LES RÔLES SUR ANSIBLE

Introduction

Dans le chapitre précédent, nous avions réussi à adapter notre playbook pour les systèmes d'exploitation de type Redhat et Debian. Je vous avais aussi précisé que nous allions **améliorer la structure de notre Playbook grâce aux systèmes de rôle** proposés par Ansible.

C'est quoi un rôle et pourquoi ?

Les rôles sont une caractéristique robuste d'Ansible qui **facilitent la réutilisation**, favorisent davantage la modularisation de votre configuration et simplifient l'écriture de vos Playbooks complexes en le divisant logiquement en **composants réutilisables** et en profiter par la même occasion pour le rendre plus **facile à lire**.

Les rôles fournissent un cadre pour une réutilisation indépendante de variables, tâches, fichiers, modèles et modules. Pour faire simple, vous pouvez comparer les rôles Ansible aux librairies dans les langages de programmation. En effet, en programmation les librairies contiennent leurs propres variables et fonctions que vous pouvez réutiliser pour vos différents projets de programmation. C'est le cas aussi pour les rôles, qui possèdent leurs propres tâches, variables, handlers etc ... que vous pouvez importer et utiliser dans vos différents playbooks.

Chaque rôle est essentiellement limité à une fonctionnalité particulière (Ex: installation d'apache) qui peut être **utilisée indépendamment** et il n'existe aucun moyen d'exécuter directement les rôles car ils ne peuvent être utilisées que depuis vos playbooks, puisque les rôles n'ont pas de paramètre explicite pour l'hôte auquel

le rôle s'appliquera, cette tâche est gérée au niveau supérieur par vos playbooks qui maintiennent les hôtes de votre fichier d'inventaire vers les rôles qui doivent être appliqués à ces hôtes.

L'anatomie d'un rôle Ansible

La **structure de répertoires des rôles** est essentielle pour créer un nouveau rôle, nous utiliserons la commande `ansible-galaxy` pour **créer le squelette de notre rôle automatiquement** :

```
ansible-galaxy init [ROLE NAME]
```

Soit dans notre exemple la création d'un rôle nommé `test-role` :

```
ansible-galaxy init test-role
```

Résultat :

```
- Role test-role was created successfully
```

Voici à quoi ressemble la structure du répertoire de notre nouveau rôle nommé `test-role` :

```
test-role
|____ defaults
|
|_________ main.yml
|____ files
|____ handlers
|
|_________ main.yml
|____ meta
|
|_________ main.yml
|____ tasks
|
|_________ main.yml
|____ templates
|____ tests
```

```
|      |_________ inventory
|
|_________ test.yml

|____ vars
|
|_________ main.yml

|____ README.md
```

Les rôles s'attendent à ce que les fichiers se trouvent dans certains noms de répertoire. Les rôles doivent inclure au moins un de ces répertoires, mais il est parfaitement correct d'exclure ceux qui ne sont pas utilisés. Lorsqu'il est utilisé, chaque répertoire doit contenir un fichier **main.yml** (sauf pour le dossier **files** et **templates**).

Voici ci-dessous **les caractéristiques de l'arborescence d'un rôle** :

- **tasks** : contient la liste principale des tâches à exécuter par le rôle.
- **handlers** : contient les handlers, qui peuvent être utilisés par ce rôle ou même en dehors de ce rôle.
- **defaults** : variables par défaut pour le rôle.
- **vars** : d'autres variables pour le rôle.
- **files** : contient des fichiers qui peuvent être déployés via ce rôle.
- **templates** : contient des modèles (jinja2) qui peuvent être déployés via ce rôle.
- **meta** : définit certaines métadonnées pour ce rôle.
- **README.md** : inclut une description générale du fonctionnement de votre rôle.
- **test** : contient notre playbook (on peut cependant déposer notre playbook à la racine du projet ou dans un dossier sous un nom différent).

Notre but dans cet article est de **répartir les tâches de notre ancien playbook LAMP sous forme de rôles** en utilisant l'arborescence ci-dessus. Nous aurons ainsi un rôle pour l'installation et la configuration de la partie web et un second pour notre base de données.

Création de nos rôles

Commencez par télécharger note ancien projet complet en cliquant ici.

La partie web

Dans la racine du projet créez un dossier nommé **roles** qui contiendra pour l'instant notre rôle **web**. Dans ce même sous dossier rajoutons l'arborescence d'un rôle avec la commande vue précédemment `ansible-galaxy` :

```
mkdir roles
cd roles
ansible-galaxy init web
cd ../..
```

Nous supprimerons ensuite le dossier **tests** , **defaults**, **vars** et **handlers** qui nous ne serviront à rien pour le moment:

```
rm -rf roles/web/{tests,defaults,handlers,vars}
```

La première étape repose sur le déplacement de nos fichiers jinja2 et de nos fichiers sources situés respectivement dans le dossier **templates** et **files** vers notre nouveau rôle **web**. Pour ce faire, placez-vous donc sur la racine de votre projet et lancez les commandes suivantes :

```
mv templates/db-config.php.j2 roles/web/templates
mv files/app  roles/web/files
```

```
rm -rf files
```

