+adlere
DIGITAL EXPERTISE

Ansible pour professionnels Linux / Unix

Johan Klotz - jklotz@adlere.fr

27 et 28 nov. 2024

+adlere
DIGITAL EXPERTISE

Rôles

Des rôles, pour quoi faire ?

Provisionnement de système

configuration hardware (eg HP iLO)
configuration configuration (vlans, …)
virtualisation
installation OS
utilisateurs
stockage
services
durcissement
[…]

ROLES

- **Configuration**
- **Provisionnement**
- **Installation d'applications**
- **Durcissement**
- **Remédiation**

- Identifier les unités logiques, les rendre ré-utilisables et agnostiques
- pas de secrets ou de données spécifiques à un sites (partageables)

Playbook d'origine : sec_settings.yml

- Playbook utilisé dans le provisionnement d'une VM; concerne la sécurité en général

```yaml
---
- name: Various security settings
  hosts: all
  gather_facts: no


  tasks:
    - name: Setting SElinux to enforcing
      ansible.posix.selinux:
        state: enforcing
        policy: targeted


    - name: Install firewalld
      ansible.builtin.package:
        name: firewalld
        state: present


    - name: Disable firewalld
      ansible.builtin.service:
        name: firewalld
        state: stopped
        enabled: false


    - name: System reboot (timeout 300 s)
      ansible.builtin.reboot:
        reboot_timeout: 300
```

SElinux

firewall

divers

sshd

- Pas réutilisable
- susceptible de croissance organique

(suite)

```yaml
    - name: set -o vi
      ansible.builtin.blockinfile:
        insertafter: EOF
        path: /etc/profile
        state: present
        block: |
          set -o vi


    - name: Disable Root Login
      ansible.builtin.lineinfile:
        dest: /etc/ssh/sshd_config
        regexp: '^PermitRootLogin'
        line: "PermitRootLogin no"
        state: present
        backup: yes
      become: yes
      notify:
        - restart ssh


  handlers:
    - name: restart ssh
      systemd:
        name: sshd
        state: restarted
```

Évolution #1 : rôles embarqués dans le projet

- On embarque un répertoire `./roles` dans le projet

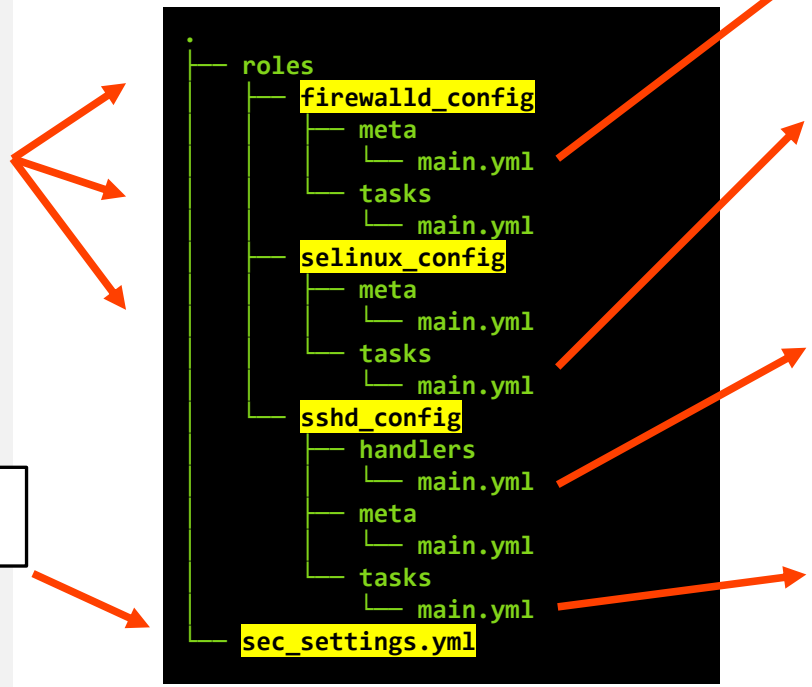
```yaml
---
- name: Various security settings
  hosts: all
  gather_facts: no

  roles:
    - firewalld_config
    - selinux_config
    - sshd_config

  tasks:
    - name: System reboot (timeout 300 s)
      ansible.builtin.reboot:
        reboot_timeout: 300

    - name: set -o vi
      blockinfile:
        insertafter: EOF
        path: /etc/profile
        state: present
        block: |
          set -o vi
```

