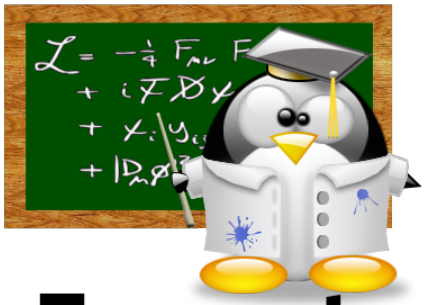

Formatux

TP Ansible Niveau 2 (corrections)

Table des matières

🎓 Objectifs

- ☐ Utiliser les variables ;
- ☐ Mettre en oeuvre des boucles et des conditions ;
- ☐ Gérer les fichiers ;
- ☐ Envoyer des notifications et réagir ;
- ☐ Gérer les fichiers ;
- ☐ Créer des tâches asynchrones.

Module 1 : Les variables

Objectifs

- Utiliser les variables dans un playbook

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_variables.html

Exercice 1.1 : Affichage d'une variable globale

Ecrire un playbook `play-vars.yml` permettant d'afficher le nom de la distribution de la cible ainsi que sa version majeure, en utilisant des variables globales :

- Pour trouver la variable globale à utiliser :

```
$  ansible ansiblecli -m setup | grep "distribution"
        "ansible_distribution": "CentOS",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/redhat-release",
        "ansible_distribution_file_variety": "RedHat",
        "ansible_distribution_major_version": "7",
        "ansible_distribution_release": "Core",
        "ansible_distribution_version": "7.4.1708",
```

Créer le fichier `play-vars.yml` avec ce contenu :

```
$ vim play-vars.yml
---
# ======================================================================
# Exercice : 1.1
# Objectif : Afficher la distribution du client
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true

  tasks:
  - name: afficher les variables globales
    debug:
      msg: "La distribution est une {{ ansible_distribution }} version {{
ansible_distribution_major_version }}"
```

Executer le playbook :

```
$ ansible-playbook play-vars.yml

PLAY [ansiblecli] ***************************************************************

TASK [Gathering Facts] **********************************************************
ok: [192.168.10.11]

TASK [afficher les variables globales] ******************************************
ok: [192.168.10.11] => {
    "msg": "La distribution est une CentOS version 7"
}

PLAY RECAP **********************************************************************
192.168.10.11              : ok=2    changed=0    unreachable=0    failed=0
```

Exercice 1.2 : Utilisation d'une variable locale

- Créer une variable locale nommée `port_http` contenant la valeur `8080`. Utiliser cette variable dans un message :

```
---
# ======================================================================
# Exercice : 1.2
# Objectif : Utiliser une variable locale
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    port_http: 8080

  tasks:

  - name: afficher les variables locales
    debug:
      msg: "Le port {{ port_http }} sera utilisé"
```

Résultat :

```
ansible-playbook play-vars.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [afficher les variables locales] ****************************************
ok: [192.168.10.11] => {
    "msg": "Le port 8080 sera utilisé"
}

PLAY RECAP *******************************************************************
192.168.10.11              : ok=2    changed=0    unreachable=0    failed=0
```

Exercice 1.3 : Utilisation d'une variable locale de type dictionnaire

- Créer une variable nommée `service` de type dictionnaire avec comme entrées :

```
service:
  name: apache
  rpm: httpd
```

- Créer une tâche permettant d'afficher les entrées du dictionnaire.

```
---
# ======================================================================
# Exercice : 1.3
# Objectif : Utiliser une variable de type dictionnaire
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    port_http: 8080
    service:
      name: apache
      rpm: httpd

  tasks:

  - name: installation du logiciel
    debug:
      msg: "Le logiciel {{ service['name'] }} sera installé par le paquet {{
service.rpm }}"
```

```
$ ansible-playbook play-vars.yml

PLAY [ansiblecli] ***************************************************************

TASK [Gathering Facts] **********************************************************
ok: [192.168.10.11]


TASK [installation du logiciel] *************************************************
ok: [192.168.10.11] => {
    "msg": "Le logiciel apache sera installé par le paquet httpd"
}

PLAY RECAP **********************************************************************
192.168.10.11              : ok=2    changed=0    unreachable=0    failed=0
```

Exercice 1.4 : Utilisation d'une variable locale de type dictionnaire à plusieurs dimensions

- En vous inspirant de l'exercice précédent, créer un dictionnaire de dictionnaires. La variable `service` aura alors cette structure :

```
service:
  web:
    name: apache
    rpm: httpd
  db:
    name: mariadb
    rpm: mariadb-server
```

