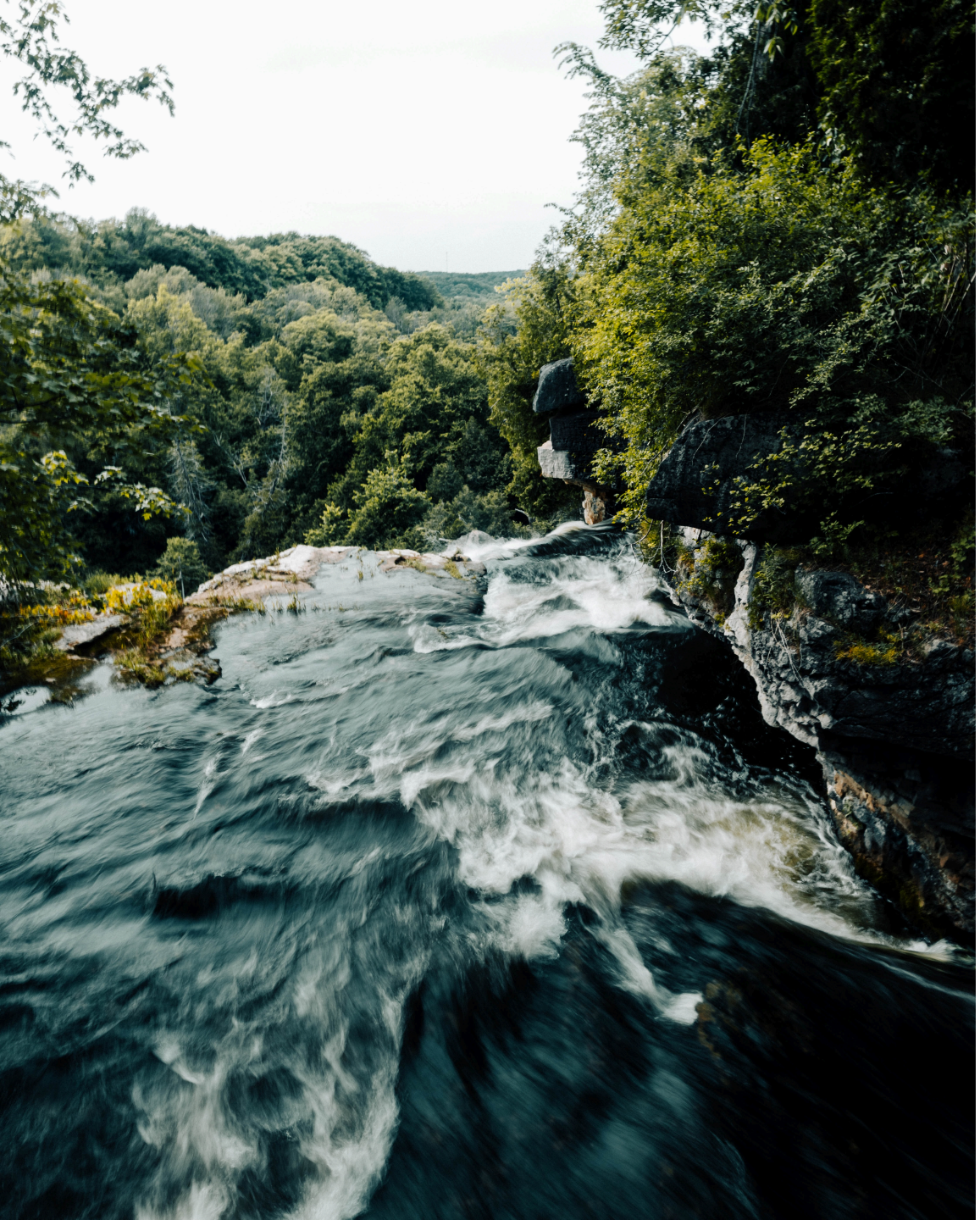
Support pédagogique Ansible - Plugins



Les plugins

Les différents types de plugins Ansible

Qu'est-ce qu'un plugin ?

