SETUP VAGRANT

I. Prérequis

II. Installations

III. Troubleshoot

IV. Commandes vagrant

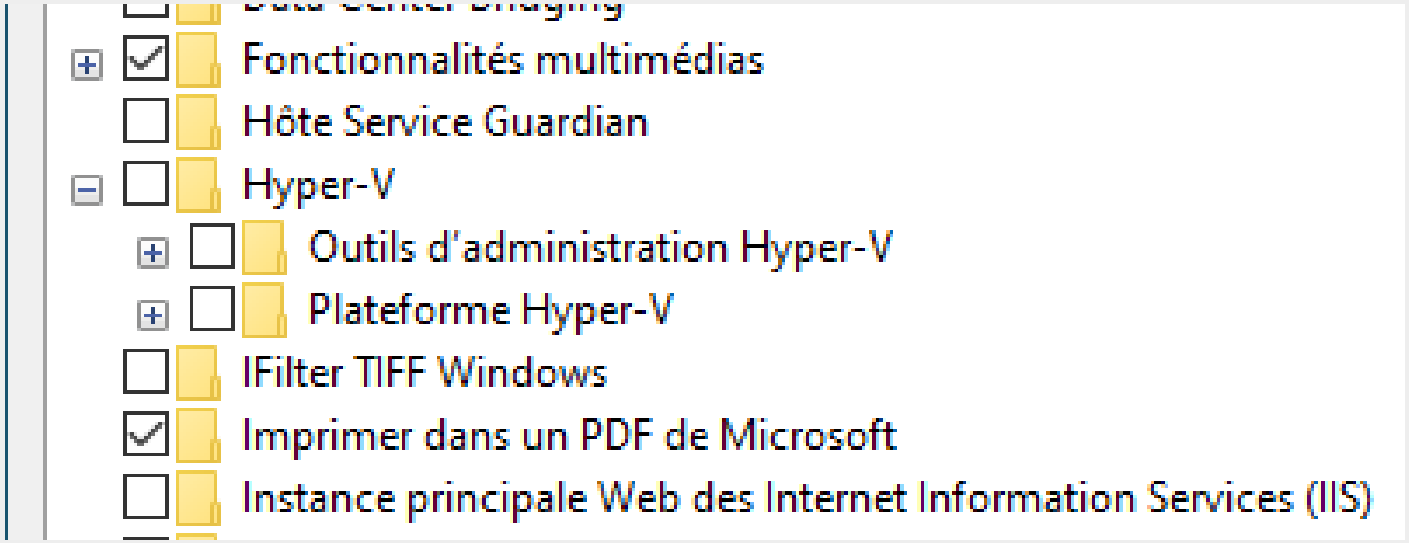
V. Troubleshoot Vagrant

I. Prérequis

- Une machine avec ~ 16Go de RAM
- Virtualbox >= 7.0 installé
- Vagrant récent installé (2.4.5)

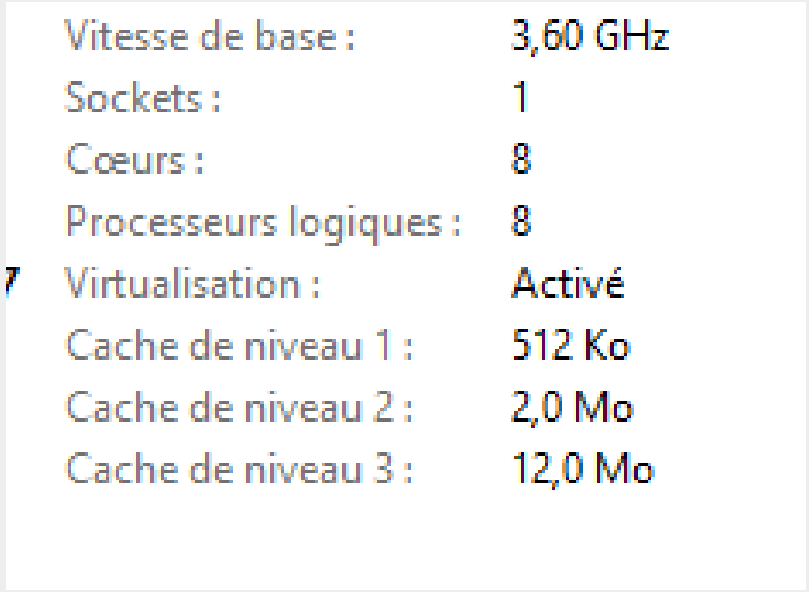
désactiver hyper-v sous Windows

- moteur de recherche de windows > « activer / désactiver les fonctionnalités Windows »



