



ANSIBLE



SOMMAIRE

- Présentation d'Ansible
 - Qu'est ce qu'Ansible et son but ?
 - Comment fonctionne t'il ?
- Installation et configuration d'Ansible
 - Installation sur différents systèmes d'exploitation
 - Configuration
- Prise en main d'Ansible
 - YAML
 - Inventaire
 - Playbook
 - Connexion
 - Commandes
- Utilisation d'Ansible
 - Modules
 - Variables
 - Conditions
 - Boucles
 - Rôles

PRÉSENTATION D'ANSIBLE

QU'EST CE QU'ANSIBLE ET QUEL EST SON BUT ?

- Outil open source d'automatisation informatique écrit en Python
- Automatise le provisionnement, la gestion des configurations, le déploiement des applications, l'orchestration, ...
- Simplifier l'automatisation

```
#!/bin/bash
# Script to add a user to Linux system
if [ $(id u) eq 0 ]; then
    username=johndoe
    if [ $? eq 0 ]; then
        echo "$username exists!"
        exit 1
    else
        useradd m p $username
    fi
fi
```

```
- name: Add the user
  ansible.builtin.user:
    name: johndoe
```

PRÉSENTATION D'ANSIBLE

COMMENT FONCTIONNE T'IL ?

- Peut s'exécuter depuis n'importe où
- Il existe des serveurs de déploiement Ansible tel que **Red Hat Ansible Automation Platform**
- Sans agent (Agentless)
- Se connecte via SSH (Linux) ou winrm (Windows)
- Cible les serveurs client via un fichier d'inventaire

INSTALLATION ET CONFIGURATION D'ANSIBLE

INSTALLATION

- Prérequis : Pip & Python 3 (Dernière version)
- PIP: `python3 -m pip install --user ansible`
- Pour s'assurer de l'installation :
`ansible --version`
- Mettre à jour :
`python3 -m pip install --upgrade --user ansible`

INSTALLATION ET CONFIGURATION D'ANSIBLE

INSTALLATION SUR OS SPÉCIFIQUE

- CentOS

```
Sudo yum install epel-release ansible
```

- Fedora

```
Sudo dnf install ansible
```

- Ubuntu

```
sudo apt update  
sudo apt install software-properties-common  
sudo add-apt-repository --yes --update ppa:ansible/ansible  
sudo apt install ansible
```

- Debian

Ajouter les sources dans `/etc/apt/sources.list` ou `/etc/apt/sources.list.d/ansible.list`

([Voir la page d'installation](#))

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-  
keys 93C4A3FD7BB9C367  
sudo apt update  
sudo apt install ansible
```

INSTALLATION ET CONFIGURATION D'ANSIBLE

CONFIGURATION

- Certains paramètres d'Ansible sont configurables via un fichier de configuration (ansible.cfg)
- La liste des éléments configuration se trouve ici :
 - https://docs.ansible.com/ansible/latest/reference_appendices/config.html#ansible-configuration-settings
- Il y a un ordre de priorité des variables :
 - ANSIBLE_CONFIG (Variable d'environnement si défini)
 - ansible.cfg (Dans le répertoire actuel)
 - ~/.ansible.cfg (Dans le répertoire utilisateur)
 - /etc/ansible/ansible.cfg

INSTALLATION ET CONFIGURATION D'ANSIBLE

EXERCICE

- Préparer une instance Debian 11
- Installer Ansible via PIP
- Valider son installation

PRISE EN MAIN

YAML

- Langage léger utilisé pour représenter des données structurées simple
- Basé sur le principe de **clé-valeur**
- Les valeurs peuvent être de différents types, comme des chaînes de caractères, des nombres, des booléens, etc

```
name: John
age: 30
is_student: false
```

- On peut aussi y retrouver des listes d'objet :

```
- name: John
  age: 30
  is_student: false
- name: Alice
  age: 21
  is_student: true
```

PRISE EN MAIN

INVENTAIRE

- Regroupe les informations des machines client
- Inventaire par défaut : /etc/ansible/hosts
- Possibilité de configurer des options de connections et des groupes d'host

- Format INI

