Fiche de révision: Virtualisation

> La virtualisation est une abstraction des ressources informatiques permettant d’exécuter plusieurs OS sur un même matériel.

1. Principes de la virtualisation

- **Partage de ressources**: Les ressources matérielles sont partagées entre les différentes machines virtuelles.
- **Isolation**: Chaque machine virtuelle est isolée des autres.
- **Indépendance**: Chaque machine virtuelle est indépendante des autres.
- **Portabilité**: Les machines virtuelles peuvent être déplacées d'une machine physique à une autre.
- **Sécurité**: Les machines virtuelles sont isolées les unes des autres.
- **Transparence**: Les machines virtuelles ne savent pas qu'elles sont virtualisées.

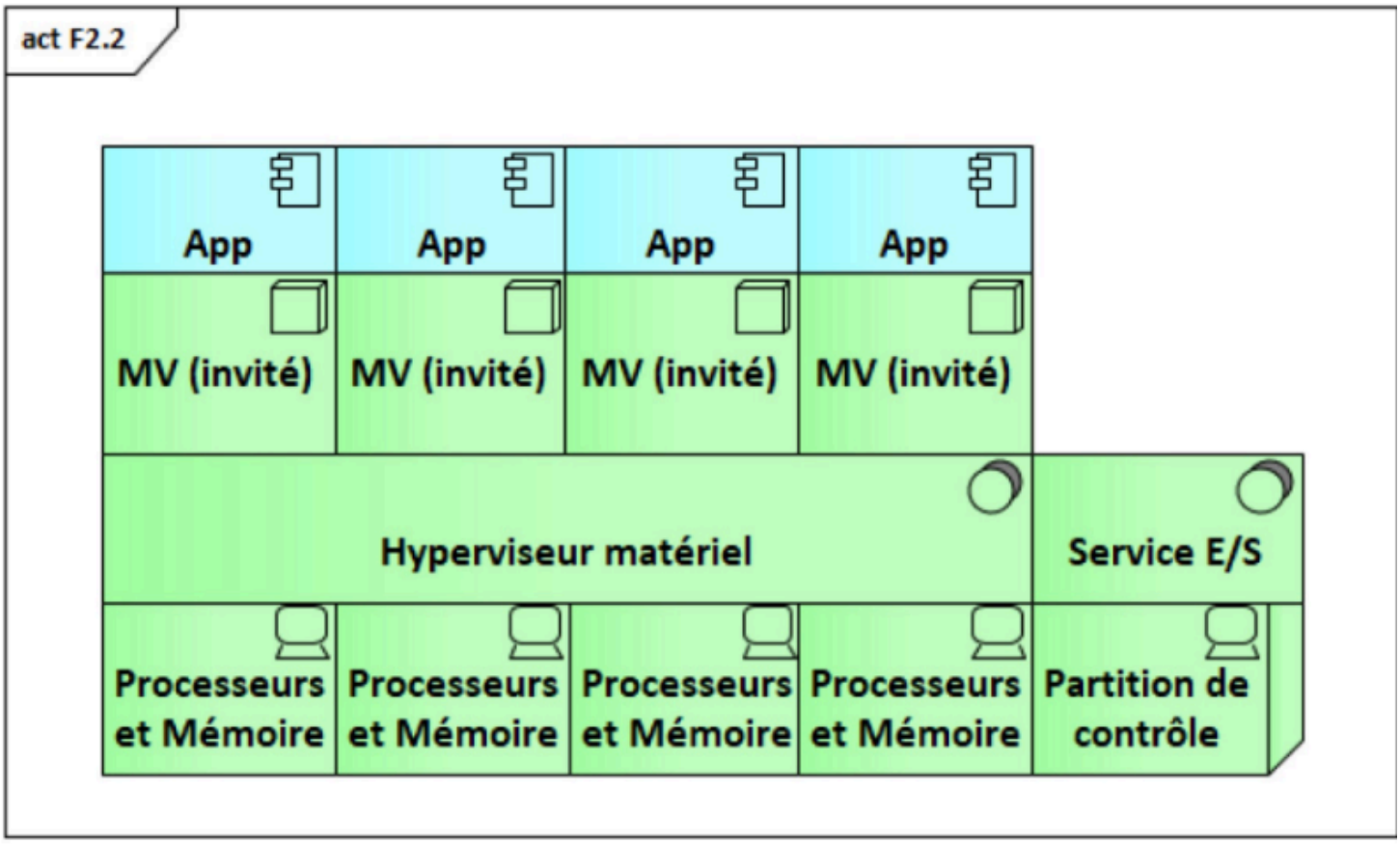
Machines virtuelles

- Environnement logiciel qui émule un système informatique.
- Ensemble de matériels virtuels (processeur, mémoire, disque, réseau...).
- Représentée par un fichier de config.

2. Les hyperviseurs

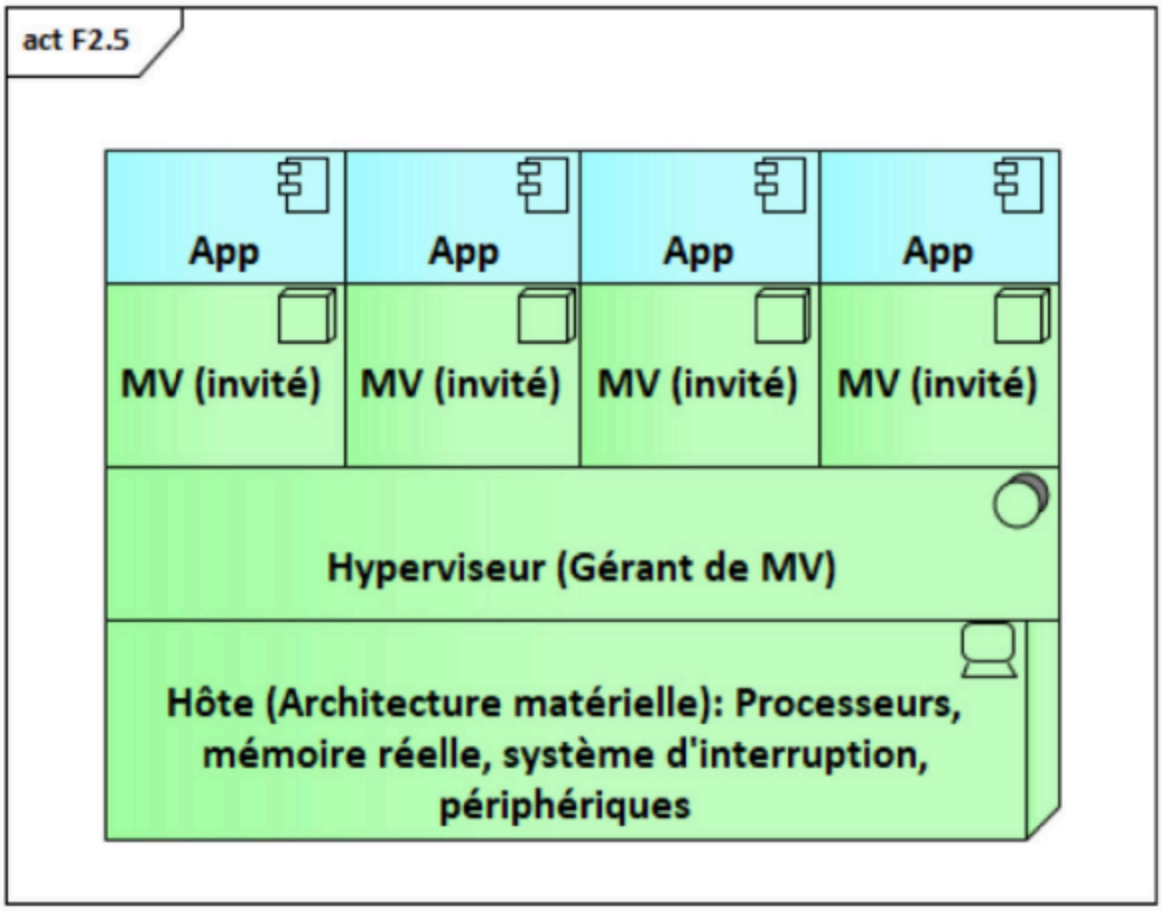
- Un hyperviseur est un logiciel qui permet de créer et de gérer des machines virtuelles.

