Atelier 7 - Configuration d'un serveur LAMP

Objectifs

- Écrire un inventaire Ansible ciblant une VM Vagrant.
- Écrire un playbook pour installer Apache, MySQL, PHP (LAMP) de façon idempotente.
- Utiliser un *handler* pour redémarrer Apache après changement de config.
- Transformer le playbook en un **role** réutilisable.

Hypothèses

- VM Vagrant basée sur Ubuntu/Debian (ex : `ubuntu/bionic64` ou `ubuntu/focal64`).
- Vous avez un control-node (poste formateur) avec Ansible installé.
- Niveau : intermédiaire.

Arborescence recommandée pour l'atelier

```
ansible-lamp-workshop/
├─ Vagrantfile
├─ inventory
├─ ansible.cfg
├─ site.yml              # playbook monolithique (exercise 2)
├─ files/
│   └─ index.html        # page test
├─ roles/
│  └─ lamp/              # role (exercise 4)
│     ├─ defaults/
│     │  └─ main.yml
│     ├─ tasks/
│     │  ├─ main.yml
│     │  └─ mysql.yml
│     ├─ handlers/
│     │  └─ main.yml
│     ├─ templates/
│     │  └─ 000-default.conf.j2
│     └─ files/
│        └─ index.html
```

Préparation rapide (contrôles)

Avant de commencer :

```
# installer ansible si nécessaire (ex Ubuntu)
sudo apt update
sudo apt install -y python3-pip
pip3 install --user ansible
export PATH="$HOME/.local/bin:$PATH"

# vérifier version
ansible --version
```

Fournir une VM Vagrant (optionnel)

Vagrantfile (créée une VM Ubuntu 20.04 simple, port 8080 -> 80)

```
# Vagrantfile (exemple)
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "private_network", ip: "192.168.56.101"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end
  # synced folder (optionnel)
  config.vm.synced_folder ".", "/vagrant", disabled: true
end
```

Démarrage :

```
vagrant up
# récupérer les infos SSH
vagrant ssh-config > vagrant_ssh_config
```

Vous pouvez utiliser `vagrant ssh` pour vérifier la VM.

Exercice 1 — Écrire un inventaire pour la VM Vagrant

But

Créer un fichier `inventory` utilisable par Ansible pour se connecter à la VM Vagrant.

Solution proposée (inventory - INI)

Créez `inventory` à la racine du projet :

```
[web]
web1 ansible_host=192.168.56.101 ansible_user=vagrant
ansible_ssh_private_key_file=~/.vagrant.d/insecure_private_key
ansible_python_interpreter=/usr/bin/python3
```

Remarques

- `ansible_host` = IP fournie dans le Vagrantfile.
- `ansible_ssh_private_key_file` : sur une installation Vagrant standard la clé par défaut est `~/.vagrant.d/insecure_private_key` ; sinon utilisez `vagrant ssh-config` pour récupérer `IdentityFile`.
- `ansible_python_interpreter` : garantit Python3 sur la cible.

Vérification

```
ansible -i inventory all -m ping
# ou
ansible web -i inventory -m ping
```

Attendu : `SUCCESS` pour `web1`. Si erreur SSH, ajuster chemin clé/ansible_user.

Exercice 2 — Playbook : installer Apache, MySQL, PHP

But

Écrire `site.yml` qui installe `apache2`, `mysql-server`, `php` + modules PHP pour Apache. Jouer idempotence.

Fichier **`site.yml`** (monolithique — solution)

```
---
- name: Installer LAMP sur web servers
  hosts: web
  become: true
  vars:
    mysql_root_password: "ChangeMe123!"    # en labo ; en prod utiliser
ansible-vault
    php_packages:
      - php
      - libapache2-mod-php
      - php-mysql
  tasks:
    - name: Mise à jour du cache apt
      apt:
        update_cache: yes
      when: ansible_os_family == "Debian"

    - name: Installer paquets LAMP
      apt:
        name:
          - apache2
          - mysql-server
          - "{{ php_packages }}"
        state: present
        update_cache: yes

    - name: S'assurer que mysql écoute (service started/enabled)
      service:
        name: mysql
        state: started
        enabled: yes

    - name: Définir mot de passe root MySQL (non-interactif, Debian/Ubuntu)
      debconf:
        name: "mysql-server"
```

```
        question: "mysql-server/root_password"
        value: "{{ mysql_root_password }}"
        vtype: "password"

    - name: Re-définir mot de passe root MySQL confirmation
      debconf:
        name: "mysql-server"
        question: "mysql-server/root_password_again"
        value: "{{ mysql_root_password }}"
        vtype: "password"

    - name: Create test database
      shell: "mysql -uroot -p'{{ mysql_root_password }}' -e \"CREATE DATABASE
IF NOT EXISTS lamp_demo;\""
      args:
        warn: false
      register: createdb
      changed_when: "'ERROR' not in createdb.stderr and createdb.rc == 0"

    - name: Copy index.html simple
      copy:
        src: files/index.html
        dest: /var/www/html/index.html
        owner: www-data
        group: www-data
        mode: '0644'
      notify: restart apache

    - name: Ensure apache is started and enabled
      service:
        name: apache2
        state: started
        enabled: yes

  handlers:
    - name: restart apache
      service:
        name: apache2
        state: restarted
```

