latest/collections/
https://docs.ansible.com/projects/ansible/latest/collections/index_module.html

Le module ansible.builtin.apt

module-ansible.builtin.apt.yml

```
---
- name: Module ansible.builtin.apt
  hosts: test
  become: true
  tasks:
    - name: Install. nginx
      ansible.builtin.apt:
        name: nginx
        state: present
        update_cache: true
```

- Le module ansible.builtin.apt permet d'installer, de mettre à jour ou de supprimer des paquets logiciels au moyen du gestionnaire de paquets APT (donc il ne fonctionne que sur les distributions Linux utilisant ce gestionnaire de paquets, comme Debian et Ubuntu).
- Par rapport au module générique ansible.builtin.package, le module ansible.builtin.apt dispose de paramètres supplémentaires.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/apt_module.html

https://fr.wikipedia.org/wiki/Advanced_Packaging_Tool

Le module ansible.builtin.assert

module-ansible.builtin.assert.yml

```
---
- name: Module ansible.builtin.assert
  hosts: test
  become: true
  tasks:
    - name: Vérification du système hôte
      ansible.builtin.assert:
        that:
          - ansible_os_family == 'RedHat'
          - ansible_distribution_major_version == '7'
        fail_msg: >-
          Il n'est possible d'effectuer ceci que sur
          une distribution Linux de la famille Red Hat
          et de version 7. Or la distribution est :
          {{ ansible_os_family }} version
          {{ ansible_distribution_major_version }}.
```

- Le module ansible.builtin.assert s'assure qu'une ou plusieurs expressions (avec la même syntaxe que l'instruction when) sont vérifiées et échoue dans le cas contraire.
- Les conditions de la séquence suivant le paramètre that doivent toutes être vérifiées pour que le module ansible.builtin.assert réussisse.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/assert_module.html

Le module ansible.builtin.command

module-ansible.builtin.command.yml

```
---
- name: Module ansible.builtin.command
  hosts: test
  become: true
  tasks:
    - name: Création clé rndc
      ansible.builtin.command:
        cmd: rndc-confgen -a
        creates: /etc/rndc.key
```

- Le module ansible.builtin.command permet d'exécuter une commande en spécifiant ses conditions d'exécution (répertoire courant, exécution ou pas en fonction de la présence ou de l'absence d'un fichier).
- L'idempotence doit être gérée manuellement (grâce aux paramètres creates ou removes) et elle est imparfaite.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/command_module.html

Le module ansible.builtin.copy

module-ansible.builtin.copy.yml

```
---
- name: Module ansible.builtin.copy
  hosts: test
  become: true
  tasks:
    - name: Config. syslog
      ansible.builtin.copy:
        src: openldap.conf
        dest: /etc/rsyslog.d/openldap.conf
        owner: root
        group: root
        mode: '444'
      notify:
        - restart rsyslog
  handlers:
    - name: restart rsyslog # noqa: name[casing]
      ansible.builtin.service:
        name: rsyslog
        state: restarted
```

- Le module ansible.builtin.copy recopie à l'identique un fichier depuis la machine de contrôle vers les machines à gérer.
- La copie n'est effectuée que si l'état du fichier (contenu, caractéristiques) est différent de celui spécifié.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/copy_module.html

Le module ansible.builtin.cron

module-ansible.builtin.cron.yml

```
---
- name: Module ansible.builtin.cron
  hosts: test
  become: true
  tasks:
    - name: Sauvegarde
      ansible.builtin.cron:
        name: sauvegarde
        cron_file: svg
        minute: 0
        hour: 1
        weekday: 7
        user: root
        job: /usr/local/bin/svg
```

- Le module ansible.builtin.cron permet de manipuler des tâches à faire exécuter par cron.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/copy_module.html

