TP R320 : Introduction à la Virtualisation avec Vagrant

Auteur: Guillaume Urvoy-Keller / Amaury BOLLER
Date: 13 janvier 2025
Révision 3 Décembre 2025
Consigne: Vous devez rédiger un compte-rendu à envoyer à l'enseignant(e) en fin de séance.

---

1. VirtualBox : Autopsie d'une VM

L'objectif de cette partie est de comprendre ce qu'est une machine virtuelle (VM) au niveau de l'hyperviseur (VirtualBox) avant d'utiliser un outil d'automatisation comme Vagrant.

1. **Créez** une nouvelle machine virtuelle depuis l'interface graphique de VirtualBox :
    - Nom : **Ubuntu-Test**
    - Type : **Linux**
    - Version : **Ubuntu (64-bit)**
    - Laissez toutes les autres valeurs par défaut (mémoire, disque, etc.).
2. **Démarrez** la VM. (Elle va probablement échouer au boot car il n'y a pas d'OS installé, c'est normal).
3. **Q1.3 – Processus VirtualBox.** Listez les processus actifs de VirtualBox sur votre machine hôte et identifiez leur rôle.
    - *Astuce :* Sous Windows, utilisez le Gestionnaire des tâches ou `Get-Process` en PowerShell. Sous Linux/macOS, utilisez `ps aux | grep VirtualBox`.
    - Aidez-vous de la documentation officielle (Chapitre 10) : VirtualBox Technical Background.
4. **Q1.4 – Inventaire des fichiers.** Localisez le répertoire où VirtualBox stocke les fichiers de la VM que vous venez de créer.
    - **Analysez** chaque fichier présent dans ce dossier :
        - Quel est son extension ?
        - Est-ce un fichier texte (ASCII/XML) ou binaire ?
        - Quel est son rôle précis ?
5. **Q1.5 – Analyse du disque virtuel.**

- Quelle est la taille du fichier représentant le disque dur sur votre machine hôte ?
- Comparez cette taille avec la taille "déclarée" lors de la création de la VM (souvent 10 Go par défaut). Pourquoi y a-t-il une différence ? (Cherchez la notion de *provisionnement dynamique*).

6. **Q1.6 – Commande VBoxManage.** Exécutez `VBoxManage list vms` et copiez l'extrait de sortie correspondant à la VM nouvellement créée. Quel est l'intérêt de cette commande pour les administrateurs ?

7. **Q1.7 – Fichiers de configuration.** Quel fichier texte (extension `.vbox`) décrit l'ensemble de la configuration matérielle de la VM ? Montrez (capture ou extrait) qu'il s'agit bien de XML lisible.

---

2. Vagrant : Premiers pas

Vagrant est un outil permettant de créer et configurer des environnements de développement virtuels reproductibles et portables.

2.1 Création d'une première VM Ubuntu

1. **Créez** une arborescence de travail propre :

```
mkdir -p ~/Vagrant/VM1
cd ~/Vagrant/VM1
```

2. **Initialisez** votre environnement Vagrant avec une image Ubuntu récente (Ubuntu 22.04 LTS "Jammy Jellyfish") :

```
vagrant init ubuntu/jammy64
```

   - Cette commande crée un fichier `Vagrantfile` dans le répertoire courant.

3. **Démarrez** la VM :

```
vagrant up
```

   - Observez les logs qui défilent. Vagrant télécharge l'image (si elle n'est pas déjà présente), l'importe dans VirtualBox, et configure le réseau.

4. **Connectez-vous** à la VM en SSH :

```
vagrant ssh
```

- Vous êtes maintenant dans la VM.