L'étape suivante consiste à déplacer les tâches de la partie web directement dans le fichier de tâches de notre rôle `web` situé dans **roles/web/tasks/main.yml** qui ressemblera après changement à ceci :

```
---
# tasks file for web
- name: install apache and php last version for (Debian os family)
  apt:
    name: ['apache2', 'php', 'php-mysql']
    state: present
    update_cache: yes
  when: ansible_facts['os_family'] == "Debian"

- name: install extra packages (Debian os family)
  apt:
    name: "{{ extra_packages_debian }}"
    state: present
  when: ansible_facts['os_family'] == "Debian" and extra_packages_debian is defined

- name: install apache and php last version for (RedHat os family)
  yum:
    name: ['httpd', 'php', 'php-mysqlnd']
    state: present
    update_cache: yes
  when: ansible_facts['os_family'] == "RedHat"

- name: install extra packages (RedHat os family)
  yum:
    name: "{{ extra_packages_redhat }}"
    state: present
  when: ansible_facts['os_family'] == "RedHat" and extra_packages_redhat is defined


- name: Give writable mode to http folder
  file:
    path: /var/www/html
    state: directory
    mode: '0755'

- name: remove default index.html
  file:
    path: /var/www/html/index.html
    state: absent

- name: upload web app source
  copy:
    src: app/
```

```
    dest: /var/www/html/

- name: deploy php database config
  template:
    src: "db-config.php.j2"
    dest: "/var/www/html/db-config.php"

- name: ensure apache service is start (Debian os family)
  service:
    name: apache2
    state: started
    enabled: yes
  when: ansible_facts['os_family'] == "Debian"

- name: ensure apache service is start (RedHat os family)
  service:
    name: httpd
    state: started
    enabled: yes
  when: ansible_facts['os_family'] == "RedHat"

- name: enable connection with remote database (RedHat os family)
  shell: setsebool -P httpd_can_network_connect_db 1
  when: ansible_facts['os_family'] == "RedHat"
```

Pour finir, il ne faut pas oublier d'importer notre rôle dans notre playbook comme suit :

```
---

# WEB SERVER

- hosts: web
  become: true
  vars_files: vars/main.yml

  roles:
    - web

# DATABASE SERVER
# suite tâches database
```

Partie base de données.

Nous referons la même manipulation pour notre nouveau rôle `database`, mais cette fois-ci nous garderons le dossier `vars` et le dossier `handlers`, car nous les

réutiliserons plus tard :

```
mkdir roles
cd roles
ansible-galaxy init database
rm -rf database/{tests,defaults}
cd ..
```

Commençons par déplacer le dernier fichier jinja2 de notre dossier **templates** vers notre nouveau rôle **database** :

```
mv templates/table.sql.j2 roles/database/templates
rm -rf templates
```

La prochaine étape comprend le déplacement des handlers utilisés dans les tâches de la partie base de données dans le fichier **roles/database/handlers/main.yml** de notre base de données, qui ressemblera à ceci :

```
---
# handlers file for database
- name: Restart mysql # Debian os family
  service:
    name: mysql
    state: restarted

- name: Restart mysqld # RedHat os family
  service:
    name: mysqld
    state: restarted
```

Si vous ouvrez votre fichier playbook **playbook.yml**, vous pouvez apercevoir la variable suivante :

```
vars:
  default_root_password: ""
```

La variable **default_root_password** n'est utilisable que par nos tâches de la partie de la base de données, elle permet en effet de récupérer automatiquement le mot de passe root mysql. L'utilisateur du playbook n'a pas à modifier cette variable, on

peut donc la déplacer vers le fichier **roles/database/vars/main.yml**, comme suit :

```
---
# defaults file for database
default_root_password: ""
```

