Lab4 _ Mettre en œuvre les rôles

Aborder les rôles :

Un rôle est en réalité un « playbook » qui va réaliser certaines tâches prédéfinies. Vous allez donc inclure ce rôle dans votre playbook pour pouvoir l'utiliser, les rôles constituent une sorte de librairie de fonctions.

Ces rôles incluent bien sûr du code, mais aussi des variables, des tâches, des « handlers », etc. La structure d'un rôle est en réalité une arborescence de fichiers YML dans des répertoires spécifiques et des fichiers principaux main.yml.

Utiliser les rôles dans un playbook :

Soit le playbook suivant :

```
---
- name: Installation du serveur
  hosts: web1
  tasks:
    - name: Installation des dépendances
      apt: name={{ item }} update_cache=yes state=latest
      with_items:
        - vim
        - htop
        - git
    - name: Création d'un utilisateur
      user: name=adrien comment='Mon nouvel utilisateur' shell=/usr/bin/bash
    - name: Ajout de la clé SSH
      authorized_key: user=adrien key="{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
```

Si nous continuos notre script, nous risquons vite d'obtenir un fchier un peu long et nous allons perdre la réutilisatbilté de notre script.

Pour éviter ce genre de problème , ansible utilise ce qu'on appelle des « roles ».

Créons un dossier **roles/utils/tasks** et ajoutons le fichier **main.yml**. Son contenu sera juste un extrait de notre playbook précédent :

```
- name: Installation des dépendances
  apt: name={{ item }} update_cache=yes state=latest
  with_items:
    - vim
    - htop
    - git
```

De même créons **roles/user/tasks** et ajoutons le fichier **main.yml** :

```
  - name: Création d'un utilisateur
    user: name=adrien comment='Mon nouvel utilisateur' shell=/usr/bin/bash
  - name: Ajout de la clé SSH
    authorized_key: user=adrien key="{{ lookup('file', '~/.ssh/id_rsa.pub') }}"
```

Et notre playbook devient :

```
- name: Installation du serveur
  hosts: web
  remote_user: root
  vars:

  roles:
    - utils
    - user
```

Utiliser les rôles avec Ansible Galaxy

Ansible Galaxy fait référence au site Web de Galaxy à partir duquel les utilisateurs peuvent partager des rôles. Il fait aussi référence à un outil en ligne de commande pour l'installation, la création et la gestion de rôles à partir de dépôts git.

Les rôles permettent de charger automatiquement certains fichiers vars_files, tasks et handlers en fonction d'une structure de fichier connue. Le regroupement de contenu par rôles permet également de les partager facilement avec d'autres utilisateurs.

En bref, une organisation en rôle n'est jamais qu'une manière d'**abstraire les tâches** d'un livre de jeu. Toutes les règles de conception d'un livre de jeu sont respectées sur base d'une structure de fichiers et de dossiers connue. On se base alors sur une structure de dossier et de fichiers ***par convention***.

Exemple de structure d'un projet utilisant des rôles.

```
site.yml
webservers.yml
fooservers.yml
roles/
    common/               # Cette hiérarchie représente un "role"
        tasks/            #
            main.yml      #  <-- peut inclure des fichiers de tâches plus
                                  petits si cela est justifié
        handlers/         #
            main.yml      #  <-- fichiers de handlers
        templates/        #  <-- fichiers à utiliser avec le module
                                "template"
            ntp.conf.j2   #  <------- modèles finissent en .j2
        files/            #
            bar.txt       #  <-- fichiers à utiliser avec le module "copy"
            foo.sh        #  <-- fichiers de script à utiliser avec le
                                  module "script"
        vars/             #
            main.yml      #  <-- variables associées avec le rôle
        defaults/         #
            main.yml      #  <-- variables par défaut(basse priorité)
```

```
        meta/              #
            main.yml       #  <-- dépendances du rôle
    webservers/
         tasks/
         defaults/
         meta/
```

Les rôles s'attendent à ce que les fichiers se trouvent dans certains répertoires dont le nom est connu. Les rôles doivent inclure au moins un de ces répertoires, mais il est parfaitement fonctionnel d'exclure ceux qui ne sont pas nécessaires. Lorsqu'il est utilisé, chaque répertoire doit contenir un fichier `main.yml`.

Création d'un rôle :

```
cd roles
ansible-galaxy init install_nginx
```

```
---
- name: "Nginx installation"
  apt:
    name: "nginx"
    state: latest
- name: nginx service activation
  service:
    name: nginx
    state: started
    enabled: yes
```

```
---
- name: "Nginx installation"
  hosts: web
  become: yes
  roles:
    - role: "install_nginx"
```

Utiliser la priorité des variables

On va parler maintenant des variables et de leurs priorités. J'ai rajouté ici, une tâche, comme nom « debug », avec le module debug pour afficher la valeur d'une variable. On va utiliser par exemple, nginx_port, pour définir le port du serveur web.

Il faudra initialiser à différents endroits. On va commencer par la priorité la plus faible. C'est celle des valeurs par défaut dans le rôle.

