

Ansible

Ressources :

8 steps to developing an Ansible role in Linux

In the article How to use Ansible to configure Vim, I developed an Ansible playbook to configure an initial Vim environment using a few Vim plugins. In this current article, I continue

 <https://www.redhat.com/sysadmin/developing-ansible-role>



Playbook

Le playbook va lire le(s) rôle(s) à utiliser d'abord dans son `pwd`, ensuite dans `/etc/ansible/roles`. La bonne pratique est de rester dans le même répertoire.

Dans les tâches du playbook, on va importer le(s) rôle(s) intéressant(s).

Exemple :

```
$ cat playbook.yml

- name: Config example
  hosts: localhost
  gather_facts: yes
  become: no # Devenir root sur les machines

  tasks:
    - name: task_name
      import_role:
        name: rôle_to_import
```

Exécuter un playbook :

```
$ ansible-playbook -bK playbook.yml
```

Rôles

```
Role_name
|
├─ defaults
|   └─ main.yml
├─ files
├─ handlers
|   └─ main.yml
├─ meta
|   └─ main.yml
├─ README.md
├─ tasks
|   └─ main.yml
├─ templates
├─ tests
|   └─ inventory
|       └─ test.yml
└─ vars
    └─ main.yml
```

Le `meta/main.yml` permet de définir les dépendances avec d'autres rôles si le rôle utilise d'autres rôles pour fonctionner. C'est ici dedans qu'il faut le mettre.

Le `tasks/main.yml` définit les tâches à réaliser, c'est ici qu'on retrouve les configurations à déployer. Par défaut, le `main.yml` est lu. Il est possible de diviser les tâches avec d'autres fichiers, il faut juste les appeler depuis le `main.yml` au moyen de `include_tasks` ou `import_tasks`. Afin de rendre le rôle le plus générique possible, il est possible de définir des variables comme `"{{ variable_name }}"`. Ce sont des variables définies dans `defaults` (voir ci-après).

Le `defaults/main.yml` permet de donner des valeurs par défaut aux variables utilisées comme juste avant.

Le `vars/main.yml` permet aussi de définir des variables par défaut. La différence est que ces variables sont supposées ne pas devoir changer.

Le répertoire `files` permet d'y placer les fichiers utilisés lors de l'utilisation du `copy` dans les tâches.

Exemple : le fichier `vimrc` doit être présent dans `files` .

```
- name: Ensure .vimrc config in place
  copy:
    src: vimrc
    dest: "{{ vimrc }}"
    owner: user
    group: group
    mode 0644 # droits
```

Le répertoire `handlers` permet d'exécuter certaines tâches. Afin de les exécuter, il faut utiliser le `notify` pour *trigger* le handler. Ils sont écrits et exécutés à la fin du `playbook` afin de démarrer, redémarrer ou stopper un service.