Le module ansible.builtin.debconf

module-ansible.builtin.debconf.yml

```
---
- name: Module ansible.builtin.debconf
  hosts: test
  become: true
  tasks:
    - name: Configuration de slapd
      ansible.builtin.debconf:
        name: slapd
        question: slapd/domain
        value: example.com
        vtype: string
      notify:
        - dpkg_reconfigure
  handlers:
    - name: Exécution de dpkg-reconfigure
      ansible.builtin.command:
        cmd: dpkg-reconfigure -f noninteractive slapd
      listen: dpkg_reconfigure
```

- Le module ansible.builtin.debconf permet d'interagir avec le mécanisme de configuration de paquets debconf (donc il ne fonctionne que sur les distributions Linux utilisant ce mécanisme, comme Debian et Ubuntu).
- Si la configuration est modifiée, il faut exécuter la commande dpkg-reconfigure dans un gestionnaire afin que les modifications soient effectives.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/debconf_module.html

https://en.wikipedia.org/wiki/Debian_configuration_system

Le module ansible.builtin.debug

module-ansible.builtin.debug.yml

```
---
- name: Module ansible.builtin.debug
  hosts: test
  tasks:
    - name: Affichage message
      ansible.builtin.debug:
        msg: 'ceci est un test'
    - name: Affichage variable
      ansible.builtin.debug:
        var: ansible_date_time
```

- Le module ansible.builtin.debug permet d'afficher un message (paramètre msg) ou le contenu d'une variable (paramètre var).
- Les paramètres msg et var sont mutuellement exclusifs.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/debug_module.html

Le module ansible.builtin.dnf

module-ansible.builtin.dnf.yml

```
---
- name: Module ansible.builtin.dnf
  hosts: test
  become: true
  tasks:
    - name: Install. nginx
      ansible.builtin.dnf:
        name: nginx
        state: present
        update_cache: true
```

- Le module ansible.builtin.dnf permet d'installer, de mettre à jour ou de supprimer des paquets logiciels au moyen du gestionnaire de paquets DNF (donc il ne fonctionne que sur les distributions Linux utilisant ce gestionnaire de paquets, comme *Red Hat Enterprise Linux* et Fedora).
- Par rapport au module générique ansible.builtin.package, le module ansible.builtin.dnf dispose de paramètres supplémentaires.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/dnf_module.html

https://fr.wikipedia.org/wiki/Dandified_Yum

Le module ansible.builtin.file

module-ansible.builtin.file.yml

```
---
- name: Module ansible.builtin.file
  hosts: test
  become: true
  tasks:
    - name: Vérif. /srv/www
      ansible.builtin.file:
        path: /srv/www
        state: directory
        owner: root
        group: root
        mode: '755'
```

- Le module ansible.builtin.file permet de positionner divers attributs sur des fichiers.
- Le module ansible.builtin.file partage de nombreuses options avec les modules ansible.builtin.copy et ansible.builtin.template.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/file_module.html

Le module ansible.builtin.get_url

module-ansible.builtin.get_url.yml

```
---
- name: Module ansible.builtin.get_url
  hosts: test
  become: true
  tasks:
    - name: Téléchargement
      ansible.builtin.get_url:
        url: http://www.example.com/test.tar.gz
        dest: /tmp/test.tar.gz
        owner: root
        group: root
        mode: '644'
```

- Le module ansible.builtin.get_url télécharge un fichier depuis les machines à gérer au moyen des protocoles HTTP, HTTPS ou FTP.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/get_url_module.html

Le module ansible.builtin.git

module-ansible.builtin.git.yml

```
---
- name: Module ansible.builtin.git
  hosts: test
  become: true
  tasks:
    - name: Paquet git
      ansible.builtin.package:
        name: git
        state: present
    - name: Dépôt ansible-lint
      ansible.builtin.git:
        repo: https://github.com/ansible/ansible-lint.git
        dest: /usr/local/src/ansible-lint
```