```
server1.mycompany.com

[web]
server2.mycompany.com
server3.mycompany.com

[mail]
server4.mycompany.com
server5.mycompany.com
```

- Format YAML

```
all:
  hosts:
    server1.mycompany.com:
  children:
    web:
      hosts:
        server2.mycompany.com:
        server3.mycompany.com:
    bdd:
      hosts:
        server4.mycompany.com:
        server5.mycompany.com:
```

PRISE EN MAIN

INVENTAIRE

■ INI

```
server1 ansible_host=server1.mycompany.com
ansible_connection=ssh ansible_user=root
ansible_ssh_pass=LinPass
server2 ansible_host=server2.mycompany.com
ansible_connection=ssh ansible_user=root
ansible_ssh_pass=LinPass
server3 ansible_host=server3.mycompany.com
ansible_connection=winrm
ansible_user=administrator
ansible_ssh_pass=WinPass

[web]
server2

[bdd]
server3

[site:children]
Web
bdd
```

■ YAML

```
all:
  hosts:
    server1.mycompany.com:
  children:
    web:
      hosts:
        server2.mycompany.com:
          ansible_connection: ssh
          ansible_user: root
          ansible_ssh_pass: LinPass
    bdd:
      hosts:
        server3.mycompany.com:
          ansible_connection: winrm
          ansible_user: administrator
          ansible_password: WinPass
```

PRISE EN MAIN

CONNEXION

- Premier connexion ssh → check de la clés serveur cible
 - Connexion ssh manuel et accepter l'échange de clé
 - Ajouter l'option **host_key_checking = False** dans le fichier Ansible.cfg
 - Non recommandé
- Pour faciliter les échanges SSH, pensez à la copie de clés
 - `ssh-copy-id -i ~/.ssh/mykey user@host`

PRISE EN MAIN

PLAYBOOK

- Un playbook est un scénario qui décrira toutes les actions réalisées par les serveurs, incluant les paquets à installer, les fichiers à créer et les actions à réaliser sur les services, etc...
- Au format Yaml
- Un playbook est composé de :
 - Un nom
 - La cible des hosts
 - De variables (facultatif)
 - D'une liste d'instructions (Appel de module et roles)

```
- name: Maintenance on TSE
  hosts: all

# vars_prompt:
#   - name: "Message"
#     prompt: "Entrez votre message"
#     private: no
#     default: "localhost"

vars:
  Message: Toto

tasks:
  - name: Ping TSE
    win_ping:

  - name: send msg
    win_shell: msg.exe /server:vc65.labvmware.local * "{{ Message }}"
```

PRISE EN MAIN

PLAYBOOK

```
- name: Mysql
hosts: bdd

tasks:
- name: Install MySQL
  ansible.builtin.apt:
    name: mysql
    state: present

- name: Ensure Mysql is started and enabled at startup
  ansible.builtin.service:
    name: mysql
    state: started
    enabled: yes
```

PRISE EN MAIN

COMMANDES

- Pour vérifier la connectivité : `ansible -m ping all --one-line`
 - Cf : `Ansible <host> -m <module> -a arguments`
- Pour lancer un playbook : `ansible-playbook playbook.yml -i inventaire.yml`
- Si besoin de debug, ajouter `-vvvv` à la commande

PRISE EN MAIN

EXERCICE

- Préparer une instance Debian 11
- Installer Ansible via PIP
- Valider son installation
- Créez un inventaire d'hôtes et configurez une connexion à un hôte en utilisant un nom d'utilisateur et un mot de passe. Vérifiez que vous pouvez vous connecter à l'hôte en exécutant une commande de base, comme "uname -a".
 - Indice : modules nécessaire :
 - command
 - debug
 - register

UTILISATION D'ANSIBLE

MODULES

- Catégorisé par groupe en fonction des fonctionnalités
 - Liste des modules
- Des exemples de modules fréquemment utilisés on peut retrouver :
 - System
 - Mount
 - User
 - Commands
 - Command
 - Shell
 - Windows
 - Win_copy
 - Win_command
 - Files
 - Copy
 - Template
 - Packaging
 - Apt
 - yum

UTILISATION D'ANSIBLE

VARIABLES

- Déclarés dans les playbooks et les rôles
- Peuvent être défini à plusieurs niveaux :
 - La ligne de commande
 - Le playbook
 - Les host_vars (variable ciblant un host)
 - Les group_vars (variable ciblant un groupe d'host)
 - Les rôles

- Déclarés avec des doubles accolades