5. **Q2.5 – Interfaces réseau.** Analysez la configuration réseau de la VM :
   - Quelles sont les interfaces réseaux actives ? (Utilisez `ip a` ou `ifconfig`).
   - Quelle est l'adresse IP de l'interface `eth0` (ou `enp0s3`) ? À quoi sert cette interface (NAT) ?
6. **Q2.6 – Fichiers VirtualBox vs Vagrant.** Retournez sur votre machine physique (tapez `exit`).
   - Allez dans le répertoire de VirtualBox où cette nouvelle VM est stockée.
   - Comparez les fichiers avec ceux de la partie 1. Le format du disque est-il le même (VDI vs VMDK) ?
   - Consultez la documentation pour comprendre les différences : VirtualBox Disk Image Files.
7. **Q2.7 – Versions logicielles.** Quelle est la version de Vagrant (`vagrant --version`) et celle de VirtualBox (`VBoxManage --version`) installées sur votre poste ?
8. **Q2.8 – Arrêt propre.** Quelle commande utilisez-vous pour arrêter proprement la VM (sans la détruire) ? Quels sont les messages affichés par Vagrant/VirtualBox lors de cet arrêt ?

---

3. Personnalisation de la VM et Provisioning

Le fichier `Vagrantfile` est écrit en Ruby. Il décrit la configuration de votre machine.

3.1 Provisioning Shell (Partie 1)

Le "provisioning" permet d'installer et configurer des logiciels automatiquement au démarrage de la VM.

1. **Q3.1 – Lecture sélective du Vagrantfile. Éditez** le fichier `Vagrantfile` dans `~/Vagrant/VM1` .
   - Vagrant génère beaucoup de commentaires. Pour voir uniquement les lignes actives, vous pouvez utiliser :

     ```
     grep -vE "^\s*#" Vagrantfile | grep -vE "^\s*$"
     ```

2. **Q3.2 – Bloc de provisioning. Ajoutez** (ou décommentez et modifiez) le bloc de provisioning pour installer le serveur web Apache :

```
config.vm.provision "shell", inline: <<-SHELL
  apt-get update
  apt-get install -y apache2
SHELL
```

3. **Q3.3 – Commande `vagrant provision`.** Appliquez les changements sur la VM en cours d'exécution :

```
vagrant provision
```

4. **Q3.4 – Contrôles dans la VM. Vérifiez** que le serveur Web fonctionne (depuis l'intérieur de la VM via `vagrant ssh`) :
   - (a) Le service est-il actif ? (`systemctl status apache2`)
   - (b) Le port 80 est-il en écoute ? (`ss -ltn` ou `netstat -antp`)
   - (c) Testez le serveur localement : `curl http://localhost`
5. **Q3.5 – Blocage côté hôte. Accès depuis l'hôte :**
   - Essayez d'accéder au site web depuis le navigateur de votre machine physique. Cela échoue. Pourquoi ?
6. **Q3.6 – Lecture des options réseau. Configuration Réseau :**
   - Il existe 3 méthodes principales pour accéder aux services de la VM depuis l'hôte. Identifiez-les dans les commentaires du `Vagrantfile` :
     1. **Port Forwarding** (Redirection de port)
     2. **Private Network** (Réseau privé / Host-only)
     3. **Public Network** (Pont / Bridged)
7. **Q3.7 – Redirection de port. Mise en pratique :**
   - Configurez une redirection de port dans le `Vagrantfile` :

```
config.vm.network "forwarded_port", guest: 80, host: 8080
```

   - Appliquez la modification (nécessite un redémarrage) :

```
vagrant reload
```

   - Testez l'accès depuis votre navigateur physique : `http://localhost:8080`.

8. **Q3.8 – Comparatif des options réseau. Réflexion :** Quels sont les avantages et inconvénients de chaque mode réseau (Forwarded Port vs Private Network vs Public Network) ?

9. **Q3.9 – `provision` vs `reload --provision`.** Expliquez la différence entre `vagrant provision` (sur une VM déjà démarrée) et `vagrant reload --provision`.

3.2 Dossiers Partagés (Synced Folders)

Vagrant permet de partager des dossiers entre l'hôte et la VM, ce qui est idéal pour éditer du code sur l'hôte et l'exécuter dans la VM.