Maintenant, il ne nous reste qu'à déplacer les tâches de la partie de notre base de données vers les tâches du rôle `database` dans le fichier **roles/database/tasks/main.yml** :

```
---
# tasks file for database
- name: install mysql (Debian os family)
  apt:
    name:
      - mysql-server
      - python-mysqldb # for mysql_db and mysql_user modules
    state: present
    update_cache: yes
  when: ansible_facts['os_family'] == "Debian"

- name: install mysql repo (Fedora)
  yum:
    name: "http://repo.mysql.com/mysql80-community-release-fc{{ ansible_facts['distri
    state: present
    update_cache: yes
  when: ansible_facts['distribution'] == "Fedora"

- name: install mysql repo (CENTOS or RedHat)
  yum:
    name: "http://repo.mysql.com/mysql80-community-release-el{{ ansible_facts['distri
    state: present
    update_cache: yes
  when: ansible_facts['os_family'] == "RedHat" and ansible_facts['distribution'] != "

- name: install mysql package (RedHat os family)
  yum:
    name: mysql-community-server
    state: present
    disablerepo: mysql80-community
    enablerepo: mysql57-community
  when: ansible_facts['os_family'] == "RedHat"

- name: install PyMySQL from pip (RedHat os family)
  pip:
    name: PyMySQL # for mysql_db and mysql_user modules
  when: ansible_facts['os_family'] == "RedHat"
```

```
- name: ensure mysql service is start (Debian os family)
  service:
    name: mysql
    state: started
    enabled: yes
  when: ansible_facts['os_family'] == "Debian"


- name: ensure mysqld service is start (RedHat os family)
  service:
    name: mysqld
    state: started
    enabled: yes
  when: ansible_facts['os_family'] == "RedHat"


- name: Allow external MySQL connections (1/2) (Debian os family)
  lineinfile:
    path: /etc/mysql/mysql.conf.d/mysqld.cnf
    regexp: '^skip-external-locking'
    line: "# skip-external-locking"
  notify: Restart mysql
  when: ansible_facts['os_family'] == "Debian"


- name: Allow external MySQL connections (2/2) (Debian os family)
  lineinfile:
    path: /etc/mysql/mysql.conf.d/mysqld.cnf
    regexp: '^bind-address'
    line: "# bind-address"
  notify: Restart mysql
  when: ansible_facts['os_family'] == "Debian"


- name: check if mysql config is correct (RedHat os only)
  shell: 'grep "^bind-address" /etc/my.cnf'
  register: test_grep
  when: ansible_facts['os_family'] == "RedHat"
  ignore_errors: yes # dont exit if it doesn't found something

- name: change mysql config (RedHat os only)
  blockinfile:
    path: /etc/my.cnf
    insertafter: EOF
    block: |
      default_authentication_plugin=mysql_native_password
      bind-address=0.0.0.0
      default_password_lifetime=0
      validate_password_policy=LOW
      validate_password_length=6
      validate_password_number_count=0
  when: ansible_facts['os_family'] == "RedHat" and test_grep.rc != 0
  notify: Restart mysqld
```

```
- name: Register temporary password (RedHat os family)
  shell: "grep 'temporary password' /var/log/mysqld.log  | awk '{print $(NF)}'"
  register: password_tmp
  when: ansible_facts['os_family'] == "RedHat"

- name: Set default root user password (RedHat os family)
  set_fact:
    default_root_password: '{{ password_tmp.stdout }}'
  when: ansible_facts['os_family'] == "RedHat"

- name: Change root SQL password and GRANT root privileges (RedHat os family)
  command: "mysql --user=root --password={{ default_root_password }} --connect-expire
  ignore_errors: yes # ignore errors because we only change mysql root password once
  when: ansible_facts['os_family'] == "RedHat"

- name: Create MySQL client config (Debian os family)
  copy:
    dest: "/root/.my.cnf"
    content: |
      [client]
      user=root
      password={{ root_password }}
    mode: 0400
  when: ansible_facts['os_family'] == "Debian"


- name: upload sql table config
  template:
    src: "table.sql.j2"
    dest: "/tmp/table.sql"

- name: add sql table to database
  mysql_db:
    name: "{{ mysql_dbname }}"
    state: present
    login_user: root
    login_password: '{{ root_password }}'
    state: import
    target: /tmp/table.sql


- name: "Create {{ mysql_user }} with all {{ mysql_dbname }} privileges (Debian os fa
  mysql_user:
    name: "{{ mysql_user }}"
    password: "{{ mysql_password }}"
    priv: "{{ mysql_dbname }}.*:ALL"
    host: "{{ webserver_host }}"
    state: present
    login_user: root
    login_password: '{{ root_password }}'
    login_unix_socket: /var/run/mysqld/mysqld.sock
  when: ansible_facts['os_family'] == "Debian"
  notify: Restart mysql
```

```
- name: "Create {{ mysql_user }} with all {{ mysql_dbname }} privileges (RedHat os fa
  mysql_user:
    name: "{{ mysql_user }}"
    password: "{{ mysql_password }}"
    priv: "{{ mysql_dbname }}.*:ALL"
    host: "{{ webserver_host }}"
    state: present
    login_user: root
    login_password: '{{ root_password }}'
    login_unix_socket: /var/lib/mysql/mysql.sock
  when: ansible_facts['os_family'] == "RedHat"
  notify: Restart mysqld
```

Enfin la dernière étape consiste à importer notre rôle `database` dans notre playbook, voici donc à quoi ressemblera la version finale de notre playbook :

```
---

# WEB SERVER
- hosts: web
  become: true
  vars_files: vars/main.yml

  roles:
    - web

# DATABASE SERVER
- hosts: db
  become: true
  vars_files: vars/main.yml

  roles:
 - database
```

Ansible Galaxy

C'est quoi ?

Ansible Galaxy est une plateforme Web où les utilisateurs peuvent **partager leurs rôles Ansible** et c'est également un outil en ligne de commande pour installer, créer

et gérer des rôles.

> **Information**
>
> Nous avions déjà eu l'occasion d'utiliser précédemment la commande `ansible-galaxy init`.

Ansible Galaxy est un site gratuit qui vous permet de rechercher, télécharger et partager des rôles développés par la communauté. Le téléchargement de rôles depuis Galaxy est un excellent moyen de relancer vos projets d'automatisation.

