CONDITIONNEMENT
CONDITIONNER L' EXÉCUTION D'UNE TÂCHE (1/4)

- A partir de l'évaluation d'une expression libre
  - Il faut répéter la condition pour toutes les tâches

Code : *exemple d'un play contenant un conditionnement*

```
---
- name: do some stuff
  command: /bin/foo
  register: cmd_result

- name: do some more stuff
  command: /usr/bin/do_stuff
  when: "'bar' in cmd_result.stdout"
```

On utilise ici le résultat d'un register pour conditionner une autre tâche

CONDITIONNEMENT
CONDITIONNER L'EXÉCUTION D'UNE TÂCHE (2/4)

- A partir d'un changement

Code : *exemple d'un play contenant un conditionnement*

```
---
- name: do some stuff
  command: /bin/foo
  register: cmd_result

- name: do some more stuff
  command: /usr/bin/do_stuff
  when: cmd_result.changed
```

Le register permet de récupérer l'état de fin d'une tâche

CONDITIONNEMENT
CONDITIONNER L'EXÉCUTION D'UNE TÂCHE (3/4)

- Définir un comportement par défaut

Code : *exemple d'un play contenant un conditionnement*

```
$ ansible-playbook -i inv.ini install.yml -e really_do_it=1
$ ansible-playbook -i inv.ini install.yml -e really_do_it=yes
$ ansible-playbook -i inv.ini install.yml -e really_do_it=true
```

Code : *exemple d'un play contenant un conditionnement*

```
---
- name: do some more stuff
  command: /usr/bin/do_stuff
  when: "really_do_it | default(false) | bool"
```

Les variables seront vues plus tard

CONDITIONNEMENT
CONDITIONNER L'EXÉCUTION D'UNE TÂCHE (4/4)

- A partir d'un fact

Code : *exemple d'un play contenant un conditionnement*

```
---
- name: do some more stuff
  command: /sbin/shutdown -t now
  when: ansible_os_familly == "Debian"
```

Utile pour conditionner des déploiements en fonction du type de VM sur laquelle le déploiement s'effectue.

CONDITIONNEMENT
CONDITIONNER L'ÉTAT CHANGED OU FAILED

notions avancées

- L'état changed ou failed d'un module / tâche peut être piloté par **changed_when** et **failed_when**

Code : *exemple pour conditionner changed_when*

```
---
- name: do some stuff
  command: /bin/foo
  register: result
  changed_when: "result.rc == 0 and 'OK' in result.stdout"
```

Utile pour gérer l'idempotence de module ne la supportant pas nativement (shell, command, …)

Code : *exemple pour conditionner failed_when*

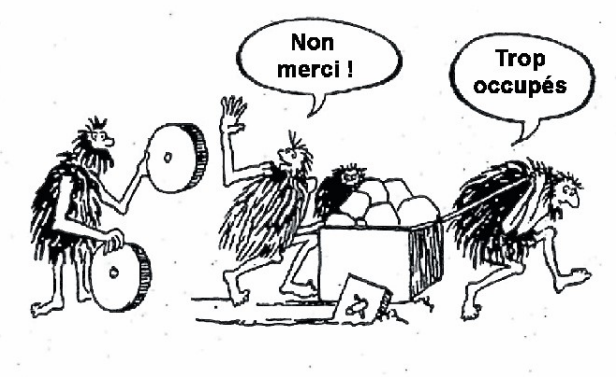
```
---
- name: S'assurer que /tmp a bien à minima 1Gb
  shell: "df -h /tmp|grep -v Filesystem|awk '{print $4}'|cut -d G -f1"
  register: tmpspace
  failed_when: "tmpspace.stdout | float < 1"
```

Utile pour s'assurer des prérequis d'une installation

RÔLES
GENERALITES (1/2)

- Les **rôles** ansible permettent de **réutiliser** du code ansible afin de développer plus rapidement des playbooks.