2.1. Hyperviseurs de type 0



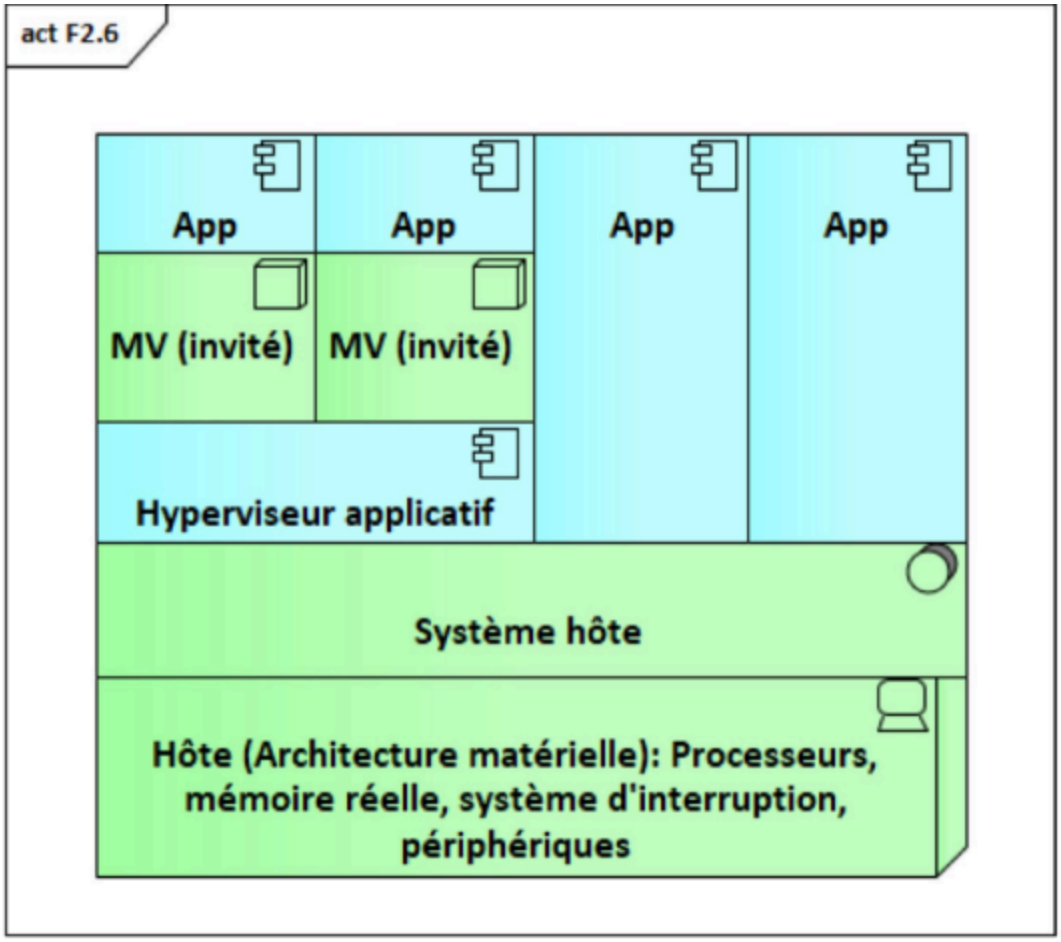
- Fournit un support pour générer des machines virtuelles à travers un firmware.
- S'exécute directement sur le matériel.
- Exemple: IBM LPAR, Oracle LDoms...