Vous pouvez également utiliser le site pour partager les rôles que vous créez. En vous authentifiant auprès du site à l'aide de votre compte GitHub. Vous pouvez importer des rôles et les rendre disponibles à la communauté Ansible. Les rôles importés deviennent disponibles dans l'index de recherche Galaxy et visibles sur le site, permettant aux utilisateurs de les découvrir et de les télécharger. Nous allons de ce fait, **partager un de nos deux rôles dans la plateforme Ansible Galaxy**.

Documentez vos rôles !

Avant toute chose, il faut commencer par documenter vos rôles. La commande `ansible-galaxy init` lancée antérieurement, créée également à la racine de vos rôles un fichier nommé **README.md** où nous devons décrire dedans l'utilisation de notre rôle. Par exemple dans mon cas, voici à quoi ressemble ma documentation pour le rôle `web` :

```
web
=========

install apache and php with a test web app.
```

````
Requirements
------------

Debian os family or RedHat OS family

Role Variables
--------------

| name                     | type   | description
|--------------------------|--------|-----------------------------------------------
| `mysql_user`             | string | Mysql user that will be used in the php app
| `mysql_password`         | string | Mysql password that will be used in the php app
| `mysql_dbname`           | string | Database name that will be used in the php app
| `db_host`                | string | Database ip/host that that will be used in the p
| `extra_packages_debian` | list   | extra Debian packages that will be downloaded
| `extra_packages_redhat` | list   | extra RedHat packages that will be downloaded

Dependencies
------------

n/a

Example Playbook
----------------

```yaml
- hosts: web
  become: yes
  vars: vars/main.yml
    - mysql_user: "admin"
    - mysql_password: "Test_34535$"
    - mysql_dbname: "blog"
    - db_host: "192.168.0.22"
    - extra_packages_debian: ['php-curl', 'php-gd', 'php-mbstring']
    - extra_packages_redhat: ['php-xml', 'php-gd', 'php-mbstring']

  roles:
    - web
```

License
-------

BSD

Author Information
------------------

[https://devopssec.fr/](https://devopssec.fr/)
````

Lorsque vous recherchez ou importez un rôle depuis Ansible Galaxy, le processus de recherche ou d'importation se base sur les métadonnées trouvées dans le fichier

meta/main.yml du rôle. Ce fichier peut contenir les informations suivantes :

```
galaxy_info:
  role_name:
  author:
  description:
  company:
  license:
  min_ansible_version:
  platforms:
  galaxy_tags:
dependencies: []
```

Ci-dessous une explication de certains paramètres :

- **`role_name`** : (Optionnel) nom du rôle.
- **`galaxy_tags`** : (Optionnel) fournir des balises qui sont unique qui permettent de classer votre rôle (une par ligne).
- **`plateformes`** : (Obligatoire) liste de plates-formes valides avec une liste de versions valides. (La liste des plate-formes est disponibles ici).
- **`dependencies`** : (Optionnel) liste de vos dépendances de rôle ici, ces dépendances seront automatiquement installées (une par ligne).

Voici à quoi ressemble par exemple le fichier **meta/main.yml** de notre rôle **`web`** :