- Le module ansible.builtin.git permet de cloner des dépôts Git.
- Pour cela, la commande git doit être installée sur les machines à gérer.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/git_module.html
https://git-scm.com/
https://fr.wikipedia.org/wiki/Git

Le module ansible.builtin.group

module-ansible.builtin.group.yml

```
---
- name: Module ansible.builtin.group
  hosts: test
  become: true
  tasks:
    - name: Création groupe
      ansible.builtin.group:
        name: ansible
        # gid: 1664
        system: true
```

- Le module ansible.builtin.group permet la gestion des groupes.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/group_module.html

Le module ansible.builtin.lineinfile

module-ansible.builtin.lineinfile.yml

```
---
- name: Module ansible.builtin.lineinfile
  hosts: test
  become: true
  tasks:
    - name: Modif. /etc/hosts
      ansible.builtin.lineinfile:
        path: /etc/hosts
        line: '198.51.100.33    test.example.com'
        regexp: '^198\.51\.100\.33'
```

- Le module ansible.builtin.lineinfile ajoute ou remplace une ligne dans un fichier.
- L'expression rationnelle spécifiée par le paramètre regexp permet d'identifier une ligne similaire afin de la remplacer.
- Il existe aussi un module ansible.builtin.blockinfile, qui permet de manipuler un bloc (plusieurs lignes) de texte dans un fichier.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/lineinfile_module.html

Le module ansible.builtin.package

module-ansible.builtin.package.yml

```
---
- name: Module ansible.builtin.package
  hosts: test
  become: true
  tasks:
    - name: Install. nginx
      ansible.builtin.package:
        name: nginx
        state: present
```

- Le module ansible.builtin.package est un module générique permettant d'installer, de mettre à jour ou de supprimer des paquets logiciels.
- Il existe également des modules ne fonctionnant que sur certaines distributions Linux (ansible.builtin.apt, ansible.builtin.dnf, etc.) mais disposant de plus d'options.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/package_module.html

Le module ansible.builtin.replace

module-ansible.builtin.replace.yml

```
---
- name: Module ansible.builtin.replace
  hosts: test
  become: true
  tasks:
    - name: Activation de postscreen
      ansible.builtin.replace:
        path: /etc/postfix/master.cf
        regexp: '^#(smtp.*postscreen)$'
        replace: '\1'
```

- Le module ansible.builtin.replace effectue des remplacements dans un fichier au moyen d'une expression rationnelle avec laquelle on peut mémoriser des chaînes de caractères — entourées par des parenthèses — dans la ligne d'origine puis les réutiliser dans la nouvelle ligne : \1 pour le contenu du premier couple de parenthèses, \2 pour celui du deuxième, etc.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/replace_module.html

Le module ansible.builtin.script

module-ansible.builtin.script.yml

```
---
- name: Module ansible.builtin.script
  hosts: test
  become: true
  tasks:
    - name: Exécution du script
      ansible.builtin.script:
        cmd: script.sh /tmp/toto
        creates: /tmp/toto
```

- Le module ansible.builtin.script transfère un script depuis la machine de contrôle vers les machines à gérer avant de l'exécuter avec un interpréteur de commandes.
- Idempotence imparfaite.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/script_module.html

Le module ansible.builtin.service

module-ansible.builtin.service.yml

```
---
- name: Module ansible.builtin.service
  hosts: test
  become: true
  tasks:
    - name: Installation de nginx
      ansible.builtin.package:
        name: nginx
        state: present
      notify:
        - enable_start_nginx
  handlers:
    - name: enable_start_nginx # noqa: name[casing]
      ansible.builtin.service:
        name: nginx
        enabled: true
        state: started
```

- Le module ansible.builtin.service est un module générique permettant la gestion des daemons (activation, démarrage, redémarrage, etc.).
- Le module ansible.builtin.service est souvent utilisé dans des gestionnaires.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/service_module.html