- Utiliser cette structure pour afficher selon une variable `type` les entrées du dictionnaire `web` ou `db`.

```
---
# ======================================================================
# Exercice : 1.4
# Objectif : Utiliser un dictionnaire de dictionnaires
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    port_http: 8080
    type: web
    service:
      web:
        name: apache
        rpm: httpd
      db:
        name: mariadb
        rpm: mariadb-server

  tasks:

  - name: afficher un dictionnaire
    debug:
      msg: "Le logiciel {{ service[type]['name'] }} sera installé par le paquet {{
service[type].rpm }}"
```

Exercice 1.5 : Surcharger une variable locale par la ligne de commande

- Surcharger la variable `type` de l'exercice 1.4 en utilisant la ligne de commande :

```
$ ansible-playbook --extra-vars "type=db" play-vars.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [afficher une variable globale] *****************************************
ok: [192.168.10.11] => {
    "msg": "La distribution est une CentOS version 7"
}

TASK [afficher une variable locale] ******************************************
ok: [192.168.10.11] => {
    "msg": "Le port 8080 sera utilisé"
}

TASK [afficher un dictionnaire] **********************************************
ok: [192.168.10.11] => {
    "msg": "Le logiciel mariadb sera installé par le paquet mariadb-server"
}

PLAY RECAP *******************************************************************
192.168.10.11              : ok=4    changed=0    unreachable=0    failed=0


TASK [installation du logiciel] **********************************************
ok: [192.168.10.11] => {
    "msg": "Le logiciel apache sera installé par le paquet httpd"
}

PLAY RECAP *******************************************************************
192.168.10.11              : ok=4    changed=0    unreachable=0    failed=0
```

Exercice 1.6 : Externaliser les variables

- Créer un fichier `vars.yml` contenant les variables définies précédement :

```
---
port_http: 8080
type: web
service:
  web:
    name: apache
    rpm: httpd
  db:
    name: mariadb
    rpm: mariadb-server
```

- Utiliser ce fichier de variables depuis le fichier `play-vars.yml`

```
---
# ======================================================================
# Exercice : 1.6
# Objectif : Externaliser les variables
# Version  : 1.0
# ======================================================================
- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars_files:
    - vars.yml

  tasks:

  - name: afficher les variables globales
    debug:
      msg: "La distribution est une : {{ ansible_distribution }} version {{
ansible_distribution_major_version }}"

  - name: afficher les variables locales
    debug:
      msg: "Le port {{ port_http }} sera utilisé"

  - name: afficher un dictionnaire
    debug:
      msg: "Le logiciel {{ service[type]['name'] }} sera installé par le paquet {{
service[type].rpm }}"
```

- Tester le bon fonctionnement, avec et sans la surcharge de variable :

```
$ ansible-playbook --extra-vars "type=db" play-vars.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [afficher une variable globale] *****************************************
ok: [192.168.10.11] => {
    "msg": "La distribution est une CentOS version 7"
}

TASK [afficher une variable locale] ******************************************
ok: [192.168.10.11] => {
    "msg": "Le port 8080 sera utilisé"
}

TASK [afficher un dictionnaire] **********************************************
ok: [192.168.10.11] => {
    "msg": "Le logiciel mariadb sera installé par le paquet mariadb-server"
}

PLAY RECAP *******************************************************************
192.168.10.11              : ok=4    changed=0    unreachable=0    failed=0
```

```
$ ansible-playbook play-vars.yml
PLAY [ansiblecli] ****************************************************************

TASK [Gathering Facts] ***********************************************************
ok: [192.168.10.11]

TASK [afficher une variable globale] *********************************************
ok: [192.168.10.11] => {
    "msg": "La distribution est une CentOS version 7"
}

TASK [afficher une variable locale] **********************************************
ok: [192.168.10.11] => {
    "msg": "Le port 8080 sera utilisé"
}

TASK [afficher un dictionnaire] **************************************************
ok: [192.168.10.11] => {
    "msg": "Le logiciel apache sera installé par le paquet httpd"
}

PLAY RECAP ***********************************************************************
192.168.10.11              : ok=4    changed=0    unreachable=0    failed=0
```

Module 2 : La gestion des boucles

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_loops.html

Exercice 2.1 : Créer des utilisateurs en boucle

- Créer un playbook `play-users.yml` qui créera 3 utilisateurs. Ces trois utilisateurs appartiennent au groupe `users`.

```
---
# ======================================================================
# Exercice : 2.1
# Objectif : Créer des utilisateurs en utilisant une 'boucle'
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    users:
      - antoine
      - erwan
      - philippe

  tasks:

  - name: creer les utilisateurs
    user:
      name: "{{ item }}"
      group: "users"
      state: "present"
    with_items: "{{ users }}"
```

Exercice 2.2 : Créer des utilisateurs en boucle (version avancée)

- Reprendre l'exercice précédent et l'améliorer.