```
- name: Set Firewall Configurations
  hosts: web
  tasks:
    - firewallld:
      service: '{{ service_name }}'
      permanent: true
      state: enabled

    - firewallld:
      port: '{{ http_port }}'/tcp
      permanent: true
      state: disabled
```

- Défini par **key : value**

```
service_name: "https"
http_port: 8080
```

UTILISATION D'ANSIBLE

BOUCLES

- Plutôt que de créer plusieurs tâches répétitives, il est possible d'utiliser des boucles

- Une création d'utilisateur sans boucle

```
- name: Create users
hosts: localhost
tasks:
  - ansible.builtin.user:
      name: johndoe
  - ansible.builtin.user:
      name: marie
  - ansible.builtin.user:
      name: steven
  - ansible.builtin.user:
      name: seb
  - ansible.builtin.user:
      name: sophie
```

- Une création d'utilisateur avec boucle sur une liste

```
- name: Create users
hosts: localhost
tasks:
  - ansible.builtin.user:
      name: "{{ item }}"
      state: present
    loop:
      - johndoe
      - marie
      - steven
      - seb
      - sophie
```

UTILISATION D'ANSIBLE

BOUCLES

- Il est aussi possible de boucler sur des dictionnaires. Dans ce cas, il est nécessaire de préciser la clé utilisé après item
- La fonction **loop** fonctionne dans la majorité des cas.
Pour des boucles spécifiques, il peut être nécessaire d'utiliser la fonction **with_<lookup>**
Liste des boucles with <lookup>

```
- name: Create users
hosts: localhost
tasks:
  - ansible.builtin.user:
      name: "{{ item.name }}"
      uid: "{{ item.uid }}"
      state: present
    loop:
      - name: johndoe
        uid: 1010
      - name: marie
        uid: 1011
      - name: steven
        uid: 1012
      - name: seb
        uid: 1013
      - name: sophie
        uid: 1014
```

PRISE EN MAIN

EXERCICE

- Écrivez un playbook qui installe Apache sur un hôte cible et vérifiez que le service Apache est en cours d'exécution sur l'hôte cible.
- Y inclure une instruction qui copie un fichier depuis votre ordinateur local vers un hôte cible, puis modifie les permissions du fichier sur l'hôte cible.
- Écrivez un playbook qui crée un utilisateur et un groupe sur un hôte cible, puis ajoute l'utilisateur au groupe.
- Écrivez un playbook qui installe plusieurs packages sur plusieurs hôtes en utilisant une boucle.

UTILISATION D'ANSIBLE

CONDITIONS

- Pour éviter de créer plusieurs playbooks, on peut utiliser des conditions
 - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_vars_facts.html
 - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_conditionals.html
 - https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_tests.html#test-syntax
- Installation de Nginx sur 2 playbooks

```
- name: Install NGINX
hosts: debian_hosts
tasks:
- name: Install NGINX on Debian
  apt:
    name: nginx
    state: present
```

```
- name: Install NGINX
hosts: redhat_hosts
tasks:
- name: Install NGINX on Red_Hat
  yum:
    name: nginx
    state: present
```

- Même installation avec une condition et l'opérateur ==

```
- name: Install NGINX
hosts: all
tasks:
- name: Install NGINX on Debian
  apt:
    name: nginx
    state: present
  when: ansible_os_family == "Debian"
- name: Install NGINX on RedHat
  yum:
    name: nginx
    state: present
  when: ansible_os_family == "RedHat"
```

UTILISATION D'ANSIBLE

CONDITIONS

- Il existe d'autres opérateurs : or, and

```
- name: Install NGINX
hosts: all
tasks:
  - name: Install NGINX on Debian
    apt:
      name: nginx
      state: present
      when: ansible_os_family == "Debian" and
ansible_distribution_version == "11"
  - name: Install NGINX on RedHat & centos
    yum:
      name: nginx
      state: present
      when: ansible_os_family == "RedHat" or
ansible_os_family == "Centos"
```

UTILISATION D'ANSIBLE

CONDITIONS

■ Exemple d'une condition dans une boucle