  A titre d'exemple, qq rôles possibles
  - installer un nginx,
  - installer une base de données postgresql,
  - …



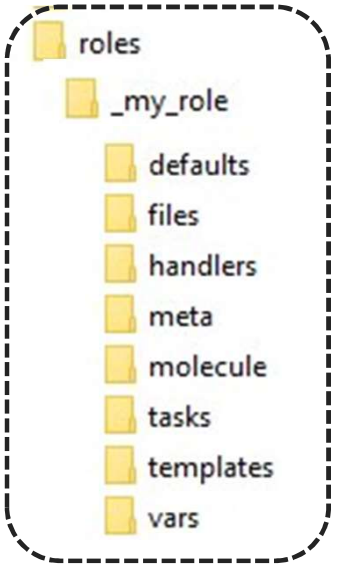
- Chaque rôle doit être **générique.**

- La **spécialisation** du rôle pour un projet s'appuiera sur
  - les **variables** du rôle qui seront passées en paramètres à l'appel du rôle
  - des **templates de fichiers** qui seront valorisés à l'exécution du rôle

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

RÔLES
GENERALITES (2/2)

- Chaque rôle ansible s'appuie sur **une structure normalisée de répertoires** au niveau de votre projet.
  - **Tous les rôles d'un projet** sont contenus au sein du **répertoire roles** du projet
  - Chacun des sous répertoires du rôle doivent contenir **un fichier main.yml** (à l'exclusion des sous répertoires templates & files).
  - **Seul le sous répertoire tasks est obligatoire.**



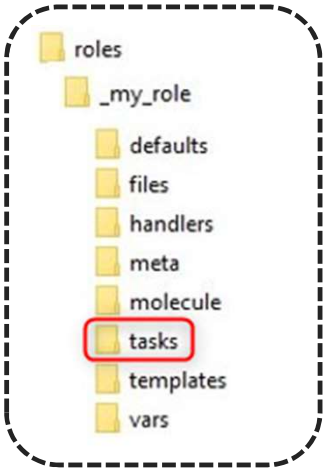
- Pour **initialiser la structure** de répertoire d'un rôle, utiliser **ansible-galaxy**

Code : *initialisation du role _my_role avec ansible-galaxy*

```
$ cd roles
$ ansible-galaxy init _my_role
```

Pour aller plus loin : https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html

RÔLES
TASKS (1/2)



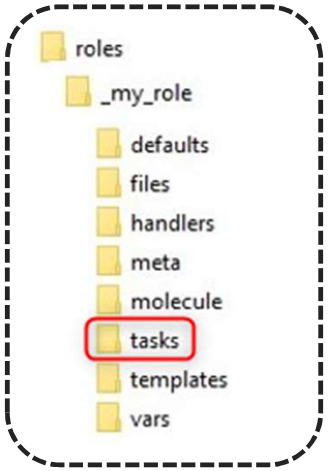
- Le fichier **tasks/main.yml** contient les différentes taches qui seront lancées au moment de l'appel du rôle dans le playbook

Code : *exemple de fichier tasks/main.yml*

```
---
- name: install httpd
  yum:
    name: httpd
    state: latest

- name: start and enable httpd service
  service:
    name: httpd
    state: started
    enabled: true

- name: ensure vhost directory is present
  file:
    path: "/var/www/vhosts/{{ ansible_hostname }}"
    state: directory
```

```
- name: deliver html content
  copy:
    src: web.html
    dest: "/var/www/vhosts/{{ ansible_hostname }}"

- name: template vhost file
  template:
    src: vhost.conf.j2
    dest: /etc/httpd/conf.d/vhost.conf
    owner: root
    group: root
    mode: 0644
  notify:
    - restart_httpd
```


RÔLES
TASKS (2/2)



Il est correct de mettre l'ensemble des taches d'un rôle au sein de tasks/main.yml

Toutefois, **pour des questions de maintenabilité,** il est fortement recommandé de **décomposer ces taches en fichiers** que l'on importe au sein du main.yml

Code : *exemple de fichier tasks/main.yml avec import*

```
- import_tasks: install_httpd.yml
- import_tasks: configure_vhost.yml
- import_tasks: deliver_content.yml
- import_tasks: start_httpd.yml
```

A noter : la résolution des fichiers se fait relativement au répertoire tasks du rôle

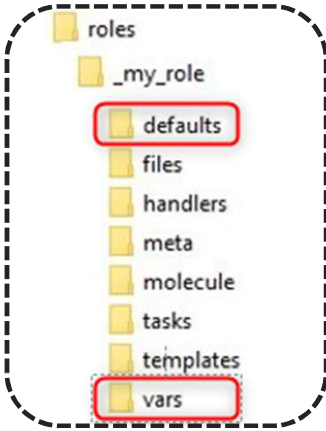
Différence entre import_* et include_* :

Pour l'import, l'ensemble des instructions est préprocessé au moment ou le playbook est parsé
Pour l'include, l'ensemble des instructions est processé au moment ou le playbook est exécuté

Code : *exemple de fichier tasks/main.yml avec include*

```
- include_tasks: '{{ ansible_os_family | lower }}/install_packages.yml'
```