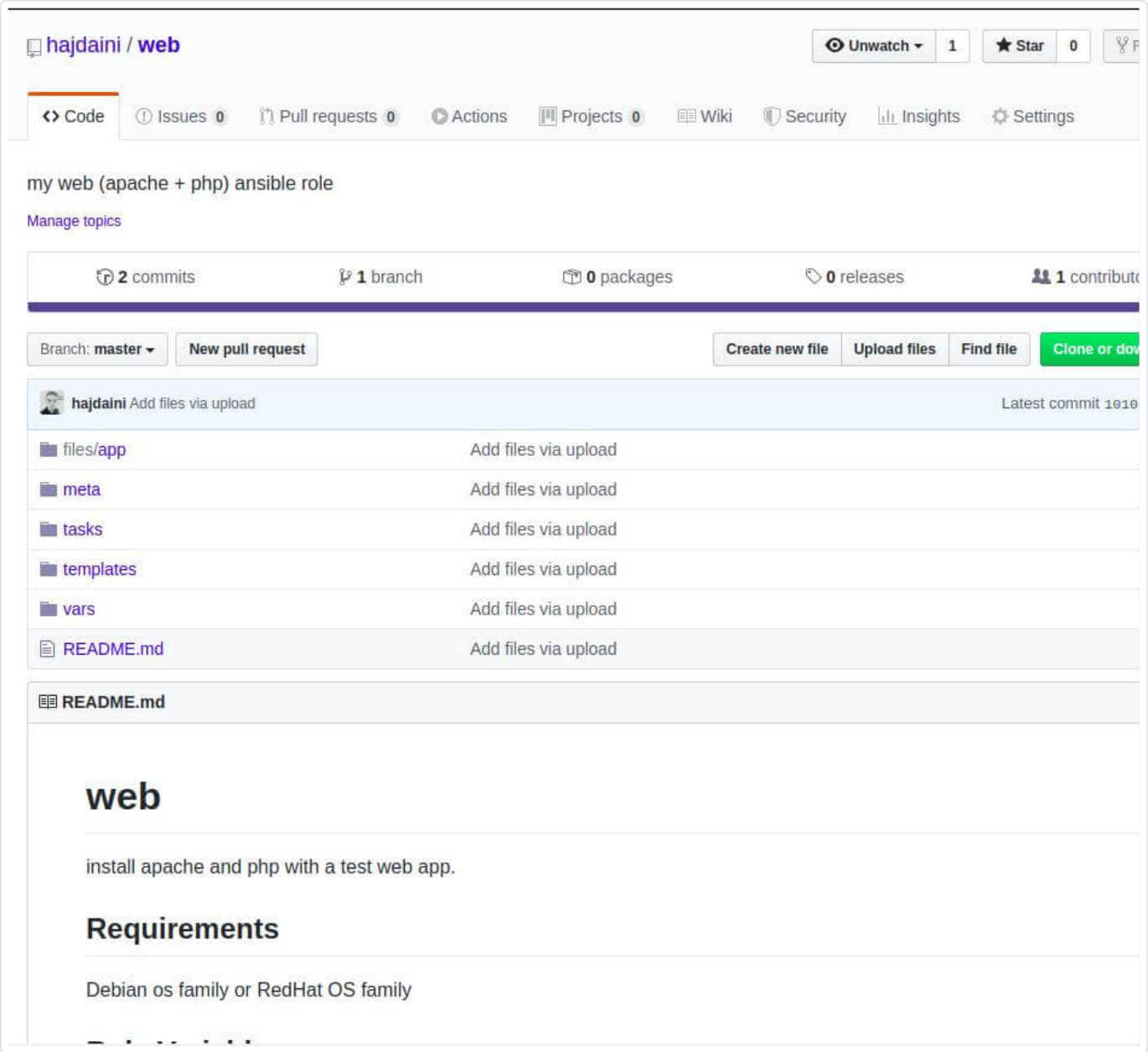
```
galaxy_info:
  author: AJDAINI Hatim
  description: install apache and php with a test web app
  company: https://devopssec.fr

  license: MIT

  min_ansible_version: 1.6

  platforms:
  - name: Fedora
    versions:
      - 28
      - 29
      - 30
```

```
      - 31
  - name: Ubuntu
    versions:
      - all
  - name: Debian
    versions:
      - all
  - name: EL
    versions:
      - 7

  galaxy_tags:
    - web
    - apache
    - php

dependencies: []
```