Virtualisation hardware activée

- Windows: « gestionnaire de tâches » (Ctrl + Suppr. + Alt)
  - onglet « performances »



- Linux 64 bit: observer les flags **vmx** ou **svm** dans `cat /proc/cpuinfo`

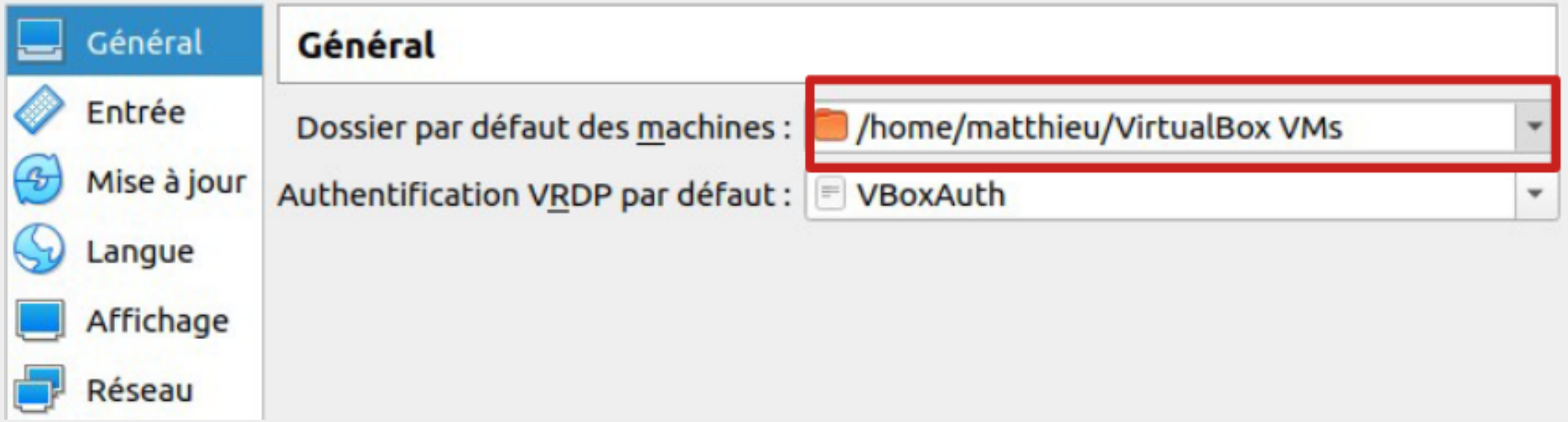
II. Installations

- Virtualbox: **ici**
- Vagrant: **ici** - Windows | Linux | MacOs - *arch AMD64*
  - paquets deb / rpm: **ici**
  - installation classique + *redémarrage (Windows)*
  - dans un terminal: `vagrant -v`
- désinstallation propre
  - Linux | MacOs: via le gestionnaire de paquet
  - windows: «Ajouter ou supprimer des programmes»

III. Troubleshoot VirtualBox

Dossier des VMS

- Windows / Linux / Mac OS: par défaut `~\VirtualBox VMs`



“ *on peut changer ce dossier si le disque courant est rempli !!* ”

IV. Lancer la VM

Principe

- Vagrant permet de
  - lancer automatiquement des **Machines Virtuelles**
  - sur des **Hyperviseurs** (ici virtualbox)
  - à partir d'**images** nommées *box vagrant*
  - et via un **script** nommé *Vagrantfile*

télécharger une box vagrant

- télécharger *en ligne de commande* une box depuis **https://portal.cloud.hashicorp.com/vagrant/discover**

```
## DANS UN TERMINAL
# version actuelle
vagrant box add <box_name>
# autre version
vagrant box add <box_name> --box-version=<version>

## après le téléchargement (attendre patiemment la fin de la commande)
# voir les boxes disponibles en local
vagrant box list
```

Fichier Vagrantfile

- script en **Ruby** pilotant création et provisionnement de la VM

```
Vagrant.configure(2) do |config|
  subject  = "xxxxxx"
  hostname = "#{subject}.lan"
  image    = "ml-registry/#{subject}"
  memory   = 8192
  cpus     = 4
```