Le module ansible.builtin.shell

module-ansible.builtin.shell.yml

```
---
- name: Module ansible.builtin.shell
  hosts: test
  become: true
  tasks:
    - name: Configuration filtrage IP
      ansible.builtin.shell:
        cmd: iptables-save > iptables
        chdir: /etc/sysconfig
        creates: /etc/sysconfig/iptables
```

- Le module ansible.builtin.shell permet d'exécuter une commande mais, à la différence du module ansible.builtin.command, la commande est exécutée par un interpréteur de commandes (/bin/sh), ce qui permet d'utiliser ses syntaxes spécifiques (métacaractères, redirections, tuyaux, substitutions, etc.).
- Idempotence imparfaite.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/shell_module.html

Le module ansible.builtin.systemd_service

module-ansible.builtin.systemd_service.yml

```
---
- name: Module ansible.builtin.systemd_service
  hosts: test
  become: true
  tasks:
    - name: Service bizarre
      ansible.builtin.systemd_service:
        name: bizarre
        daemon_reload: true
        state: started
        enabled: true
```

- Le module ansible.builtin.systemd_service permet la gestion des unités de systemd.
- Par rapport au module générique ansible.builtin.service, le module ansible.builtin.systemd_service dispose de paramètres supplémentaires, spécifiques à systemd.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/systemd_service_module.html

https://systemd.io/

Le module ansible.builtin.template

module-ansible.builtin.template.yml

```
---
- name: Module ansible.builtin.template
  hosts: test
  become: true
  tasks:
    - name: Configuration hôte virtuel
      ansible.builtin.template:
        src: nginx-vhost.conf.j2
        dest: /etc/nginx/conf.d/{{ inventory_hostname }}.conf
        owner: root
        group: root
        mode: '444'
```

- Le module ansible.builtin.template fonctionne sur le même principe que le module ansible.builtin.copy mais il effectue des traitements dans le fichier en utilisant Jinja.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/template_module.html

Le module ansible.builtin.user

module-ansible.builtin.user.yml

```
---
- name: Module ansible.builtin.user
  hosts: test
  become: true
  tasks:
    - name: Création utilisateur
      ansible.builtin.user:
        name: jlapin
        # uid: 1664
        group: users
        comment: 'Jojo Lapin'
```

- Le module ansible.builtin.user permet la gestion des comptes utilisateurs.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/builtin/user_module.html
https://www.youtube.com/watch?v=UEfYIYVe9yw

Le module ansible.posix.patch

module-ansible.posix.patch.yml

```
---
- name: Module ansible.posix.patch
  hosts: test
  become: true
  tasks:
    - name: Application du correctif
      ansible.posix.patch:
        src: hosts.patch
        dest: /etc/hosts
```

- Le module ansible.posix.patch permet d'appliquer un correctif (*patch*) à partir d'un fichier stocké sur la machine de contrôle.

https://docs.ansible.com/projects/ansible/latest/collections/ansible/posix/patch_module.html
https://fr.wikipedia.org/wiki/Patch_(informatique)
http://savannah.gnu.org/projects/patch/
https://en.wikipedia.org/wiki/Patch_(Unix)

Le module community.general.ini_file

module-community.general.ini_file.yml

```
---
- name: Module community.general.ini_file
  hosts: test
  become: true
  tasks:
    - name: Personnalisation ansible.cfg
      community.general.ini_file:
        path: /srv/ansible/ansible.cfg
        section: defaults
        option: forks
        value: 50
        owner: ansible
        group: ansible
        mode: '444'
```

- Le module community.general.ini_file permet de gérer des fichiers au format INI.