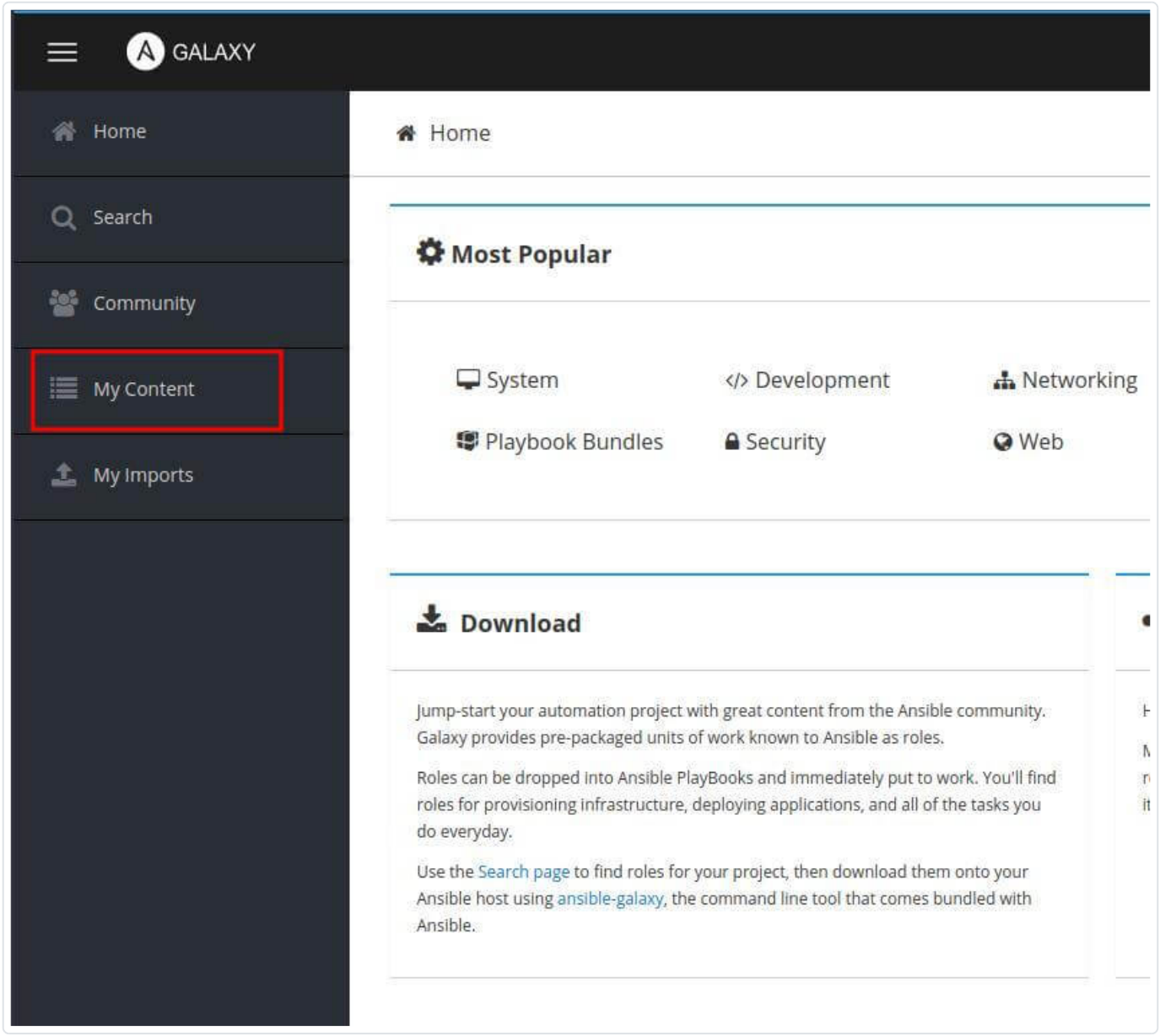
Publier votre rôle sur Ansible Galaxy

> **Attention**
>
> J'ai créé temporairement ce repository Github qui sera par la suite supprimé à la publication de cet article.

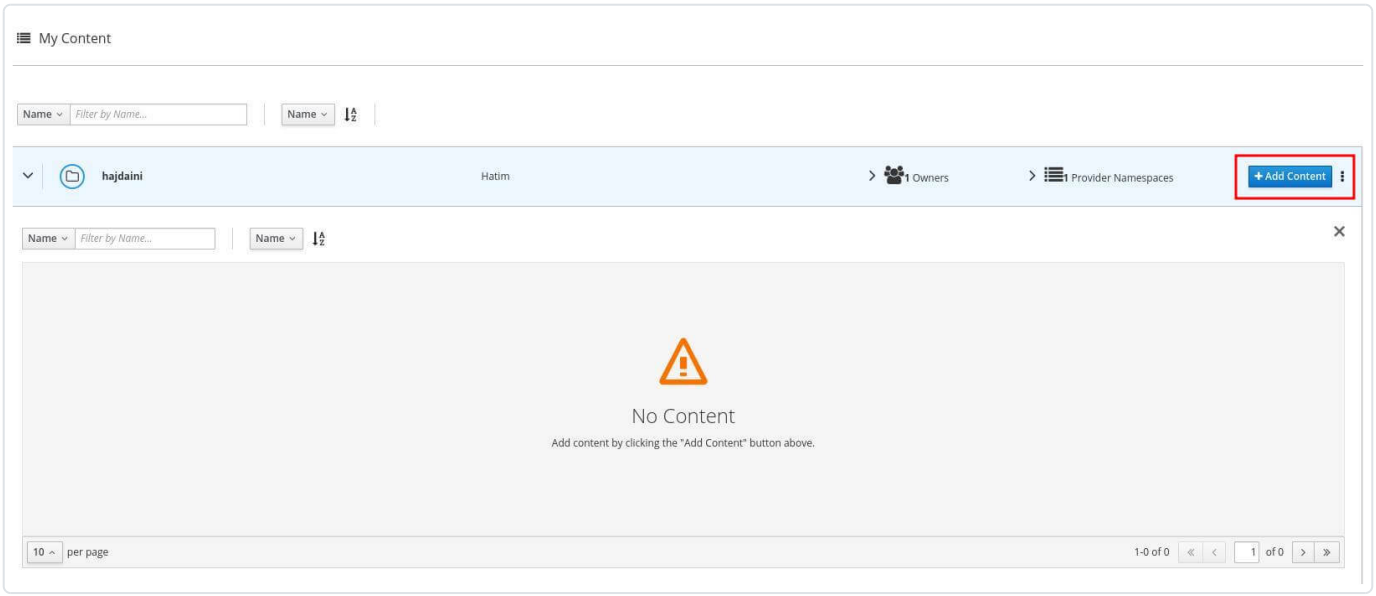
Vous devez au préalable posséder un compte Github pour publier vos rôles dans la plateforme Ansible Galaxy. La première étape consiste à créer un répository Github où vous déposerez directement dedans l'arborescence de votre rôle :



