Support pédagogique Ansible - Modules

Les modules

Créer un module personnalisé

👨‍🏫 Objectif : Comprendre et développer un module Ansible en Python.

📦 Qu’est-ce qu’un module Ansible ?

- Un **module** est une unité d'exécution.
- C'est le composant qui exécute une tâche dans un playbook.
- Ansible fournit des **modules natifs**, mais on peut créer les siens.

🤔 Pourquoi créer un module personnalisé ?

- Besoins métiers spécifiques
- Intégration avec une API interne
- Simplifier la logique dans les playbooks
- Partage dans une collection Ansible

⚙️ Structure d'un module personnalisé

Un module est un fichier Python qui utilise l'objet `AnsibleModule`.

Fichier de base :

```
library/
└── mon_module.py
```

🔧 Exemple minimal de module

```
# mkdir /root/ansible-practice/modules && cd /root/ansible-practice/modules

# mkdir library/
```

- On crée le module `mon_module.py`

```
#!/usr/bin/python

from ansible.module_utils.basic import AnsibleModule

def main():
    module = AnsibleModule(argument_spec={})
    module.exit_json(changed=False, msg="Hello world!")

if __name__ == '__main__':
    main()
```

🔍 Exploration : `AnsibleModule` en détail

L'objet `AnsibleModule` est fourni par `ansible.module_utils.basic`.

Fonctions clés :

- `params` : récupère les paramètres
- `check_mode` : indique si `--check` est actif
- `exit_json()` : sortie normale
- `fail_json()` : sortie d'erreur

🧪 Tester un module localement

```
# ANSIBLE_LIBRARY=./library ansible localhost -m mon_module -i localhost, -c local
```

- `-m` : nom du module
- `-i` : inventaire inline
- `-c` : méthode de connexion, ici "local" pour exécuter la commande sans SSH
- `ANSIBLE_LIBRARY` : variable d'environnement pour indiquer où chercher les modules personnalisés (ici le dossier "./library")

💬 Ajouter des paramètres

```
argument_spec = {
    "name": {"type": "str", "required": True}
}
```

Permet de récupérer une variable dans le playbook :

```
- name: Test module
  mon_module:
    name: "Ansible"
```

📤 Retourner des données

```
module.exit_json(changed=True, message="OK")
```

- `changed` : booléen
- `msg`, `message`, etc. → variables retournées dans le playbook

❗ Gérer les erreurs

```
module.fail_json(msg="Erreur critique")
```

🧱 Ajouter de la logique métier

```
if name == "admin":
    module.fail_json(msg="Nom réservé !")
```

🧠 Exemple complet

- On crée un deuxieme module `mon_deuxieme_module.py` dans `library`

```
#!/usr/bin/python

from ansible.module_utils.basic import AnsibleModule

def main():
    module = AnsibleModule(
        argument_spec={"name": {"type": "str", "required": True}}
    )

    name = module.params["name"]

    if name == "admin":
        module.fail_json(msg="Nom réservé !")

    module.exit_json(changed=True, msg=f"Bonjour {name}")

if __name__ == '__main__':
    main()
```

🔍 Utilisation dans un playbook

- On crée le playbook `playbook_module.yml`

```yaml
- name: Exemple Ansible module personnalisé
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Dire bonjour
      mon_deuxieme_module:
        name: Jean
      register: result

    - name: Afficher le message
      debug:
        msg: "{{ result.msg }}"
```

🔍 Utilisation dans un playbook

```
PLAY [Exemple Ansible module personnalisé] ****************************

TASK [Dire bonjour] ***************************
changed: [localhost]

TASK [Afficher le message] ****************************
ok: [localhost] => {
    "msg": "Bonjour Jean"
}

PLAY RECAP ****************************
localhost                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

🧪 Tester avec `--check` et `--diff`

Utilisez vos modules avec :

- `--check` pour la simulation
- `--diff` pour voir les changements

🔄 Module idempotent

Bon module Ansible = **pas de changement si rien ne change**

```
if already_exists(name):
    module.exit_json(changed=False, msg="Déjà en place")
```

Un exemple

- Parcourir et comprendre `mon_troisieme_module.py`
- Créer un playbook `playbook_troisieme_module.yml`
- Lancer le playbook

- `playbook_troisieme_module.yml`

```
- name: Test du module personnalisé mon_module
hosts: localhost
connection: local
gather_facts: false

tasks:

    - name: Utiliser mon_troisieme_module pour dire bonjour
    mon_troisieme_module:
        name: Alex
        path: /tmp/bonjour_ansible-2.txt
    register: result