https://docs.ansible.com/projects/ansible/latest/collections/community/general/ini_file_module.html

https://fr.wikipedia.org/wiki/Fichier_INI

La commande ansible-doc

- La commande ansible-doc permet d'afficher la documentation du ou des modules en arguments :

```
$ ansible-doc ansible.builtin.lineinfile
```

ou avec l'ancien nom du module :

```
$ ansible-doc lineinfile
```

- L'option -l (ou --list) affiche la liste des modules disponibles :

```
$ ansible-doc -l
```

https://docs.ansible.com/projects/ansible/latest/cli/ansible-doc.html

Sommaire

Bibliographie

[1] Russ McKendrick
Learn Ansible — Automate your cloud infrastructure, security configuration, and application deployment with Ansible
Anglais
2e édition
Packt Publishing, mai 2024
url : https://www.packtpub.com/en-fr/product/learn-ansible-9781835082171

[2] James Freeman, Fabio Alessandro Locati et Daniel Oh
Practical Ansible
Anglais
2e édition
Packt Publishing, septembre 2023
url : https://www.packtpub.com/en-fr/product/practical-ansible-9781805129974

Bibliographie

[3] Yannig Perré
Ansible — Gérez la configuration de vos serveurs et le déploiement de vos applications
3e édition
Éditions ENI, mai 2023
url : https://www.editions-eni.fr/livre/ansible-gerez-la-configuration-de-vos-serveurs-et-le-deploiement-de-vos-applications-3e-edition-9782409039720

[4] Gineesh Madapparambath
Ansible for Real-Life Automation
Anglais
Packt Publishing, septembre 2022
url : https://www.packtpub.com/en-fr/product/ansible-for-real-life-automation-9781803235417

Bibliographie

[5] Bas Meijer, Lorin Hochstein et René Moser
Ansible Up & Running — Automating Configuration Management and Deployment the Easy Way
Anglais
3e édition
O'Reilly Media, juillet 2022
url : http://www.ansiblebook.com/

[6] James Freeman et Jesse Keating
Mastering Ansible
Anglais
4e édition
Packt Publishing, décembre 2021
url : https://www.packtpub.com/en-fr/product/mastering-ansible-4th-edition-9781801818780

Bibliographie

[7] Josh Duffney
become Ansible — Zero to Production-Ready
Anglais
Octobre 2020
url : https://joshduffney.gumroad.com/l/become-ansible

[8] Philippe Pinchon
Red Hat Ansible Engine — Gérez l'automatisation de vos configurations Linux
Éditions ENI, octobre 2020
url :
https://www.editions-eni.fr/livre/red-hat-ansible-engine-gerez-l-automatisation-de-vos-configurations-linux-9782409027291

Bibliographie

[9] Jeff Geerling
Ansible for DevOps — Server and configuration management for humans
Anglais
2e édition
5 août 2020
url : https://www.ansiblefordevops.com/

[10] Fabio Alessandro Locati
Learning Ansible 2.7
Anglais
3e édition
Packt Publishing, avril 2019
url : https://www.packtpub.com/en-fr/product/learning-ansible-27-9781789950007

Bibliographie

[11] Mohamed Alibi
Ansible Quick Start Guide
Anglais
Packt Publishing, septembre 2018
url : https://www.packtpub.com/en-fr/product/ansible-quick-start-guide-9781789538731

[12] Aditya Patawari et Vikas Aggarwal
Ansible 2 Cloud Automation Cookbook
Anglais
Packt Publishing, février 2018
url : https://www.packtpub.com/en-fr/product/ansible-2-cloud-automation-cookbook-9781788298773

Bibliographie

[13] Akash Mahajan et Madhu Akula
Security Automation with Ansible 2
Anglais
Packt Publishing, décembre 2017
url : https://www.packtpub.com/en-fr/product/security-automation-with-ansible-2-9781788398725

[14] Jonathan McAllister
Implementing DevOps with Ansible 2
Anglais
Packt Publishing, juillet 2017
url : https://www.packtpub.com/en-fr/product/implementing-devops-with-ansible-2-9781787126510

Bibliographie

[15] Gourav Shah
Ansible Playbook Essentials
Anglais
Packt Publishing, août 2015
url : https://www.packtpub.com/en-fr/product/ansible-playbook-essentials-9781784395612