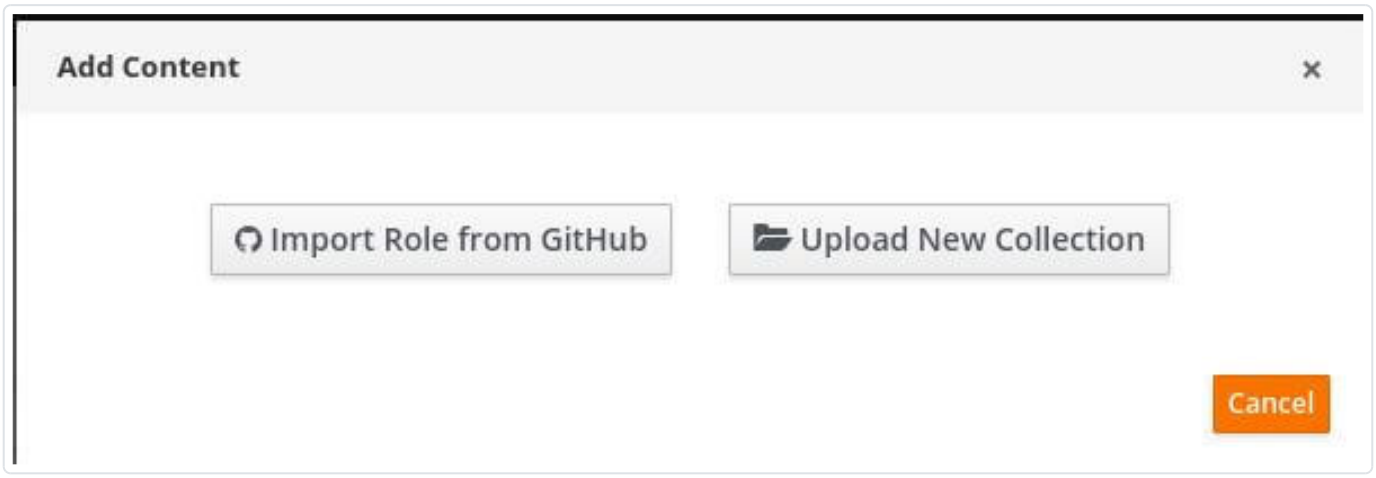
Connectez-vous ensuite avec votre compte Github sur le site Ansible Galaxy et cliquez sur le bouton "My content" sur le menu à gauche :



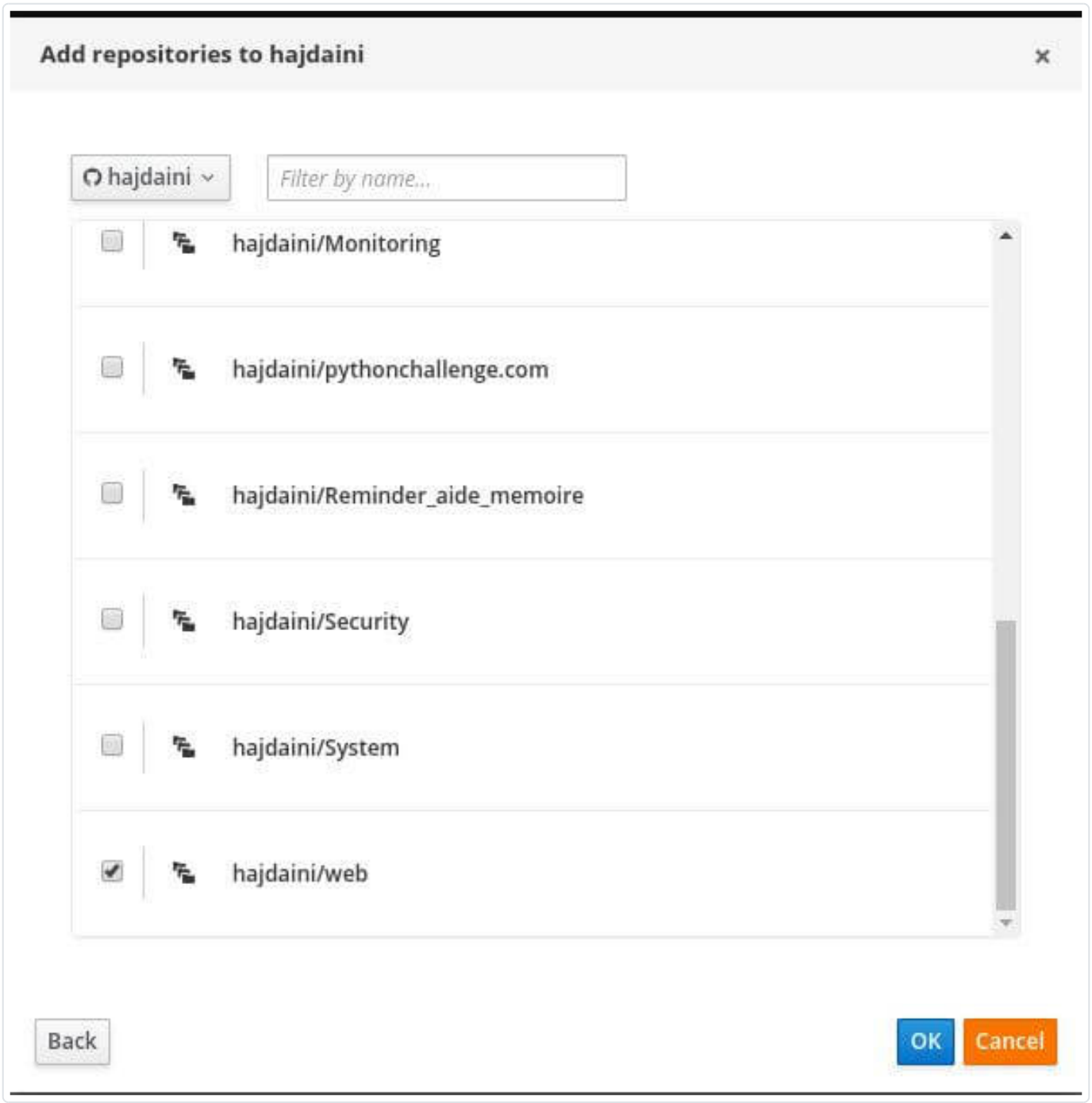
Par la suite, cliquez sur le bouton "+ Add Content" :