Vous utiliserez un fichier de variables externe `users.yml` qui contiendra un dictionnaire d'utilisateurs. Chaque utilisateur aura comme propriété : un nom, un groupe, un état (present,absent).

Le fichier de variables users.yml

```
users:
  antoine:
    group: users
    state: present
  erwan:
    group: users
    state: present
  philippe:
    group: users
    state: absent
```

La boucle utilisée sera de type `with_dict: "{{ users }}"`.

- Le nom de l'utilisateur sera récupéré avec la variable `{{ item.key }}`.
- Le groupe sera récupéré avec la variable `{{ item.value.group }}`, l'état avec `{{ item.value.state }}`.

Le playbook play-users.yml

```
---
# ======================================================================
# Exercice : 2.1
# Objectif : Créer des utilisateurs en utilisant une 'boucle' et un
#            dictionnaire
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars_files:
    - users.yml

  tasks:

  - name: creer les utilisateurs
    user:
      name: "{{ item.key }}"
      group: "{{ item.value.group }}"
      state: "{{ item.value.state }}"
    with_dict: "{{ users }}"
```

Exercice 2.3 : Afficher les valeurs d'un dictionnaire en boucle

- Reprendre l'exercice 1.6 en vous inspirant de l'exercice précédent. Le playbook permettra d'afficher plusieurs services.

```
  - name: afficher un dictionnaire
    debug:
      msg: "Le logiciel {{ service[item]['name'] }} sera installé par le paquet {{
service[item].rpm }}"
    with_items: "{{ type }}"
```

Module 3 : Les conditions

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_conditionals.html

Exercice 3.1 : Installation de logiciel avec conditions

En vous inspirant de l'exercice 1.6 :

- créer deux tâches, la première permettant d'installer le service web, la seconde le service de base de données
- en utilisant le dictionnaire `service`
- en fonction de la valeur de la variable `type` fournie en arguments à la ligne de commande
- uniquement sur les distribution CentOS

```
---
type: web
service:
  web:
    name: apache
    rpm: httpd
  db:
    name: mariadb
    rpm: mariadb-server
```

```
---
# ======================================================================
# Exercice : 3.1
# Objectif : Utiliser une condition when
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars_files:
    - vars.yml

  tasks:

  - name: installer le service de bdd
    yum:
      name: "{{ service['db']['rpm'] }}"
      state: latest
    when: (type == "db" and ansible_distribution == "CentOS")

  - name: installer le service web
    yum:
      name: "{{ service['web']['rpm'] }}"
      state: latest
    when:
      - type == "web"
      - ansible_distribution == "CentOS"
```

```
$ ansible-playbook --extra-vars "type=web" play-vars.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [installer le service de bdd] *******************************************
skipping: [192.168.10.11]

TASK [installer le service web] **********************************************
changed: [192.168.10.11]

PLAY RECAP *******************************************************************
192.168.10.11              : ok=2    changed=1    unreachable=0    failed=0

$ ansible-playbook --extra-vars "type=db" play-vars.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [installer le service de bdd] *******************************************
changed: [192.168.10.11]

TASK [installer le service web] **********************************************
skipping: [192.168.10.11]

PLAY RECAP *******************************************************************
192.168.10.11              : ok=2    changed=1    unreachable=0    failed=0
```

Module 4 : La gestion des fichiers

Plus d'informations sur http://docs.ansible.com/ansible/latest/list_of_files_modules.html

Exercice 4.1 : Utiliser le module ini_file

- Modifier le fichier `/etc/yum.conf` pour exclure le kernel des paquets mis à jour.