playbook principal




```yaml
- name: Disable firewalld
  ansible.builtin.service:
    name: firewalld
    state: stopped
    enabled: false
```

```yaml
- name: Setting SElinux to enforcing
  ansible.posix.selinux:
    state: enforcing
    policy: targeted
```

```yaml
- name: restart ssh
  ansible.builtin.systemd:
    name: sshd
    state: restarted
```

```yaml
- name: SSHD ROLE | Enable Root Login
  ansible.builtin.lineinfile:
        dest: /etc/ssh/sshd_config
        regexp: '^PermitRootLogin'
        line: "PermitRootLogin no"
        state: present
        backup: yes
  become: yes
  notify:
    - restart ssh
```

- on aurait pu mettre chaque rôle directement dans le répertoire du projet
- les rôles peuvent croître indépendamment l'un de l'autre et s'enrichir
- meilleure organisation, plus lisible

Évolution #2 : rôles partagés et réutilisables

- Chaque rôle est (dé)placé dans un repository git spécifique

```
.
├── collections.yml
├── README.md
├── roles_requirements.yml
└── sec_settings.yml
```

```
$ cat collections.yml
collections:
  - ansible.posix
```

(ansible.posix nécessité par le rôle selinux)

```
$ cat roles_requirements.yml
---
- src: https://gitlab.com/myrepo/roles/firewalld_config
  scm: git
  name: firewalld_config

- src: https://gitlab.com/myrepo/roles/selinux_config
  scm: git
  name: selinux_config

- src: https://gitlab.com/myrepo/roles/sshd_config
  scm: git
  name: sshd_config
```

```
---
- name: Various security settings
  hosts: all
  gather_facts: no

  roles:
    - firewalld_config
    - selinux_config
    - sshd_config

  tasks:
    - name: System reboot (timeout 300 s)
      ansible.builtin.reboot:
        reboot_timeout: 300

    - name: set -o vi
      blockinfile:
        insertafter: EOF
        path: /etc/profile
        state: present
        block: |
          set -o vi
```

le playbook ne change pas

- rôles et collections sont décrits dans des fichiers spécifiques
- on les installe avec

```
ansible-galaxy collection install -r collections.yml
ansible-galaxy role install -r roles_requirements.yml
```

- Un repository par rôle
- Partageable, réutilisable, bonne lisibilité, peuvent croître indépendamment

Évolution #3 : appel d'un "meta"-rôle et de ses dépendances


```
.
├── collections.yml
├── README.md
├── role_requirement.yml
└── sec_settings.yml
```

```
$ cat collections.yml
collections:
  - ansible.posix
```

(ansible.posix nécessité par le rôle selinux)

```
$ cat role_requirement.yml
---
- src: https://gitlab.com/myrepo/roles/security
  scm: git
  name: security
```

```yaml
---
- name: Various security settings
  hosts: all
  gather_facts: no

  roles:
    - security

  tasks:
    - name: System reboot (timeout 300 s)
      ansible.builtin.reboot:
        reboot_timeout: 300

    - name: set -o vi
      blockinfile:
        insertafter: EOF
        path: /etc/profile
        state: present
        block: |
          set -o vi
```