```
- name: Install NGINX
hosts: all
vars:
  packages:
    - name: nginx
      required: True
    - name: mysql
      required: True
    - name: nginx
      required: False
tasks:
- name: Install "{{ item.name }}"
  apt:
    name: "{{ item.name }}"
    state: present
  when: item.required == True
  loop: "{{ packages }}"
```

■ Exemple d'une condition avec variables enregistrées

```
- name: Check status of a service and email if
its docker network
hosts: all
tasks:
- command: service httpd status
  register: result

- mail:
  to: admin@mycompany.com
  subject: Service Alert
  body: Httpd Service is down
  when: result.stdout.find('down') != -1
```

```
when: script_output.rc != 0
when: script_output is changed
```


PRISE EN MAIN

EXERCICE

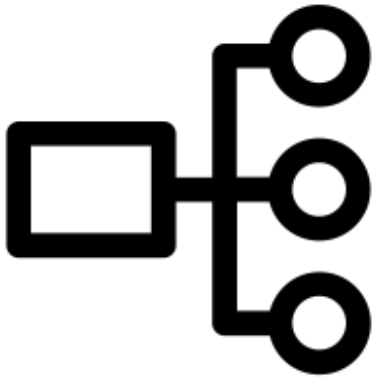
- Écrivez un playbook qui exécute une commande sur un hôte cible et enregistre la sortie de la commande dans un fichier sur votre ordinateur local.
- Écrivez un playbook qui déploie une application sur un hôte cible en utilisant des tâches conditionnelles pour vérifier si les prérequis de l'application sont déjà installés sur l'hôte cible.
- Écrivez un playbook qui exécute un script shell sur un hôte cible et envoie un rapport d'exécution par courriel à l'administrateur système.
- Créer un playbook qui déploie une application flask (python) - En utilisant des boucles, conditions et variables.
 - L'appli web sera exécuté à la fin du playbook et sera accessible

UTILISATION D'ANSIBLE

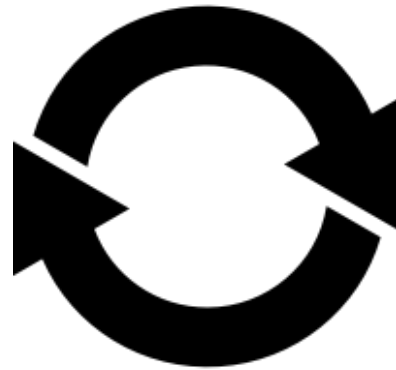
ROLES

- Ce sont des ensembles de tâches regroupés par thématique

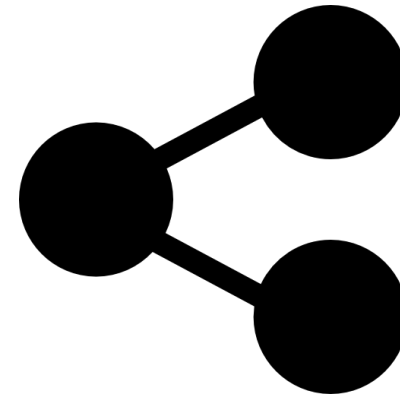
- d'organiser le code



- Le rendre réutilisable



- Le partager



```
- name: Install and Configure MySQL
hosts: db-server
tasks:
  #Install MySQL
  - name: Install Pre-Requisites
    yum: name=pre-req-packages state=present
  - name: Install MySQL Packages
    yum: name=mysql state=present
  - name: Start MySQL Service
    service: name=mysql state=started
  - name: Configure Database
    mysql_db: name=db1 state=present
  # Install Nginx
  - name: Install Pre-Requisites
    yum: name=pre-req-packages state=present
  - name: Install Nginx Packages
    yum: name=Nginx state=present
  - name: Start Nginx Service
    service: name=Nginx state=started
  - name: Configure Database
    copy: src=nginx.conf dest=/etc/nginx.conf
```

UTILISATION D'ANSIBLE

ROLES

- `./playbook/install.yml`