- Extension du comportement natif d’Ansible
- Écrits en Python
- Permettent de personnaliser ou d'étendre des fonctionnalités

> Si vous souhaitez le vérifier, consultez docs.ansible.com/ansible/latest/plugins/plugins.html

Types de plugins

1. Action Plugins

- Contrôlent la manière dont les modules sont exécutés
- Exemples : `command`, `copy`

2. Callback Plugins

- Permettent d'afficher ou de traiter les résultats
- Utilisés pour le logging, les notifications, etc.
- Exemple :
  - `default` : sortie standard
  - `json`, `yaml`, `minimal`, `slack`, `email`

Types de plugins

3. Connection Plugins

- Gèrent la méthode de connexion aux hôtes
- Exemples : `ssh`, `paramiko`, `local`, `docker`, `httpapi`

4. Lookup Plugins

- Permettent de récupérer des données depuis des sources externes
- Exemples : `file`, `env`, `passwordstore`, `redis_kv`
- Exemple d'utilisation :

```
- debug: msg="{{ lookup('env', 'HOME') }}"
```

Types de plugins

5. Inventory Plugins

- Utilisés pour définir dynamiquement les inventaires
- Exemples : `yaml`, `ini`, `constructed`, `aws_ec2`, `azure_rm`

6. Vars Plugins

- Injectent des variables dans le contexte d'exécution
- Exemples : `host_group_vars`, `extra_vars`, `etcd`

Types de plugins

7. Filter Plugins

- Permettent de transformer des données dans Jinja2
- Exemples :

```
{{ mylist | join(',') }}
{{ some_string | regex_replace('a', 'b') }}
```

8. Test Plugins

- Utilisés dans des expressions conditionnelles (tests booléens)
- Exemples :

```
when: myvar is defined
when: mylist is iterable
```

Types de plugins

9. Strategy Plugins

- Définissent la manière dont les tâches sont exécutées (linéaire, free, debug)
- Par défaut : `linear`
- Autres : `free`, `debug`

Où les trouver et comment les charger

- Chemins standards : `/usr/share/ansible/plugins/`, `~/.ansible/plugins/` (dépend du path d'installation de Ansible)
- Ou spécifiés dans `ansible.cfg` :

```
[defaults]
callback_plugins = ./plugins/callback
```

- Certains plugins nécessitent une activation explicite via `ansible.cfg` ou ligne de commande

Les Plugins / Lookup

> plugin-lines.yml

```
- name: checking out lines and pipes lookup plugins
  hosts: localhost
  gather_facts: no
  tasks:
  - name: print the first line of any file
    debug:
      msg: "{{ item[0] }}"
    loop:
    - "{{ query('lines', 'cat /etc/hosts') }}"
    - "{{ query('lines', 'cat /etc/passwd') }}"
```

> Créer un répertoire plugins : mkdir plugins && cd plugins
>
> Lancer le playbook et indiquer ce qu'il fait ?

Gestion du Callback d’affichage

> L'affichage des résultats du `playbook` peut être modifié ou personnalisé à l'aide des variables de `CALLBACK`

- Ajout de deux plugins alternatifs

```
# ansible.cfg
[defaults]
callbacks_enabled = timer, profile_roles, profile_tasks
```

- La liste des plugins :

```
$ ansible-doc -t callback -l
```

- Relance le playbook précédent

```
# ansible-playbook plugin-lines
```

Un exemple avec la configuration de Apache2

```
[...]
TASK [geerlingguy.apache : Check if localhost cert exists (RHEL 8 and later).] ************************
samedi 19 novembre 2022  11:45:45 +0100 (0:00:00.528)       0:00:04.923 *******
ok: [apche-role]

TASK [geerlingguy.apache : Ensure httpd certs are installed (RHEL 8 and later).] ***********************
samedi 19 novembre 2022  11:45:45 +0100 (0:00:00.308)       0:00:05.232 *******
skipping: [apche-role]

TASK [geerlingguy.apache : Ensure Apache has selected state and enabled on boot.] ***********************
samedi 19 novembre 2022  11:45:45 +0100 (0:00:00.036)       0:00:05.269 *******
ok: [apche-role]

PLAY RECAP *********************************************************************************************
apche-role                 : ok=14   changed=0    unreachable=0    failed=0    skipped=5
   rescued=0    ignored=0

samedi 19 novembre 2022  11:45:46 +0100 (0:00:00.630)       0:00:05.899 *******
===============================================================================
Gathering Facts ---------------------------------------------------------------------------- 1.19s
geerlingguy.apache : Enable Apache mods. --------------------------------------------------- 1.17s
geerlingguy.apache : Ensure Apache is installed on RHEL. ----------------------------------- 0.98s
geerlingguy.apache : Ensure Apache has selected state and enabled on boot. ----------------- 0.63s
geerlingguy.apache : Add apache vhosts configuration. -------------------------------------- 0.53s
geerlingguy.apache : Get installed version of Apache. -------------------------------------- 0.40s
geerlingguy.apache : Configure Apache. ----------------------------------------------------- 0.39s
geerlingguy.apache : Check if localhost cert exists (RHEL 8 and later). -------------------- 0.31s
geerlingguy.apache : Define apache_packages. ----------------------------------------------- 0.05s
geerlingguy.apache : Ensure httpd certs are installed (RHEL 8 and later). ------------------ 0.04s
geerlingguy.apache : Configure Apache. ----------------------------------------------------- 0.03s
geerlingguy.apache : Include OS-specific variables. ---------------------------------------- 0.03s
[...]
```