2.2. Hyperviseurs de type 1



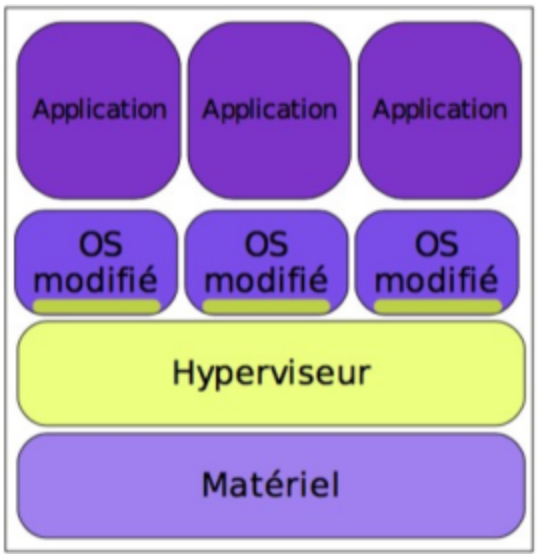
- Logiciel qui s'exécute directement sur le matériel pour fournir la virtualisation.
- S'exécute directement sur le matériel.
- Exemple: VMware ESXi, Nutanix AHV, Citrix XenServer...

2.3. Hyperviseurs de type 2



- Logiciel qui s'exécute sur un système d'exploitation hôte.
- Exemple: VirtualBox, VMware Workstation, QEMU...

2.4. Paravirtualisation

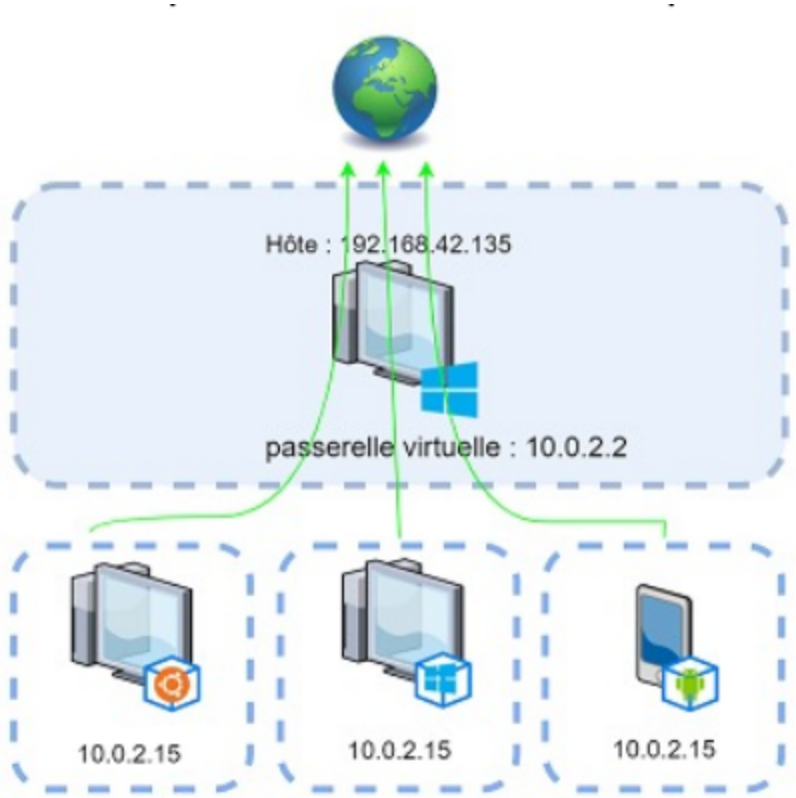


- Technique de virtualisation qui modifie le système d'exploitation invité pour qu'il soit conscient de la virtualisation.
- Permet d'obtenir de meilleures performances.
- Exemple: Xen, KVM...