- 3 configurations réseau usuelles pour connecter la VM
  - **NAT:** utiliser la machine hôte + redirecton de ports *hôte <-> VM*
  - **Bridge Public**: (cf infra)
  - **Bridge Privé**: (cf infra)

configuration réseau NAT

- dans un environnement restrictif/contrôlé sur les **adresses ip**
- *entreprises / universités / résidence uni.*
- décommenter la config **NAT** et commenter la config **Bridge**

```
machine.vm.provider "virtualbox" do |v|
  v.memory = "#{memory}"
  v.cpus = "#{cpus}"
  v.name = "#{hostname}"
  ...
  ## configuration réseau NAT
  v.customize ["modifyvm", :id, "--natpf1", "https,tcp,127.0.0.1,8443,,443"]
  v.customize ["modifyvm", :id, "--natpf1", "yyyy,tcp,127.0.0.1,zzzz,,ttt"]
```

configuration réseau Bridge Public

- dans un environnement souple sur les **adresses ip disponibles**
- décommenter la config **Bridge** et commenter la config **NAT**

```
interface = "interface Ethernet|wifi: ipconfig /all (Windows) | ip a (Linux)"
ip        = "ip disponible pour la VM sur l'interface: ping | nslookup"
cidr      = "CIDR: masque de sous réseau 24 <==> 255.255.255"
...
  # configuration réseau bridge
  machine.vm.network "public_network",
    bridge: "#{interface}",
    ip: "#{ip}",
    netmask: "#{cidr}"
```

configuration réseau Bridge Privé

- pour certaines formations
- pas d'accès directs aux ports (navigateurs)

```
...
  # configuration réseau bridge privé
  machine.vm.network "private_network",
    ip: "192.168.50.4",
    virtualbox__intnet: "my_private_network"
```

DNS local

- si les *droits admin* sont accessibles
- éditer le fichier **hosts** de la machine hôte avec les *droits admin !!!*
- associer l'adresse IP de cette interface au nom d'hôte de la VM

```
## LINUX / MAC OS => /etc/hosts
## WINDOWS        => c:\Windows\System32\drivers\etc\hosts
## -----IP----- ---Domain---
## ajouter une nouvelle ligne (Bridge)
xxx.yyy.zzz.ttt <hostname>
## OU ajouter un nouveau hostname sur une ligne existante (NAT)
127.0.0.1 localhost ... <hostname>
```

Lancer !

```
## DANS UN TERMINAL, DANS LE DOSSIER CONTENANT LE FICHIER VAGRANTFILE

vagrant up

# se connecte à la VM en ssh
vagrant ssh
# voir la configuration cliente sous-jacente
vagrant ssh-config

## EN CAS DE VMS MULTIPLES, vagrant up lance TOUT !
# lancer une machine avec son nom "v.name"
vagrant up <vm_name>
vagrant ssh <vm_name>
```

V. Troubleshoot Vagrant

dossier des boxes

- `~/.vagrant.d/boxes/`
- vérifier le nom et la taille de l'image

changer de version

1. `vagrant box remove <box_name> [ --box-version=<version> ]`
2. `vagrant box add <box_name> [ --box-version=<version> ]`

« Rip it up & Start again »

en cas de problème de lancement

1. si vm lancée mais non connectée, *arrêter* `vagrant halt`
2. *détruire complètement* la vm `vagrant destroy [-f]`
3. supprimer le dossier courant `.vagrant/`
4. re-vérifier la configuration du **Vagrantfile**, **réseau (IP/DNS)**
5. `vagrant up`

changement de configuration (RAM / CPU ...)

- `vagrant reload` *== halt & up*

manipuler plusieurs VMs

- si le **Vagrantfile** définit plusieurs VMs
- `vagrant up | halt | destroy` : pour *toutes les VMs*
- `vagrant up | halt | destroy |ssh <vm_name>` : *pour une VM*

configuration SSH

- `vagrant ssh` : exécute un client ssh avec une *configuration embarquée* dans l'image vagrant
- configuration avec: `vagrant ssh-config`
- utilise l'interface **NAT** créée automatiquement avec la box
- analogue à ce qu'on peut faire dans `~/.ssh/config`