    - name: Afficher le message du module
    debug:
        msg: "{{ result.message }}"

    - name: Afficher le diff (si présent)
    debug:
        var: result.diff
    when: result.diff is defined
```

Un exemple

- `# ANSIBLE_LIBRARY=./library ansible-doc mon_troisieme_module`
- `# ansible-playbook playbook_troisieme_module.yml`
- Modifier le `name:` **Pierre** dans le playbook
- `ansible-playbook playbook_troisieme_module.yml --check --diff`
- `ansible-playbook playbook_troisieme_module.yml --diff`
- `ansible-playbook playbook_troisieme_module.yml`

🚧 Limites des modules personnalisés

- Maintenance manuelle
- Nécessitent des tests
- Peuvent être plus lents sans optimisation

Ansible - Sanity Tests

Vérification des modules et collections

Objectifs

- Comprendre le rôle des tests sanity
- Identifier les erreurs courantes
- Apprendre à lancer les tests sur un module personnalisé

Qu'est-ce que `ansible-test` ?

- Outil CLI fourni par Ansible
- Utilisé pour tester la validité des modules/collections
- Exécute plusieurs types de tests : syntaxe, style, doc, compatibilité

Commande de base

```
ansible-test sanity --local
```

- À exécuter dans le dossier racine d'une **collection** (contenant `galaxy.yml`)

Structure requise

```
ansible_collections/
└── <namespace>/
    └── <collection>/
        ├── galaxy.yml
        └── plugins/
            └── modules/
                └── mon_module.py
```

Tests les plus courants

- `validate-modules` : vérifie `DOCUMENTATION`, `EXAMPLES`, etc.
- `pep8` : style Python
- `pylint` : analyse statique
- `yamllint` : style YAML
- `runtime-metadata` : métadonnées dans galaxy.yml

Exemple de sortie

```
FAILED: validate-modules
Erreur: missing author
FAILED: pep8
Erreur: ligne trop longue
```

Exécuter un test ciblé

```
ansible-test sanity --test validate-modules --local
```

✅ Utile pour corriger un seul aspect à la fois

- Pour voir tous les tests

```
# ansible-test sanity --list-tests
```

En résumé ✅

- `ansible-test sanity` = outil essentiel pour publier du code propre
- Il valide style, structure, doc et compatibilité Python
- À intégrer dans tout développement de module ou collection

🧪 Testons notre mdoule !

Utiliser `ansible-test` pour valider notre module personnalisé :

- On va mettre le module dans une arborescence de collection

```
# mkdir /root/ansible-practice/projet00/collections/ansible_collections/coll/test/plugins/modules/
```

```
# cp /root/ansible-practice/modules/library/mon_troisieme_module.py \
/root/ansible-practice/projet00/collections/ansible_collections/coll/test/plugins/modules/
```

```
# cd /root/ansible-practice/projet00/collections/ansible_collections/coll/test
```

```
# ansible-test sanity --test validate-modules --local
```

```
Running sanity test "validate-modules"
ERROR: Found 4 validate-modules issue(s) which need to be resolved:
ERROR: plugins/modules/mon_troisieme_module.py:0:0: invalid-documentation: DOCUMENTATION.
author: Invalid author for dictionary value @ data['author']. Got ['Moi !!']
ERROR: plugins/modules/mon_troisieme_module.py:0:0: invalid-documentation: DOCUMENTATION.
module: not a valid value for dictionary value @ data['module']. Got 'mon_module'
ERROR: plugins/modules/mon_troisieme_module.py:0:0: missing-gplv3-license: GPLv3 license
header not found in the first 20 lines of the module
ERROR: plugins/modules/mon_troisieme_module.py:3:0: import-before-documentation: Import
found before documentation variables. All imports must appear below DOCUMENTATION/EXAMPLES/RETURN.
See documentation for help: https://docs.ansible.com/ansible-core/2.15/dev_guide/testing/
sanity/validate-modules.html
FATAL: The 1 sanity test(s) listed below (out of 1) failed. See error output above for
details.
validate-modules
```

Les corrections à faire

```
author:
- "Alex (@alex)"
```

```
module: mon_troisieme_module
```

```
#!/usr/bin/python
# Copyright: (c) 2024, Ton Nom <ton@email.com>
# GNU General Public License v3.0+ (see COPYING or https://www.gnu.org/licenses/gpl-3.0.txt)
```

```
from ansible.module_utils.basic import AnsibleModule
import os

def main():
```