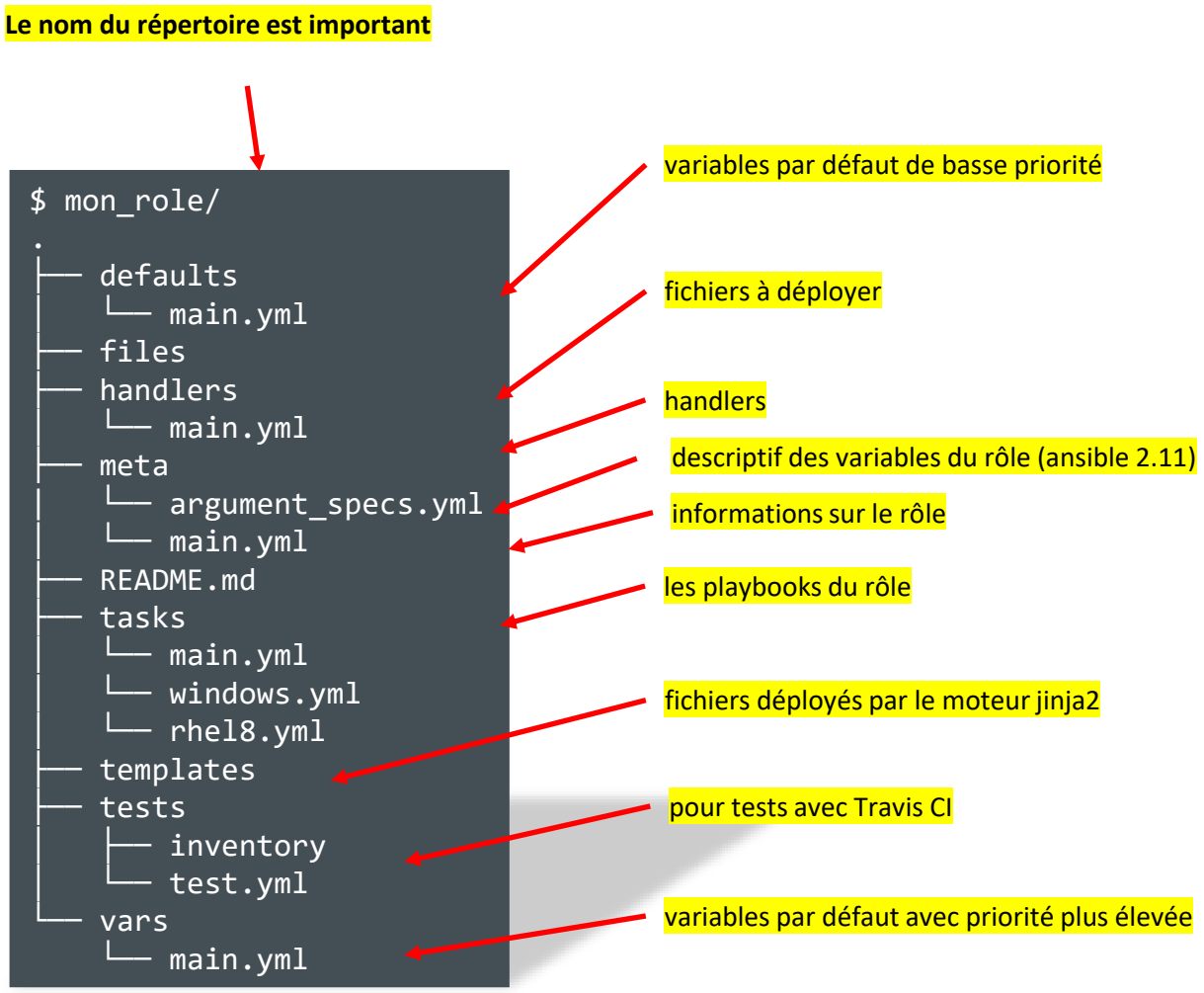
Le rôle 'security' ,'a qu'un seul fichier `meta/main.yml`, qui liste les dépendances du rôle

```yaml
dependencies:
  - role: sshd_config
    src: https://gitlab.com/myrepo/ansible/roles/sshd_config
    scm: git
  - role: selinux_config
    src: https://gitlab.com/myrepo/ansible/roles/selinux_config
    scm: git
  - role: firewalld_config
    src: https://gitlab.com/myrepo/ansible/roles/firewalld_config
    scm: git
```

le playbook référence juste un rôle

Rôles ansible

- Ensemble de playbooks et leurs ressources
- code générique, ré-utilisable, distribuable
- apportent structure et organisation logique aux playbooks
- nécessitent de la réflexion
- appelés à partir d'un playbook
- doivent respecter une structure définie
- `ansible-galaxy [role] init` initialises la structure
  - adapter `README.md` et `meta/main.yml`
    - cf. les licences
  - enlever les objets (répertoires) non utilisés