Pour cela :

- vous allez utiliser le module `ini_file` pour modifier le fichier de configuration de yum.
- le fichier `/etc/yum.conf` doit contenir :

```
[main]
...
exclude=kernel*
```

Il n'y a pas d'espace avant et après le '=' d'une option dans le fichier `/etc/yum.conf`. Vous allez donc devoir utiliser le paramètre `no_extra_spaces=True` du module `ini_file`.

```
---
# ======================================================================
# Exercice : 4.1
# Objectif : Gérer des fichiers ini
# Version  : 1.0
# ======================================================================

- hosts: ansiblesrv
  remote_user: ansible
  become: true
  vars:
    - yum_conf : /etc/yum.conf

  tasks:

  - name: exclure le kernel des mises a jour
    ini_file: path="{{ yum_conf }}" no_extra_spaces=True section=main option=exclude
value=kernel*
```

Exercice 4.2 : Utiliser le module lineinfile

- Créer un nouveau playbook qui permettra d'exclure également les paquets `java*` de la mise à jour avec le module `lineinfile`.

Le module `ini_file` permet d'ajouter une option dans une section précise d'un fichier `ini` contrairement à `lineinfile` qui est moins précis de ce point de vue là.

Maintenant que nous savons que le fichier contient la ligne `exclude=` nous pouvons utiliser le module `lineinfile` pour la modifier.

```
---
# ======================================================================
# Exercice : 4.2
# Objectif : Modifier une ligne dans un fichier
# Version  : 1.0
# ======================================================================

- hosts: ansiblesrv
  remote_user: ansible
  become: true
  vars:
    - yum_conf : /etc/yum.conf

  tasks:

  - name: s'assurer que le java est aussi exclu
    lineinfile:
      path: "{{ yum_conf }}"
      regexp: '^exclude='
      line: 'exclude=kernel* java*'
```

Exercice 4.3 : Utiliser le module copy

Depuis le serveur Ansible :

- Créer un fichier local `testfile.txt`
- Créer un playbook `sendfile.yml` qui envoie ce fichier vers le client

```
---
# ======================================================================
# Exercice : 4.3
# Objectif : Copier un fichier vers un client
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    - source: testfile.txt
    - destination: /home/ansible/test.txt

  tasks:

  - name: copie un fichier vers le client
    copy:
      src: "{{ source }}"
      dest: "{{ destination }}"
      owner: ansible
      group: users
      mode: 0640
```

- Modifier le fichier sur le serveur, envoyer le fichier à nouveau sur le client
- Modifier le fichier sur le client, envoyer le fichier à nouveau sur le client. Vérifier.

Exercice 4.4 : Utiliser le module fetch

- Créer un playbook `getfile.yml` qui copie sur le serveur le fichier `testfile.txt` de l'exercice précédent présent sur le client.

```
---
# ======================================================================
# Exercice : 4.4
# Objectif : Copier un fichier depuis un client
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    - source: /home/ansible/test.txt
    - destination: test-{{ inventory_hostname }}.txt

  tasks:

  - name: copie un fichier depuis le client
    fetch:
      src: "{{ source }}"
      dest: "{{ destination }}"
      flat: yes
```

Exercice 4.5 : Déployer un template

Les templates jinja feront l'objet d'un module plus complet sur la question.

- Sur le serveur Ansible, créer le modèle suivant :

```
Je suis un fichier de test
Je suis déployé sur le serveur {{ inventory_hostname  }}
```

- Créer un playbook sendtemplate.yml permettant de déployer ce template

```
---
# ======================================================================
# Exercice : 4.5
# Objectif : Deployer un template vers un client
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    - source: test-template.j2
    - destination: /home/ansible/test.txt

  tasks:

  - name: copie un modele vers le client
    template:
      src: "{{ source }}"
      dest: "{{ destination }}"
      owner: ansible
      group: users
      mode: 0640
```

- Vérifier sur le client

```
$ cat test.txt
Je suis un fichier de test
Je suis déployé sur le serveur 192.168.10.11
```

Exercice 4.6 : Utiliser le module get_url

- Créer un playbook geturl.yml permettant de télécharger sur le client le fichier https://ftp.drupal.org/files/projects/drupal-8.4.4.tar.gz

```
---
# ======================================================================
# Exercice : 4.6
# Objectif : Telecharger un fichier sur client
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true
  vars:
    - source: https://ftp.drupal.org/files/projects/drupal-8.4.4.tar.gz
    - destination: /home/ansible/

  tasks:

  - name: copie un fichier vers le client
    get_url:
      url: "{{ source }}"
      dest: "{{ destination }}"
      owner: ansible
      group: users
      mode: 0640
```

Module 5 : Les handlers

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_intro.html#handlers-running-operations-on-change

Exercice 5.1 : Gestion des handlers

- Créer un playbook qui :
  - installe httpd,
  - démarre le service,
  - modifie la valeur du ServerAdmin dans `/etc/httpd/conf/httpd.conf`
  - redémarre le service si besoin