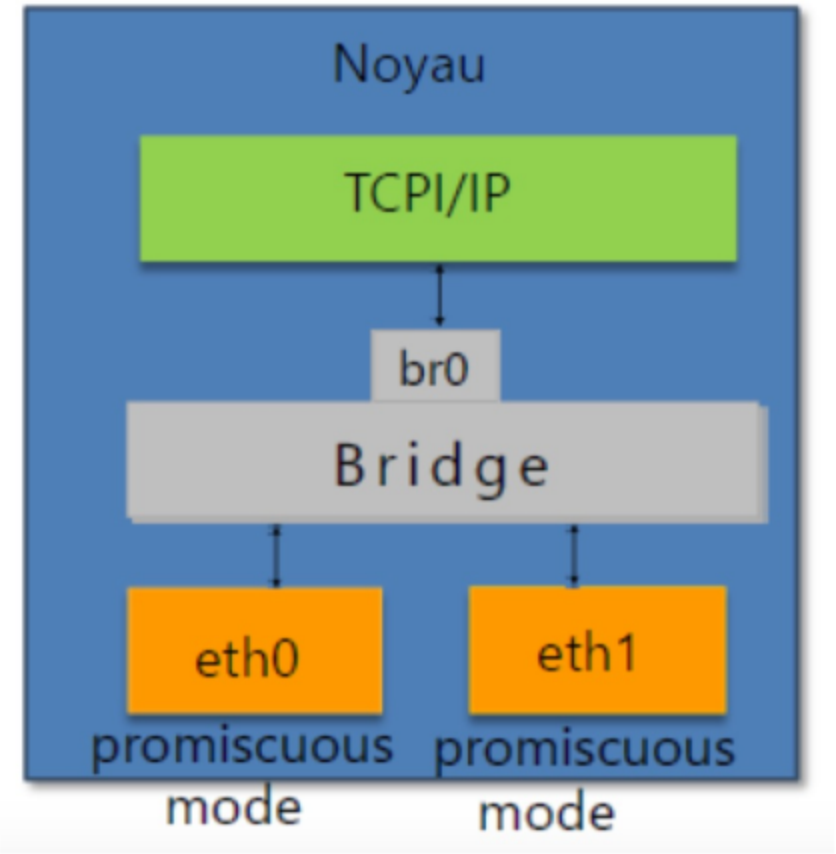
3. Virtualisation du réseau

3.1. Modes de connexion des VM :

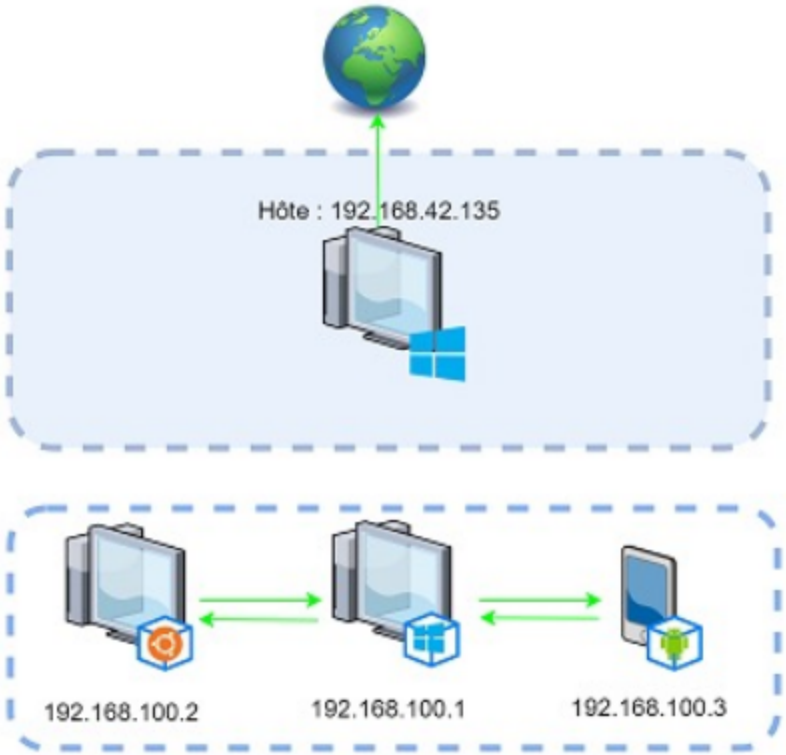
- **NAT** : Accès à Internet, mais pas de communication entre VM.