1. **Q3.10 – Préparation du contenu web.** Sur votre machine hôte, dans `~/Vagrant/VM1`, **créez** un dossier `site_content`.

2. **Q3.11 – Page d'accueil personnalisée. Créez** un fichier `index.html` dans ce dossier :

```
<h1>Bienvenue sur ma VM Vagrant !</h1>
```

3. **Q3.12 – Montage d'un dossier partagé. Modifiez** le `Vagrantfile` pour monter ce dossier dans `/var/www/html` (la racine d'Apache) :

```
# Remplacez le dossier par défaut d'Apache
config.vm.synced_folder "./site_content", "/var/www/html"
```

(Note : Assurez-vous que le provisioning shell n'écrase pas cette configuration ou ne crée pas de conflits).

4. **Q3.13 – Redémarrage avec synchro. Appliquez** les changements :

```
vagrant reload
```

5. **Q3.14 – Validation côté navigateur. Testez :** Rafraîchissez la page `http://localhost:8080`. Vous devriez voir votre nouveau fichier HTML. Modifiez le fichier sur l'hôte, rafraîchissez la page : la modification est instantanée.

6. **Q3.15 – Pourquoi le provisioning est optionnel ?** Expliquez pourquoi Vagrant n'exécute pas automatiquement le provisioning à chaque démarrage.

7. **Q3.16 – Vérification finale.** Indiquez la commande (ou l'URL) utilisée pour afficher la page `vagrantsite` exposée par Apache et joignez une capture/sortie démontrant la bonne prise en compte des changements.

3.3 Création d'une "Box" personnalisée (Packaging)

Si votre VM est parfaitement configurée, vous pouvez en faire un modèle (une "box") pour la réutiliser.

1. **Q3.17 – Préparation de la box. Préparez** la VM pour l'export.
    - Il est conseillé de nettoyer le cache apt (`apt-get clean`) avant, pour réduire la taille.
2. **Q3.18 – Commande `vagrant package` . Créez le package** :

    ```
    vagrant package
    ```

    - Cela crée un fichier `package.box` .
3. **Q3.19 – Ajout de la box à la bibliothèque. Ajoutez** cette box à votre liste locale :

    ```
    vagrant box add --name ma-box-web package.box
    ```

4. **Q3.20 – Inventaire des boxes. Vérifiez** :

    ```
    vagrant box list
    ```

    - Où sont stockées physiquement ces boxes sur votre machine ? (`~/.vagrant.d/boxes`).
5. **Q3.21 – Emplacement des boxes.** Listez le contenu de `~/.vagrant.d/boxes` et identifiez les images installées.
6. **Q3.22 – Nettoyage du package.** Est-ce que l'on peut effacer `/Vagrant/VM1/package.box` (avec une commande `rm`) sans impacter la box ajoutée dans Vagrant ? Justifiez.

---

4. Vagrant Avancé

4.1 Vagrant Cloud

HashiCorp (l'éditeur de Vagrant) propose un catalogue d'images : Vagrant Cloud.

1. **Q4.1 – Rechercher des images fiables.** Recherchez les images officielles pour `debian/bookworm64` (Debian 12) et `almalinux/9` (Clone RHEL).
2. **Q4.2 – Critères de confiance.** Comment savoir si une image est "de confiance" ? (Regardez le nombre de téléchargements, l'auteur, la date de mise à jour).

4.2 Commandes utiles et Snapshots

1. **Q4.3 – État global.**

   ```
   vagrant global-status
   ```

   - Cette commande liste toutes les VMs Vagrant actives sur votre machine, quel que soit le dossier.

2. **Q4.4 – Comparer les snapshots.**

   - Avant de faire une manipulation risquée, créez un instantané :

     ```
     vagrant snapshot save backup_avant_crash
     ```

   - Pour restaurer :

     ```
     vagrant snapshot restore backup_avant_crash
     ```

   - Quelle est la différence entre un snapshot Vagrant et un snapshot VirtualBox classique ?