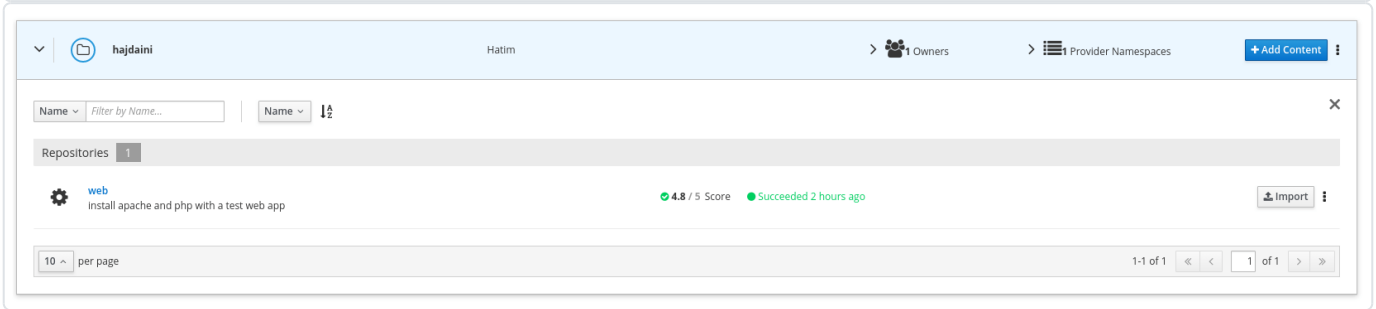


Après cela, cliquez sur le bouton "Import Role from GitHub" :



Choisissez ensuite le rôle adéquat :





Récupérer votre rôle depuis Ansible Galaxy

N'importe qui peut dorénavant télécharger votre rôle ! Vous pouvez soit lancer une recherche directement depuis la plateforme Ansible Galaxy, soit directement depuis la ligne de commande `ansible-galaxy`. Pour notre exemple, nous utiliserons la commande avec l'option `--author` afin de mieux filtrer nos recherches :

```
ansible-galaxy search web --author hajdaini
```

Résultat :

```
Found 1 roles matching your search:

Name         Description
----         -----------
hajdaini.web install apache and php with a test web app
```

Vous pouvez également **récupérer des informations supplémentaires sur votre rôle** en tapant la commande suivante :

```
ansible-galaxy info hajdaini.web
```

Résultat :

```
Role: hajdaini.web
description: install apache and php with a test web app
active: True
commit: 8a1302a30b4657f3a290287762c0dd1a7fff4b17
commit_message: Update main.yml
commit_url: https://api.github.com/repos/hajdaini/web/git/commits/8a1302a30b4657f3a2
company: https://devopssec.fr
created: 2020-02-11T12:30:22.362817Z
download_count: 1
forks_count: 0
github_branch: master
github_repo: web
github_server: https://github.com
github_user: hajdaini
id: 46451
imported: 2020-02-11T07:43:39.834522-05:00
is_valid: True
issue_tracker_url: https://github.com/hajdaini/web/issues
license: MIT
min_ansible_version: 1.6
modified: 2020-02-11T12:43:39.849377Z
```

Vous pouvez ensuite **installer le rôle** depuis la commande suivante :

```
ansible-galaxy install hajdaini.web
```

Vous pouvez également définir un fichier un fichier nommé **requirements.yml**, où vous spécifiez les différents rôles à télécharger :

```
---
roles:
  - src: https://github.com/hajdaini/web
    version: master
    name: web
  - src: https://github.com/hajdaini/database
    version: master
    name: database
```

```
ansible-galaxy install -r requirements.yml -p [CHEMIN_DE_DESTINATION]
```

Vous trouverez plus d'informations sur la documentation officielle.

Conclusion

Grâce aux systèmes de rôles, nous nous approchons de plus en plus vers un playbook de plus en plus modulable. Vous pouvez télécharger le projet complet en cliquant ici.

Le prochain chapitre se concentrera plus sur l'aspect sécurité de notre playbook LAMP. Je vous donne donc rendez-vous au prochain chapitre, où on étudiera Ansible Vault.