```
- name: Install and Configure MySQL & nginx
  hosts: db-server
  roles:
    - mysql
    - nginx
```

- `./role/Nginx-Role/tasks/main.yml`

```
---
- name: Install Pre-Requisites
  yum: name=pre-reg-packages state=present
- name: Install Nginx Packages
  yum: name=Nginx state=present
- name: Start Nginx Service
  service: name=Nginx state=started
- name: Configure Database
  copy: src=nginx.conf dest=/etc/nginx.conf
```

- `./role/Mysql-Role/tasks/main.yml`

```
---
- name: Install Pre-Requisites
  yum: name=pre-reg-packages state=present
- name: Install MySQL Packages
  yum: name=mysql state=present
- name: Start MySQL Service
  service: name=mysql state=started
- name: Configure Database
  mysql_db: name=db1 state=present
```

UTILISATION D'ANSIBLE

ROLES

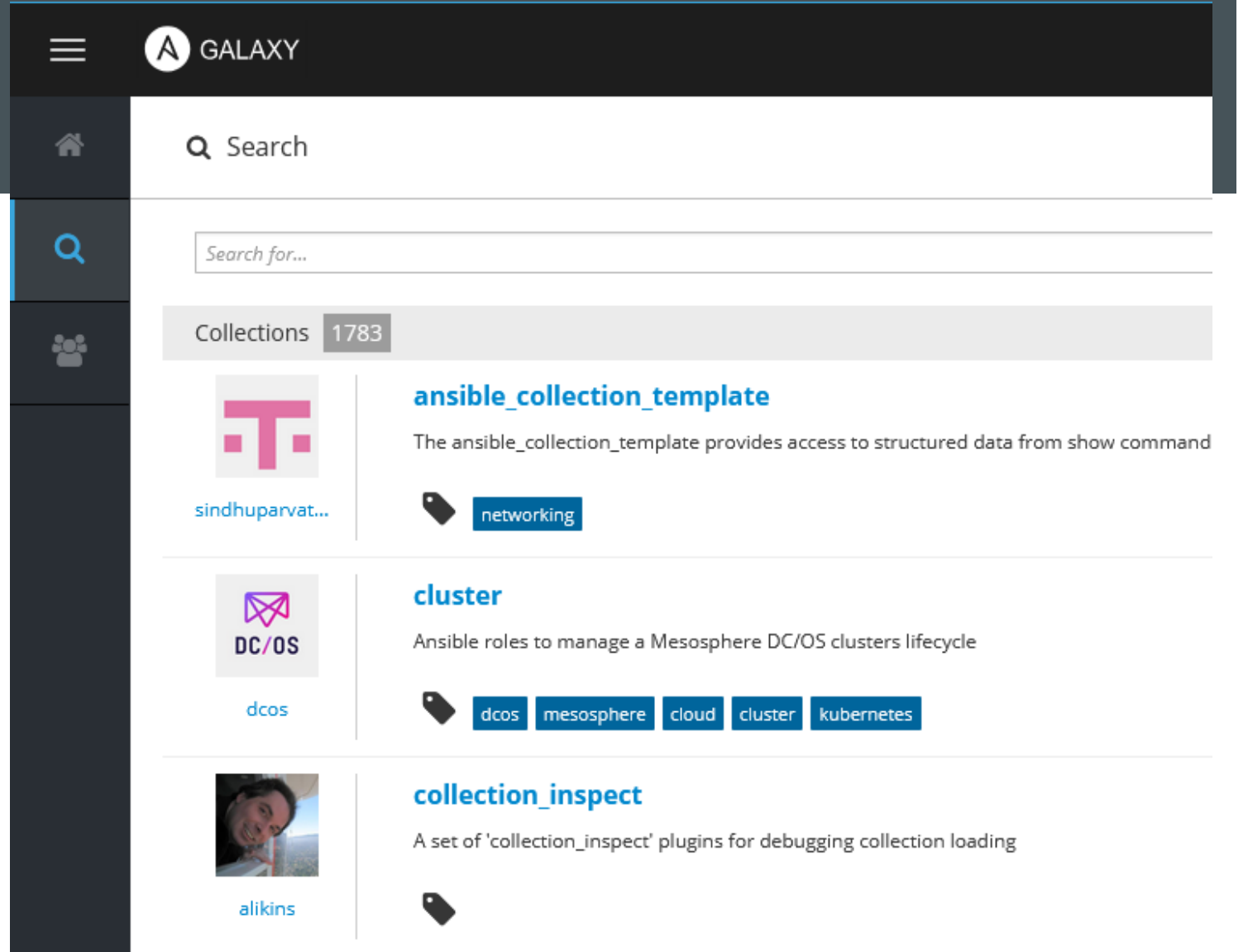
- Ils se structurent de la manière suivantes :