Si on va sur defaults et main.yml, on pourra créer cette variable, et on va lui donner un port de 1 000, on sauvegarde, et on va exécuter la commande ansible-playbook, et le nom du playbook.

```
---
- name: "Nginx installation"
  hosts: web
  become: yes
  roles:
    - role: "install_nginx"
  tasks:
    - name: "Debug"
      debug:
        msg : "{{ nginx_port }}"
```

Maintenant, il y a un niveau plus élevé, qui est celui du playbook. Si dans le playbook, on crée la variable, comme par exemple, vars ici, avec nginx_port : et on va lui donner une valeur de 2 000, et on sauvegarde. On va ré-exécuter la commande

```
---
- name: "Nginx installation"
  hosts: web
  vars:
    nginx_port: 2000
  become: yes
  roles:
    - role: "install_nginx"
  tasks:
    - name: "Debug"
      debug:
        msg : "{{ nginx_port }}"
```

Dans vars/main.yml on va initialiser la variable à ce niveau-là, et on va lui donner cette fois-ci, la valeur 3 000, et on va à nouveau ré-exécuter le playbook.

Il reste une dernière possibilité, qui est la plus forte, c'est d'exécuter le playbook, et de passer par ses paramètres extra, donc on peut le faire à ce niveau-là : -e nginx_port = cette fois-ci 4 000, et on ré-exécute notre commande.

Utiliser les modèles de rôles

On va voir maintenant comment combiner les « templates », les templates Jinja et les rôles. Donc, si on fait ls -l ici, on va avoir un rôle create_user et un playbook qui va en fait utiliser le rôle.

```
---
- hosts: all
  tasks:
     - include_role:
         name: create_user
       vars:
         user_name: robert
         user_state: present
```

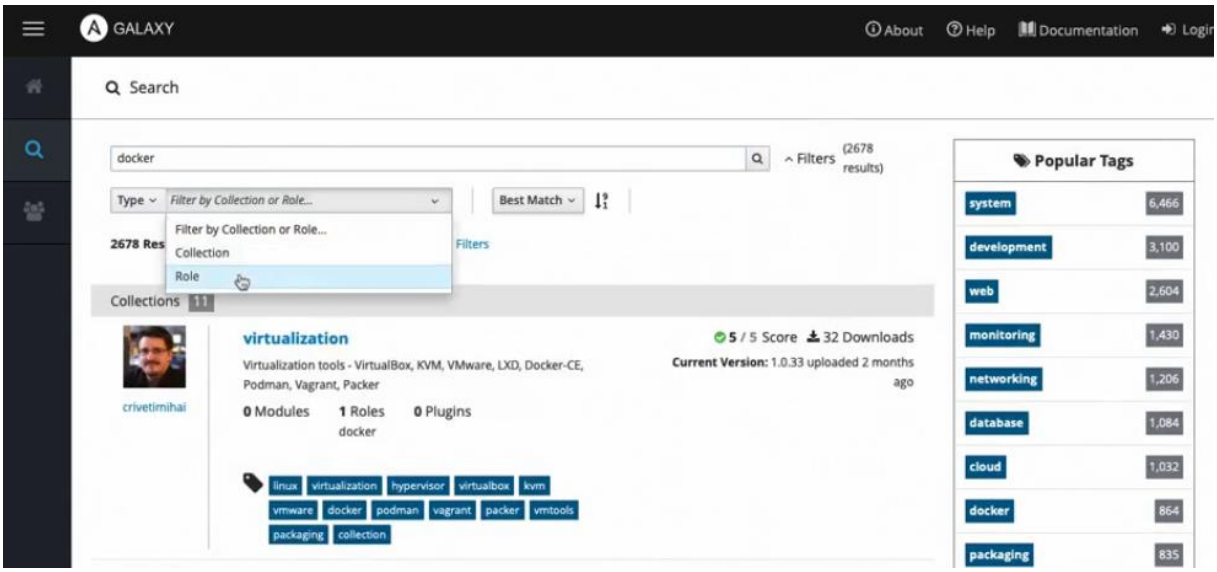
On a aussi la possibilité d'utiliser un « template » pour pouvoir en fait modifier le contenu du fichier, par exemple : fichier de configuration, du shell ou du bash.

```
---
# tasks file for user_create
- name: Create user on remote host
  become: yes
  user:
    name: '{{user_name}}'
    state: '{{user_state}}'
    remove: yes
    shell: /bin/bash
    append: yes

- name: Add bashrc to include host and user
  become: yes
  template:
    dest: '~{{user_name}}/.bashrc'
    src: templates/bashrc.j2
```

Découvrir Ansible Galaxy

Nous avons créé un ensemble de rôles pour pouvoir les réutiliser, mais aussi, pour pouvoir les mettre à disposition et les partager. Pour cela, Ansible nous fournit un site Galaxy qui permet de centraliser l'ensemble des rôles qui sont écrits par la communauté. On a d'abord en fait, une vision par thème : Système, Développement, Web et bien sûr, on peut aussi faire des recherches par mot-clé.



Pousser un rôle vers Galaxy

On va voir maintenant comment « pusher », comment envoyer votre rôle au niveau de Galaxy. Il faut d'abord aller dans le rôle. Comme le rôle est associé à un Git au niveau du fichier source, on va créer un Git et l'initialiser.

$ cd create-user

$ git init

$ git add .

$ git commit -m "premier_commit"

- Connecter à votre compte git et crèer un repository "create_user"
- Importer le git local (différents moyen de le faire) : Copier et exécuter le code depuis push an existing repository from the command line.
- Connecter à ansible galaxy:
- Vous allez trouver votre rôle si vous avez validé le chargement automatique si non on peut charger le rôle depuis le bouton add content