Documenter le rôle : **`meta/main.yml`** et **README.md**

https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html

Rôle inclus dans un projet

- les rôles peuvent aller dans un sous-répertoire 'roles' du projet
- ou directement dans le projet
- ce sont des bouts de code réutilisables et partageables
- ils peuvent exister dans d'autres emplacements (chemins spécifiques à définir)

```
mon_projet/
├── ansible.cfg                 # Fichier de configuration
├── inventory/                  # Répertoire de fichiers d'inventaire
├── group_vars/                 # Répertoires des variables de groupe
├── host_vars/                  # Répertoire de variables par système
├── roles/                      # Répertoire des rôles
│   ├── common/                 # Rôle 'common'
│   │   ├── tasks/              # Playbooks du rôle
│   │   │   └── main.yml
│   │   ├── handlers/           # Handlers
│   │   ├── templates/          # Fichier template Jinja
│   │   ├── files/              # Fichiers à recopier
│   │   ├── vars/               # Variables;  internes du rôle
│   │   └── defaults/           # Variables par défaut du rôle
│   └── webserver/              # Rôle 'webserver'
│       ├── […]
├── playbooks/                  # Répertoire de playbooks
├── files/                      # Fichiers à recopier
├── templates/                  # Fichiers pour template Jinja2
└── README.md                   # Documentation
```

meta/main.yml

- décrit le rôle et ses caractéristiques à travers des mots-clefs
- les tags et autres caractéristiques (platform, documentation, license, …) catégorisent le rôle sur galaxy.ansible.com ou dans un Private Automation Hub
- définit les dépendances du rôle

Dépendances

Dans meta/main.yml :

```
[…]
dependencies:
    - geerlingguy.haproxy
    - name: geerlingguy.ansible
      version: 2.0.2
[…]
```

Dans meta/requirements.yml :

```
---
  - geerlingguy.haproxy
  - name: geerlingguy.ansible
    version: 2.0.2
```

Installation (avec les dépendances)

- Depuis galaxy.ansible.com :

```
ansible-galaxy install nom.role
```

```
$ ansible-galaxy install role1.tar.gz
Starting galaxy role install process
- extracting role1.tar.gz to /home/admjkl/.ansible/roles/role1.tar.gz
- role1.tar.gz was installed successfully
- adding dependency: geerlingguy.haproxy
- adding dependency: geerlingguy.ansible (2.0.2)
- downloading role 'haproxy', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-haproxy/archive/1.3.0.tar.gz
- extracting geerlingguy.haproxy to /home/admjkl/.ansible/roles/geerlingguy.haproxy
- geerlingguy.haproxy (1.3.0) was installed successfully
- downloading role 'ansible', owned by geerlingguy
- downloading role from https://github.com/geerlingguy/ansible-role-ansible/archive/2.0.2.tar.gz
- extracting geerlingguy.ansible to /home/admjkl/.ansible/roles/geerlingguy.ansible
- geerlingguy.ansible (2.0.2) was installed successfully
$
```

- Depuis un repository Git :

```
ansible-galaxy role install git+https://github.com/geerlingguy/ansible-role-certbot
```

Installation avec un fichier de prérequis

- S'installe avec : `ansible-galaxy role install -r requirements.yml`

```
---
 roles:
 # Installe la dernière version disponible depuis Ansible Galaxy
 - name: geerlingguy.firewall

 # Installe une version spécifique, depuis Ansible Galaxy
 - name: geerlingguy.php
   version: 4.3.1

 # Installe une version spécifique  depuis GitHub
 - src: https://github.com/geerlingguy/ansible-role-passenger
   name: passenger
   version: 2.0.0

 # Installe une archive depuis un serveur web
 - src: https://www.example.com/ansible/roles/my-role-name.tar.gz
   name: my-role
```

Emplacements de recherche par défaut

1. Dans les collections
2. dans `./roles/`, relativement au fichier du playbook

```
$ tree -L 3
.
├── ansible.cfg
├── inventory
├── main_import.yml
├── main_include.yml
├── main_mix.yml
├── main_traditional.yml
└── roles
    └── role1
        ├── defaults
        ├── files
        ├── handlers
        ├── meta
        ├── README.md
        ├── tasks
        ├── templates
        ├── tests
        └── vars
```