Fichier `files/index.html` (exemple)

```
<!doctype html>
<html>
  <head><meta charset="utf-8"><title>LAMP Demo</title></head>
  <body>
    <h1>LAMP deployé par Ansible</h1>
    <p>Host : {{ inventory_hostname }}</p>
  </body>
</html>
```

(Copier `files/index.html` localement ; Ansible `copy` enverra le fichier sur la VM.)

Important — ordre des étapes MySQL

- Pour Debian/Ubuntu, l'installation de `mysql-server` peut demander un mot de passe. On utilise `debconf` pour préconfigurer le password (non interactif).
- Un *gotcha* fréquent : `mysql-server` sur Ubuntu 18+/MySQL 5.7+ utilise parfois l'auth socket et le mot de passe root n'est pas actif ; la commande `mysql -uroot -p` peut échouer. En labo simple on échappe ces particularités ou on crée un utilisateur SQL séparé.

Vérification (après exécution)

1. Lancer en dry-run :

```
ansible-playbook -i inventory site.yml --syntax-check
ansible-playbook -i inventory site.yml --check --diff
```

2. Lancer réel :

```
ansible-playbook -i inventory site.yml
```

3. Vérifier :

```
# depuis le control node
curl http://192.168.56.101      # devrait renvoyer votre index.html
# ou via SSH dans la VM
vagrant ssh
curl http://localhost
```

4. Vérifier mysql :

```
# en SSH dans la VM
mysql -uroot -p'ChangeMe123!' -e "SHOW DATABASES;"
```

Exercice 3 — Handler : redémarrer Apache après changement de config

But

S'assurer qu'un changement de fichier de config provoque un *notify* vers un handler `restart apache`.

Exemple (déjà présent dans `site.yml`)

La tâche `copy` notifie `restart apache` :

```
- name: Copy index.html simple
  copy:
```

```
    src: files/index.html
    dest: /var/www/html/index.html
  notify: restart apache
```

Le handler :

```
handlers:
  - name: restart apache
    service:
      name: apache2
      state: restarted
```

Test du handler

1. Mettre à jour `files/index.html` (changez le texte).
2. Relancer le playbook :

```
ansible-playbook -i inventory site.yml
```

3. Dans les logs on doit voir la tâche marked `changed` puis le handler `restart apache` exécuté.
4. Vérifier la page : `curl http://192.168.56.101` doit refléter le nouveau contenu.

Exercice 4 — Transformer le playbook en un rôle réutilisable

But

Extraire la logique LAMP dans un rôle `lamp` et appeler ce rôle depuis `site.yml`.

Arborescence rôle **`roles/lamp/`**

```
roles/lamp/
├─ defaults/
│  └─ main.yml           # valeurs par défaut (mot de passe, packages)
├─ tasks/
│  ├─ main.yml
│  └─ mysql.yml
├─ handlers/
│  └─ main.yml
├─ files/
│  └─ index.html
└─ templates/
   └─ 000-default.conf.j2   # optionnel : template vhost apache
```