```
roles/
  common/          # this hierarchy represents a "role"
    tasks/         #
      main.yml      # <-- tasks file can include smaller files if warranted
    handlers/      #
      main.yml      # <-- handlers file
    templates/     # <-- files for use with the template resource
      ntp.conf.j2   # <----- templates end in .j2
    files/         #
      bar.txt       # <-- files for use with the copy resource
      foo.sh        # <-- script files for use with the script resource
    vars/          #
      main.yml      # <-- variables associated with this role
    defaults/      #
      main.yml      # <-- default lower priority variables for this role
    meta/          #
      main.yml      # <-- role dependencies
    library/       # roles can also include custom modules
    module_utils/  # roles can also include custom module_utils
    lookup_plugins/ # or other types of plugins, like lookup in this case

  webtier/         # same kind of structure as "common" was above, done for the webtier role
  monitoring/      # ""
```

UTILISATION D'ANSIBLE ROLES

- Partage avec la communauté
- Disponible principalement sur Ansible Galaxy ou bien GitHub



The screenshot displays the Ansible Galaxy web application. The top navigation bar includes a hamburger menu, the 'A GALAXY' logo, and a search bar. A left sidebar contains icons for home, search, and a community group. The main content area shows a search bar with the placeholder 'Search for...'. Below this, a 'Collections' section indicates 1783 items. Three collections are listed:

- ansible_collection_template** by [sindhuparvat...](#): The ansible_collection_template provides access to structured data from show command. Tag: **networking**.
- cluster** by [dcos](#): Ansible roles to manage a Mesosphere DC/OS clusters lifecycle. Tags: **dcos**, **mesosphere**, **cloud**, **cluster**, **kubernetes**.
- collection_inspect** by [alikins](#): A set of 'collection_inspect' plugins for debugging collection loading. Tag: **collection_inspect**.

UTILISATION D'ANSIBLE

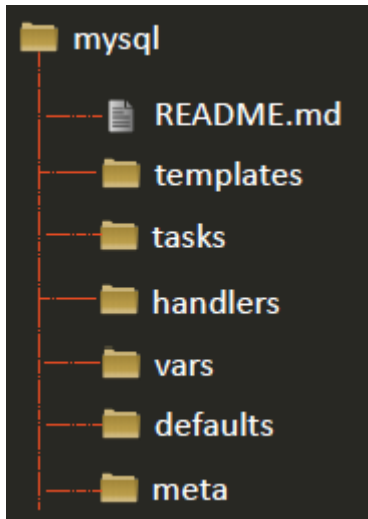
ROLES

- Le chemin des rôles peut être configuré dans le fichier **ansible.cfg**

`roles_path = /etc/ansible/roles`

Initier un rôle

```
ansible-galaxy init mysql
```



Récupérer un rôle

```
ansible-galaxy search mysql
```

Found 1126 roles matching your search. Showing first 1000.

Name	Description
-----	-----
Outsider.ansible_zabbix_agent	Installing and maintaining zab
1mr.unattended	install and configure unattended up
1nfinitum.mysql	Simply installs MySQL 5.7 on Xenial.

```
ansible-galaxy install 1nfinitum.mysql
```

UTILISATION D'ANSIBLE

ROLES

- Liste les rôles installés

```
Ansible-galaxy list
```

- Affiche la config d'Ansible

```
Ansible-config dump
```

- Installe un rôle dans un dossier spécifique

```
Ansible-galaxy install infinitem.mysql -p ./roles
```


PRISE EN MAIN

EXERCICE

- Créer un rôle pour installer Apache