Lancer plusieurs fois votre playbook en changeant la valeur du ServerAdmin et vérifier que le Handler soit bien exécuté !

```
---
# ======================================================================
# Exercice : 5.1
# Objectif : Telecharger un fichier sur client
# Version  : 1.0
# ======================================================================


- hosts: ansiblecli
  remote_user: ansible
  become: true

  tasks:

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

  - name: demarre httpd
    systemd:
      name: httpd
      state: started

  - name: modifie le serveradmin
    lineinfile:
      path: /etc/httpd/conf/httpd.conf
      regexp: '^ServerAdmin'
      line: 'ServerAdmin test1@fr.clara.net'
    notify:
      - redemarre httpd

  handlers:
    - name: redemarre httpd
      systemd:
        name: httpd
        state: restarted
```

```
ansible-playbook deploy-httpd.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [install httpd] *********************************************************
ok: [192.168.10.11]

TASK [demarre httpd] *********************************************************
ok: [192.168.10.11]

TASK [modifie le serveradmin] ************************************************
changed: [192.168.10.11]

RUNNING HANDLER [redemarre httpd] ********************************************
changed: [192.168.10.11]

PLAY RECAP *******************************************************************
192.168.10.11              : ok=5    changed=2    unreachable=0    failed=0
```

Module 6 : Les rôles

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_reuse_roles.html

Exercice 6.1 : Créer un rôle

- Créer un rôle en reprenant l'exercice précédent
- Initialiser le rôle :

```
$ ansible-galaxy init serveurweb
```

- Définir les variables du rôle dans le fichier `./serveurweb/vars/main.yml` :

```
# vars file for serveurweb
service: httpd
rpm: httpd
admin: test@fr.clara.net
```

- Définir les handlers du rôle dans le fichier `./serveurweb/handlers/main.yml ` :

```
---
# handlers file for serveurweb
- name: redemarre httpd
  systemd:
    name: httpd
    state: restarted
```

- Définir les tâches qui composent le rôle dans le fichier `./serveurweb/tasks/main.yml ` :

```
---
# tasks file for serveurweb
- name: install httpd
  yum:
    name: "{{ rpm }}"
    state: latest

- name: demarre httpd
  systemd:
    name: "{{ service }}"
    state: started

- name: modifie le serveradmin
  lineinfile:
    path: /etc/httpd/conf/httpd.conf
    regexp: '^ServerAdmin'
    line: "ServerAdmin {{ admin }}"
  notify:
  - redemarre httpd
```

- Définir notre playbook `deploy-httpd-role.yml` qui utilise le rôle :

```
---
- hosts: ansiblecli
  remote_user: ansible
  become: true
  roles:
    - serveurweb
```

- Exécuter le playbook :

```
$  ansible-playbook deploy-httpd-role.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [serveurweb : install httpd] ********************************************
ok: [192.168.10.11]

TASK [serveurweb : demarre httpd] ********************************************
ok: [192.168.10.11]

TASK [serveurweb : modifie le serveradmin] ***********************************
changed: [192.168.10.11]

RUNNING HANDLER [serveurweb : redemarre httpd] *******************************
changed: [192.168.10.11]

PLAY RECAP *******************************************************************
192.168.10.11              : ok=5    changed=2    unreachable=0    failed=0
```

Module 7 : Les tâches asynchrones

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_async.html

Exercice 7.1 : Attendre la fin d'une tâche

- Créer un playbook qui met à jour le client, le redémarre sans afficher de message d'erreur.

Ici toute l'astuce est d'attendre 2s puis de lancer le reboot pour éviter que le serveur ne coupe la connexion avant la fin du playbook !

```
---
# ======================================================================
# Exercice : 7.1
# Objectif : Attendre la fin d'une tâche
# Version  : 1.0
# ======================================================================

- hosts: ansiblecli
  remote_user: ansible
  become: true

  tasks:

  - name: mise a jour du systeme
    yum: name=* state=latest

  - name: redemarrage
    shell: sleep 2 && reboot
    async: 1
    poll: 0
    ignore_errors: true
```

```
ansible-playbook patchmanagement.yml

PLAY [ansiblecli] ************************************************************

TASK [Gathering Facts] *******************************************************
ok: [192.168.10.11]

TASK [mise a jour du systeme] ************************************************
ok: [192.168.10.11]

TASK [redemarrage] ***********************************************************
changed: [192.168.10.11]

PLAY RECAP *******************************************************************
192.168.10.11              : ok=3    changed=1    unreachable=0    failed=0
```