Exercice

> Créer un playbook Ansible qui utilise un plugin de callback pour enregistrer automatiquement les résultats de l'exécution dans un fichier de log.

log-playbook.yml

```
- name: Exemple log dans un fichier
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Test de ping
      ping:

    - name: Message de test
      debug:
        msg: "Ce message sera loggé dans le fichier"
```

Créer un plugin

Créer un plugin

> Nous allons créer un **plugin de test personnalisé** qui vérifie si une chaîne de caractères correspond à une nuance de bleu.

- On crée un repertoire dédié pour notre plugin de test personnalisée

```
# mkdir /root/ansible-practice/plugins/test_plugins -p && cd /root/ansible-practice/plugins/test_plugins
```

Créer un plugin

- Créer `blue.py`

```
def is_blue(string):
    blue_values = [
        'blue',
        '#0000ff',
        '#00f',
        'rgb(0,0,255)',
        'rgb(0%,0%,100%)',
    ]
    if string in blue_values:
        return True
    else:
        return False

class TestModule(object):
    ''' Ansible core blue test '''

    def tests(self):
        return {
            'blue': is_blue,
        }
```

Créer un plugin

- On crée le playbook `playbook_blue.yml`

```
- hosts: localhost
  vars:
    my_color_choice: blue
  tasks:
    - name: "Verify {{ my_color_choice }} is a form of blue."
      assert:
        that: my_color_choice is blue
```

```
$ ansible-playbook playbook_blue.yml
```

```
TASK [Verify blue is a form of blue.] ******************************************************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}
```

Créer un plugin

> verifier avec `my_color_choice: red`

```
# ansible-playbook playbook_blue.yml -e my_color_choice=red
```

> verifier avec `my_color_choice: #0000ff`

```
# ansible-playbook playbook_blue.yml -e my_color_choice=#0000ff
```

Un plugin appliqué à une collection

- On se place dans le projet créé précédemment

```
# cd /root/ansible-practice/projet00
```

- On copie notre plugin personnalisé `blue` dans notre collection `coll`

> Mettre `blue.py` dans `collections/ansible_collections/coll/test/plugins/test/`

```
 # mkdir  collections/ansible_collections/coll/test/plugins/test/
```

```
 # cp ~/ansible-practice/plugins/test_plugins/blue.py collections/ansible_collections/coll/test/plugins/test/
```

Un plugin appliqué à une collection

- Modification du `site.yml` pour intégrer la variable `my_color_choice`

```
---
- name: test collection import
  hosts: localhost
  gather_facts: false
  tasks:
    - ansible.builtin.import_role:
        name: coll.test.rtest
      vars:
        my_color_choice: blue
```

Créer un plugin / collection

Un plugin appliqué à une collection

- On ajuste le role `rtest` pour integrer le plugin `blue`

```
cat collections/ansible_collections/coll/test/roles/rtest/tasks/main.yml
```

```
- name: "Verify {{ my_color_choice }} is a form of blue."
  ansible.builtin.assert: { that: my_color_choice is coll.test.blue }
```

> Attention FQCN ! `coll.test.blue`

Un plugin appliqué à une collection

- On relance le playbook `site.yml`

```
# ansible-playbook site.yml

TASK [coll.test.rtest : Debug test.]

ok: [localhost] => {
    "msg": "Un role test !!!"
}

TASK [coll.test.rtest : Verify blue is a form of blue.]

ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}
```

Un plugin appliqué à une collection

```
# ansible-playbook site.yml -e my_color_choice=red
```

```
PLAY [test collection import]

TASK [coll.test.rtest : Debug test.]

ok: [localhost] => {
    "msg": "Un role test !!!"
}

TASK [coll.test.rtest : Verify red is a form of blue.]

fatal: [localhost]: FAILED! => {
    "assertion": "my_color_choice is coll.test.blue",
    "changed": false,
    "evaluated_to": false,
    "msg": "Assertion failed"
}
```