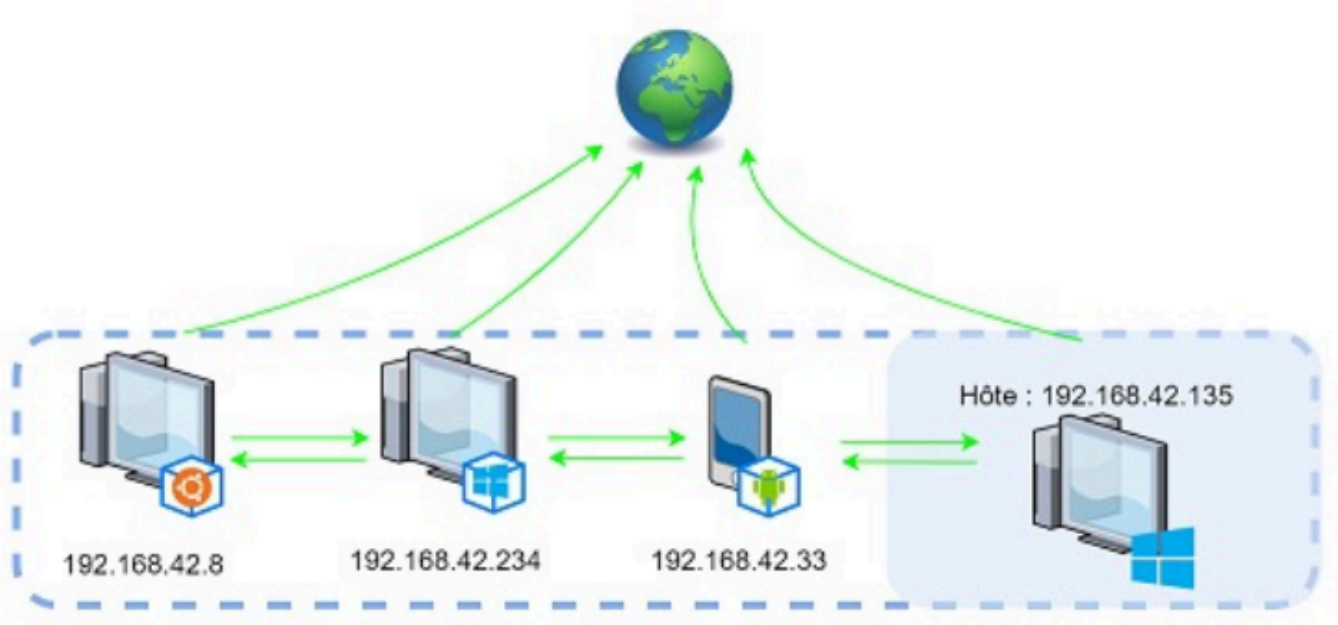
- **Bridging** : Permet aux VM d’être visibles sur le réseau.



- **Réseau interne** : Communication entre VM sans accès à Internet.



- **Accès par pont** : Communication entre VM et accès à Internet.



3.2. VLAN (Virtual Local Area Network) :

> Sépare logiquement les réseaux tout en partageant la même infrastructure physique.
> Permet de segmenter un réseau en plusieurs sous-réseaux virtuels.
> Plus de sécurité et de performances.

- Basé sur **les ports** : Affectation selon les ports du switch.
- Basé sur les **adresses MAC** : Identification par adresse MAC.
- Basé sur l'**adresse IP** : Regroupement en sous-réseaux.

3. Les conteneurs

- Perception d'environnements isolés et indépendants
- Partage du noyau de l'hôte, réduit les ressources nécessaires
- Pas de virtualisation complète de l'OS

3.1. Principes

- **Encapsulation**: Contient tout ce qui est nécessaire pour l'exécution de l'application.
- **Images**: Modèle de base pour la création de conteneurs.
- **Isolation**: Chaque conteneur est isolé des autres.
- **Portabilité**: Les conteneurs peuvent être déplacés d'une machine à une autre.
- **Efficacité**: Légers et rapides à démarrer.
- **Scalabilité**: Facile à dupliquer.

3.2. Docker

- Empaqueter une application et ses dépendances dans une image Docker.
- Utilisation de volumes pour la persistance des données.
- Gestion des conteneurs avec Docker Engine.