RÔLES
DEFAULTS ET VARS



- Ces 2 sous répertoires d'un rôle contiennent les variables utiles pour le rôle

  - **defaults/main.yml** contient les **variables par défaut, surchargeables** pour un utilisateur du rôle

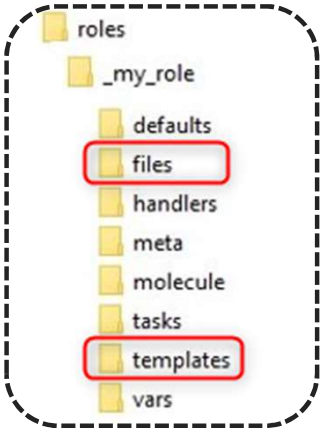
    exemple :
    - la version d'un progiciel,
    - le répertoire de déploiement d'un projet

    Code : *exemple de fichier defaults/main.yml*

    ```
    molecule__virtualenv_dir: ~/.venv/molecule
    ```

  - **vars/main.yml** contient **les variables internes** au rôle qui **ne peuvent pas être surchargées sans avoir une bonne connaissance du rôle**.

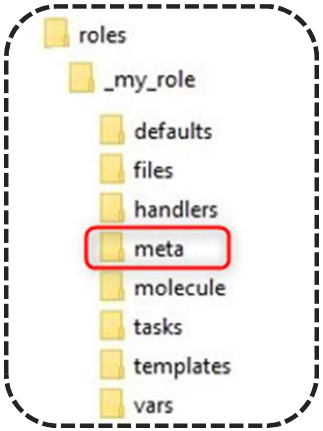
RÔLES
FILES ET TEMPLATES



- Ces 2 sous répertoires d'un rôle contiennent les fichiers utiles pour le rôle
  - **files** contient les fichiers statiques nécessaires aux tâches du rôle
  - **templates** contient les modèles Jinja2 qui seront valorisées dynamiquement par les taches du rôle

L'utilisation des **templates** Jinja2 sera abordé dans le **§ Templating & Variables**

RÔLES
META



- **meta/main.yml** contient :
  - les **dépendances** nécessaires au rôle
  - le nom de l'auteur,
  - le mode de licensing
  - les plateformes supportées par le rôle

Code : *exemple de* fichier meta/main.yml

```
galaxy_info:
  role_name: myrole
  author: myname
  license: mylicense
  min_ansible_version: 2.9
  platforms:
  - name: ubuntu
    versions:
    - 20.04
    - 18.04
dependencies:
  - role: ‘mydependency1‘
  - role: ‘mydependency2‘
```

A noter : les rôles mydependency1 et 2 seront lancés avant _my_role

RÔLES
META - PHILOSOPHIE

Avec Meta

- Vous n'avez pas nécessairement
  - l'expertise des stacks dont dépend votre rôle
  - La connaissance des policy à respecter
- Promeut
  - La modularité
  - La capitalisation de composant transverse

Tout est question d'équilibre

Sans Meta

- Lorsque vous avez une bonne maitrise du composant à déployer
- Permet d'avoir des rôles autonomes
- Réduit la complexité de vos playbooks
- Promeut la simplicité

RÔLES
HANDLERS

- **handlers/main.yml** contient les définitions des différents handlers du rôle, activés par l'instruction notify

Code : *exemple de taches déclenchant le handler*

```
- name:          Ensure SSH port is defined
  become:        yes
  lineinfile:
    dest:        '/etc/ssh/sshd_config'
    regexp:      '^Port'
    line:        'Port {{security_ssh__incoming_port}}'
  notify:        event__security_ssh__configuration_modified
  when:          security_ssh__allow_incoming_connection
```

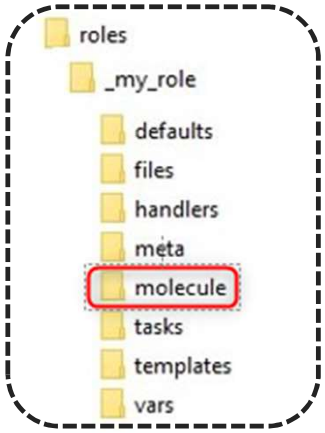
Code : *exemple de fichier handlers/main.yml*

```
- name:          on__security_ssh__configuration_modified
  become:        yes
  systemd:
    name:        ssh
    state:       reloaded
  listen:        event__security_ssh__configuration_modified
```

Proposition de convention de nommage
handler : on__{nom du role}__{nom de l'action}
notify : event __{nom du role}__{nom de l'action}