3. dans la directive `roles_path`, par défaut :
   `~/.ansible/roles`
   `/usr/share/ansible/roles`
   `/etc/ansible/roles`

   Sinon personnalisable dans `ansible.cfg` :

```
[defaults]
roles_path = /path/to/roles
```

   (ou dans la variable `ANSIBLE_ROLES_PATH`)

4. dans le répertoire du playbook

Appeler des rôles depuis un playbook


```yaml
---
- name: Mon playbook
  hosts: all
  gather_facts: no


  roles:
    - role1
    - role2
    - role: '/chemin/vers/role3'

  tasks:
    - name: Suite des tâches
```

Avec la directive "roles"

```yaml
---
- name: Mon playbook
- hosts: all
  gather_facts: no


  tasks:
    - name: Import du role1
      ansible.builtin.import_role:
        name: role1
      vars:
        myvar: 'valeur'


    - name: Import du role2
      ansible.builtin.import_role:
        name: role2
```

Import (= statique)

```yaml
---
- name: Mon playbook
  hosts: all
  gather_facts: no


  tasks:
    - name: Include du role1
      ansible.builtin.include_role:
        name: role1

    - name: Include du role2
      ansible.builtin.include_role:
        name: role2
```

Include (= dynamique)

Débogage de rôle

- `ansible-playbook -vv` affiche l'emplacement du rôle exécuté

```
PLAYBOOK: main_traditional.yml ***************************
1 plays in ./main_traditional.yml

PLAY [Main playbook] *************************************

TASK [role1 : Play 1, task 1] ****************************
task path: /etc/ansible/roles/role1/tasks/main.yml:2
ok: [localhost] => {
    "msg": "This is play1, task1"
}

TASK [role2 : Play2, task 1] *****************************
task path: /etc/ansible/roles/role2/tasks/main.yml:2
ok: [localhost] => {
    "msg": "This is play2, task 1"
}
```

- Recommandé de nommer de façon explicite les tâches d'un rôle pour faciliter la compréhension à l'exécution

```
---
- name: APACHE RHEL | Installation Apache
  ansible.builtin.apt:
    name:
      - apache2
```

Résumé des commandes

Objet	Commande
Création de la structure de répertoire Créera le sous-répertoire ROLE_NAME avec les fichiers nécessaires. Recommandé : enlever le superflu	`ansible-galaxy [role] init ROLE_NAME`
Préparer un rôle pour distribution depuis le répertoire parent, le même où on a fait la commande d'init	`tar czvf ./ROLE_NAME.tar.gz ./ROLE_NAME`

Installer un rôle
 	
Par défaut, va dans `~/.ansible/roles`
	`ansible-galaxy install ./ROLE_NAME.tar.gz`
	
	`ansible-galaxy install ROLE_NAME`
	
	`[-p destination_path]`
	`[--server https://my.server.example.com]`

- guide utilisateur ansible-galaxy : https://docs.ansible.com/ansible/latest/cli/ansible-galaxy.html
- Documenter les rôles dans **meta/main.yml** et **README.md**

Résumé des commandes

ansible-galaxy role	Description
`init`	`ansible-galaxy role init MON_ROLE`
`remove`	efface un rôle en local
`delete`	efface un rôle de Galaxy Ansible `ansible-galaxy delete github_user github_repo`
`list`	Liste les rôles installés sur le système
`search`	`ansible-galaxy search cyberark`
`import`	Importe un rôle de GitHub vers galaxy.ansible.com (`ansible-galaxy login` d'abord)
`info`	Affiche des informations sur un rôle donné `ansible-galaxy role info geerlingguy.haproxy`
`install`	Installe un rôle en local, par défaut dans `~/.ansible/roles` Si nom de rôle seul, télécharge depuis Galaxy Ansible ou selon configuration de ansible.cfg

sur le site web Ubuntu : https://manpages.ubuntu.com/manpages/kinetic/en/man1/ansible-galaxy.1.html


Merci