Contenu (exemples)

roles/lamp/defaults/main.yml

```
---
mysql_root_password: "ChangeMe123!"
apache_pkg: apache2
mysql_pkg: mysql-server
php_packages:
  - php
  - libapache2-mod-php
  - php-mysql
doc_root: /var/www/html
index_file: index.html
```

roles/lamp/tasks/main.yml

```
---
- name: Mise à jour cache apt
  apt:
    update_cache: yes
  when: ansible_os_family == "Debian"

- name: Installer paquets LAMP
  apt:
    name:
      - "{{ apache_pkg }}"
      - "{{ mysql_pkg }}"
      - "{{ php_packages }}"
    state: present
    update_cache: yes

- name: Ensure mysql service
  service:
    name: mysql
    state: started
    enabled: yes

- name: Set mysql root password (debconf)
  debconf:
    name: "mysql-server"
    question: "mysql-server/root_password"
    value: "{{ mysql_root_password }}"
    vtype: "password"

- name: Set mysql root password again
  debconf:
    name: "mysql-server"
    question: "mysql-server/root_password_again"
    value: "{{ mysql_root_password }}"
    vtype: "password"

- name: Create DB lamp_demo
  shell: "mysql -uroot -p'{{ mysql_root_password }}' -e \"CREATE DATABASE IF
NOT EXISTS lamp_demo;\""
  args:
    warn: false
  register: _createdb
  changed_when: "'ERROR' not in _createdb.stderr and _createdb.rc == 0"
```

```
- name: Deploy index file
  copy:
    src: "{{ index_file }}"
    dest: "{{ doc_root }}/index.html"
    owner: www-data
    group: www-data
    mode: '0644'
  notify: Restart apache

- name: Ensure apache started
  service:
    name: "{{ apache_pkg }}"
    state: started
    enabled: yes
```

roles/lamp/handlers/main.yml

```
---
- name: Restart apache
  service:
    name: "{{ apache_pkg }}"
    state: restarted
```

roles/lamp/files/index.html (copier l'index de earlier)

Appel du rôle depuis `site.yml`

Modifiez `site.yml` :

```
---
- name: Déployer LAMP via rôle
  hosts: web
  become: true
  roles:
    - role: lamp
```

Vérification et idempotence

1. `ansible-playbook -i inventory site.yml --check --diff`
2. `ansible-playbook -i inventory site.yml`
3. Relancer : `ansible-playbook -i inventory site.yml` → la plupart des tâches doivent afficher `ok`, attestant l'idempotence.

Tests & vérifications

- Page web : `curl http://192.168.56.101` (ou `vagrant ssh` + `curl localhost`) doit renvoyer votre index.
- Service Apache : `systemctl status apache2` dans VM.
- MySQL : `mysql -uroot -p'ChangeMe123!' -e "SHOW DATABASES;"` dans VM.
- Idempotence : 2ème exécution du playbook → pas de changements significatifs.

Bonnes pratiques et améliorations

- **Sécuriser mot de passe** : ne PAS garder `mysql_root_password` en clair ; utilisez `ansible-vault` :
- `ansible-vault create group_vars/web/vault.yml`
- `# puis ajouter vars_files: - group_vars/web/vault.yml`
- **Utiliser modules MySQL** : dans un contexte réel, utilisez `community.mysql.mysql_user` et `community.mysql.mysql_db` (nécessite collection `community.mysql`).
- **Handlers et templates** : produire un `000-default.conf.j2` pour config Apache via template et notifier handler si change.
- **Testing** : ajouter `molecule` pour tester le role automatiquement (voir annexe précédente).
- **Multi-OS** : implémenter conditionnels pour RedHat vs Debian (yum vs apt, noms de services).

Debug & erreurs courantes

- **SSH auth fails** : vérifier `ansible_user`, `ansible_ssh_private_key_file` et exécuter `ssh -i <key> vagrant@192.168.56.101`.
- **MySQL password / socket auth** : moderne MySQL/Ubuntu peut utiliser `auth_socket`; la gestion via `debconf` peut ne pas suffire. Si souci, exécutez manuellement dans VM pour diagnostiquer.
- **Service names differ** : Debian=`apache2`, RHEL=`httpd`. Conditionnez via variable `apache_pkg`.
- **Volume miroir écrase fichiers** : si vous montez votre répertoire local dans `/var/www/html`, la copie du rôle sera écrasée ; attention.

Annexes utiles

```
# tests rapides
ansible -i inventory all -m ping
ansible-playbook -i inventory site.yml --syntax-check
ansible-playbook -i inventory site.yml --check --diff

# exécution
ansible-playbook -i inventory site.yml

# si vous utilisez vault
ansible-playbook -i inventory site.yml --ask-vault-pass

# debug logs + verbosité
ansible-playbook -i inventory site.yml -vvv
```