3. **Q4.5 – Suspension vs reprise.**

   - Pour mettre en pause la VM sans l'éteindre (sauvegarde de la RAM sur disque) :

     ```
     vagrant suspend
     ```

   - Pour reprendre : `vagrant resume`.

4.3 Vagrantfile Avancé (Exemple commenté)

Q4.6 – Analyse d'un Vagrantfile avancé. Analysez ce `Vagrantfile` plus complexe. Quelles fonctionnalités utilise-t-il ?

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.define "web-server" do |web|
    web.vm.box = "ubuntu/jammy64"

    # Réseau : Port forwarding
    web.vm.network "forwarded_port", guest: 80, host: 8888
```

```
    # Dossier partagé avec type spécifique (rsync)
    web.vm.synced_folder "./data", "/opt/data", type: "rsync",
      rsync__args: ["--verbose", "--archive", "--delete", "-z"]

    # Provisioning
    web.vm.provision "shell", inline: <<-SHELL
      apt-get update
      apt-get install -y nginx
    SHELL

    # Configuration spécifique au provider VirtualBox
    web.vm.provider "virtualbox" do |v|
      v.name = "MonServeurWebAvance"
      v.memory = 2048
      v.cpus = 2
      v.gui = false # Mode headless (sans écran)
    end
  end
end
```

---

5. Vagrant Multi-machines

Vagrant permet de définir plusieurs VMs dans un seul `Vagrantfile`. C'est très utile pour simuler un réseau (ex: un serveur web et un serveur de base de données).

1. Créez un nouveau dossier `~/Vagrant/Multi` et créez le `Vagrantfile` suivant :

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  # Configuration commune
  config.vm.box = "ubuntu/jammy64"

  # VM 1 : Base de données
  config.vm.define "db" do |db|
    db.vm.hostname = "db01"
    db.vm.network "private_network", ip: "192.168.56.10"
  end

  # VM 2 : Serveur Web
```

```
  config.vm.define "web" do |web|
    web.vm.hostname = "web01"
    web.vm.network "private_network", ip: "192.168.56.11"

    # Provisioning : on ajoute l'IP de la DB dans le fichier hosts du Web
    web.vm.provision "shell", inline: <<-SHELL
      echo "192.168.56.10 db01" >> /etc/hosts
    SHELL
  end
end
```

2. **Démarrez** l'environnement :

```
vagrant up
```

3. **Q5.3 – Tests de connectivité.**
   - Connectez-vous au serveur web : `vagrant ssh web`
   - Pinguez la base de données : `ping 192.168.56.10` ou `ping db01` .
   - Est-ce que cela fonctionne ? Pourquoi ?
4. **Q5.4 – Boucles Ruby (exemple à analyser).**
   - Étudiez l'extrait suivant :

```
(1..5).each do |i|
  host_id = format("%02d", i)
  config.vm.define "web#{host_id}" do |node|
    node.vm.box = "ubuntu/jammy64"
    node.vm.hostname = "web#{host_id}"
    node.vm.network "private_network", ip: "192.168.56.2#{i}"
  end
end
```

   - Expliquez précisément ce que cette boucle produit (noms/hôtes créés, adresses attribuées, rôle de `format`).
   - Indiquez pourquoi ce type de construction est utile dans un `Vagrantfile` multi-machines.

---

6. Pour aller plus loin (Hors TP)

- **Providers :** Vagrant peut piloter autre chose que VirtualBox (VMware, Hyper-V, Docker, AWS...).
- **Provisioners :** Au lieu de scripts Shell, on utilise souvent **Ansible**, **Puppet** ou **Chef** pour des configurations complexes et maintenables.
- **Packer :** Outil complémentaire pour créer vos propres "boxes" de manière industrielle.