3.3. Comparaison VM/Conteneur

-	VM	Conteneur
Avantages	Isolation complète Sécurisé Adapté pour avoir des OS distincts	Léger et rapide Facile à déployer et migrer Idéal pour le dev. et les microservices.
Inconvénients	Déploiement lourd Gourmand en ressources	Moins sécurisé Dépendant du noyau de l'hôte Isolation moins stricte

4. Tendances futures

- **Hybridation**: Combinaison de virtualisation et de conteneurs pour combiner les avantages des deux.
- **Kubernetes**: Orchestration de conteneurs pour automatiser le déploiement, la mise à l'échelle et la gestion des applications.

5. Vagrant

5.1. Présentation

- Outil en ligne de commande pour la gestion des machines virtuelles.
- Utilise des **Vagrantfile** pour configurer les machines virtuelles.
- Fonctionne avec VirtualBox, VMware...

5.2. Commandes

- `vagrant init` : Initialise un Vagrantfile.
- `vagrant up` : Démarre la machine virtuelle.
- `vagrant ssh <nom>` : Se connecte à la machine virtuelle.

- `vagrant halt <nom>` : Arrête la machine virtuelle.
- `vagrant destroy <nom>` : Supprime la machine virtuelle.

5.3. Vagrantfile

- **Provisionnement**: Installation de logiciels sur la machine virtuelle.
- **Réseau**: Configuration du réseau de la machine virtuelle.
- **Allocation des ressources**: Mémoire, CPU...
- **Définition des machines virtuelles**.
- **Partage de fichiers**.

```
# Déclaration de la version de l'API Vagrant utilisée
Vagrant.configure("2") do |config|

  # Configuration de la première machine virtuelle : serveur web
  config.vm.define "web" do |web|
    # Utilisation de l'image Ubuntu 20.04
    web.vm.box = "ubuntu/focal64"

    # Dossier partagé entre l'hôte et la machine virtuelle
    web.vm.synced_folder ".", "/vagrant" # Local, distant

    # Configuration du réseau privé pour la machine web
    web.vm.network "private_network", ip: "192.168.56.101"

    # Allocation des ressources
    web.vm.provider "virtualbox" do |vb|
      vb.gui = true         # Interface graphique
      vb.name = "web"       # Nom de la machine virtuelle
      vb.memory = "1024"  # 1 GB de mémoire
      vb.cpus = 2          # 2 CPU
    end

    # Provisionnement : installation d'Apache
    web.vm.provision "shell", inline: <<-SHELL
      sudo apt-get update
      sudo apt-get install -y apache2
      sudo systemctl enable apache2
      sudo systemctl start apache2
    SHELL
  end

  # Configuration de la deuxième machine virtuelle : serveur de base de données
  config.vm.define "db" do |db|
    # Utilisation de la même image Ubuntu 20.04
    db.vm.box = "ubuntu/focal64"

    # Configuration du réseau privé pour la machine db
    db.vm.network "private_network", ip: "192.168.56.102"

    # Allocation des ressources
    db.vm.provider "virtualbox" do |vb|
      vb.memory = "2048"  # 2 GB de mémoire
      vb.cpus = 2          # 2 CPU
```

```
    end

    # Provisionnement : installation de MySQL
    db.vm.provision "shell", inline: <<-SHELL
      sudo apt-get update
      sudo apt-get install -y mysql-server
      sudo systemctl enable mysql
      sudo systemctl start mysql
    SHELL
  end
end
```

6. Docker

6.1. Présentation

- Plateforme de conteneurisation permettant de créer, déployer et exécuter des applications dans des conteneurs.

6.2. Commandes

- `docker run <image>` : Crée un conteneur à partir d'une image.
- `docker pull <image>` : Télécharge une image.
- `docker ps` : Liste les conteneurs en cours d'exécution.
- `docker images` : Liste les images disponibles.
- `docker exec -it <conteneur> <commande>` : Exécute une commande dans un conteneur.
- `docker stop <conteneur>` : Arrête un conteneur.
- `docker rm <conteneur>` : Supprime un conteneur.
- `docker rmi <image>` : Supprime une image.
- `docker build -t <nom> .` : Construit une image à partir d'un Dockerfile.