RÔLES
MOLECULE

notions avancées



- Ce répertoire contient l'ensemble des fichiers nécessaires aux **tests unitaires** du rôle s'appuyant sur le framework 'molecule'

L'utilisation des tests unitaires avec **Molecule** sera abordée dans le **§ Tests Unitaires**

RÔLES
UTILISATION AU SEIN D'UN PLAYBOOK

Code : *exemple de playbook déclenchant 2 rôles*

```
- name: ansible nodes post install
  hosts:
  - iac_servers
  roles:
  - automation/ansible_post_installation
  - automation/molecule_installation
```

A noter : par défaut, les rôles sont cherchés au sein du sous répertoire rôles, par rapport au playbook exécuté

Code : *exemple de playbook déclenchant 1 rôle avec passage de paramètre*

```
- name: molecule install
  hosts:
  - iac_servers
  roles:
  - role: automation/molecule_installation
    virtualenv_dir: ~/venv_mol
```

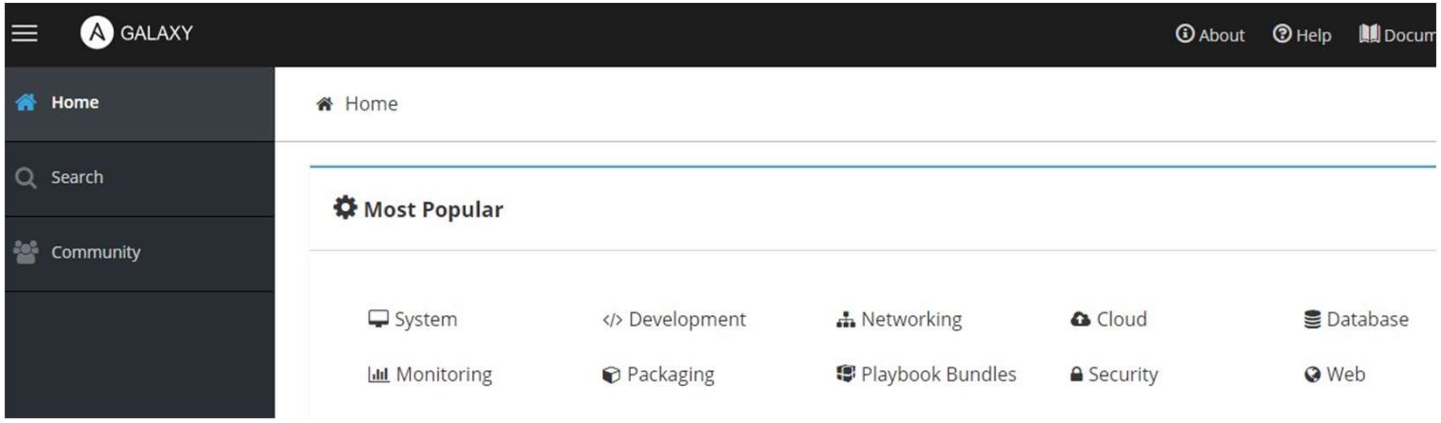
Idéalement, l'argument virtualenv_dir est défini dans defaults/main.yml du rôle, ce qui permet sa surcharge dans le cadre du playbook ou dans le cadre de l'inventaire

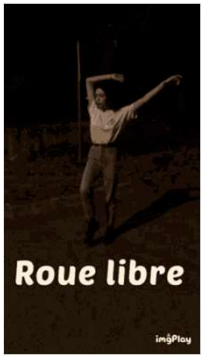
Si un rôle est déclaré plusieurs fois au sein d'un playbook, il ne sera exécuté qu'une seule fois, à moins que les paramètres du rôle ne soient différents à chaque déclaration

RÔLES
NE PAS REINVENTER LA ROUE

- Redhat met à disposition un site centralisant des rôles / collections mises à disposition par la communauté : https://galaxy.ansible.com/

  C'est l'équivalent de dockerhub pour Docker





- Une fois le rôle / collection trouvé, il ne reste plus qu'à l'installer sur le nœud ansible via une commande ansible-galaxy. La commande d'installation est précisée au sein de chaque rôle / collection.

Différence entre rôle et collection :
les collections sont un nouveau concept introduit dans les dernières versions d'Ansible. Voyez-le comme un rôle ++

RÉCAPITULATIF

Si le notify n’est pas déclenché, le handler est-il exécuté ?

Qu'est-ce qu’un handler ?

Qu'est-ce qu'un rôle ?

A quoi correspond les metas ?

Peut-on utiliser plusieurs rôles dans un playbook ?