Kabir AMOUSSA
1TSSIO-D GA

Déploiement d’un réseau interne multi-systèmes sous VirtualBox pour tests et validation

Tout d’abord, l’environnement de travail repose sur VirtualBox, qui permet de créer et gérer des machines virtuelles simulant un réseau complet sans matériel physique supplémentaire. Nous avons installé quatre machines. Le but est de connecter toutes ces machines installées dans un même réseau avec des adresses IP fixes pour assurer une communication fiable entre elles.

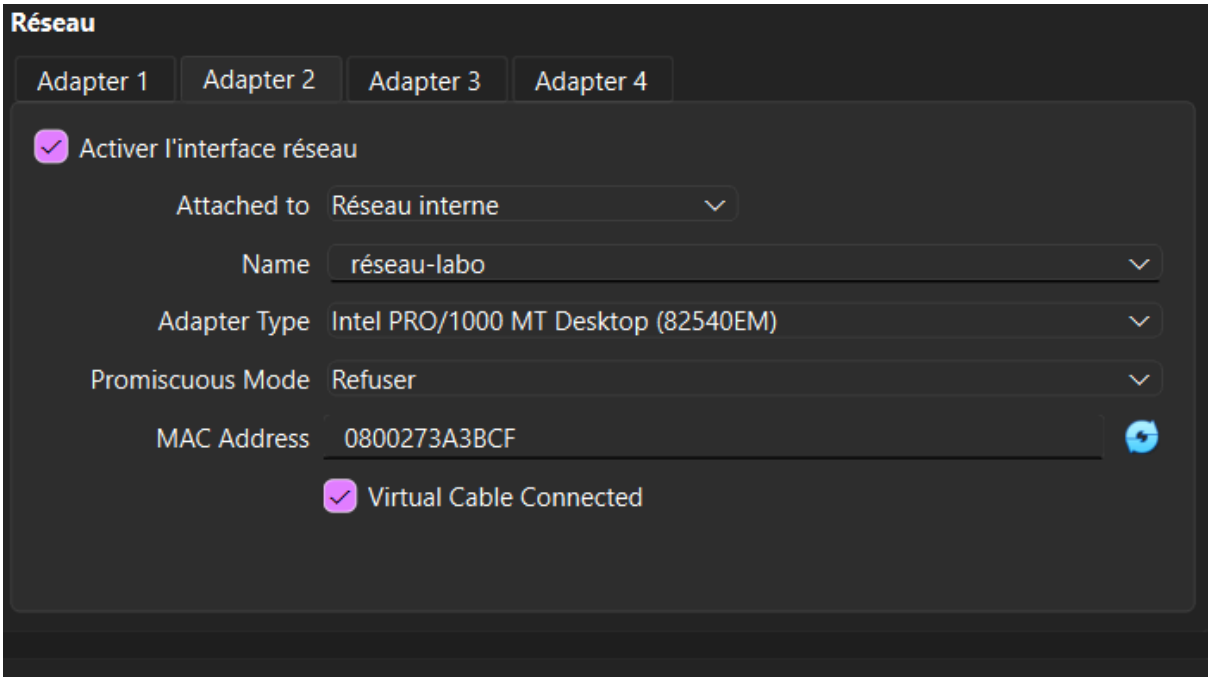
Voici le tableau récapitulatif des adresses attribuées à mes quatres machines.

Plan d’adressage:

Machine	**Système**	**Rôle**	**Interface réseau**	**Adresse IP**	**Masque**	**Passe relle**	**Mode Virtual Box**
Debian 11	Debian Server	Serveur Linux	enp0s8	172.21.10.2	255.255.255.0		Réseau Interne
VM-Windows Server	Windows Server 2019	Serveur Windows	Ethernet	172.21.10.3	255.255.255.0		Réseau Interne
VM- Ubuntu	Ubuntu Desktop	Client Linux	enp0s8	172.21.10.4	255.255.255.0		Réseau Interne
VM-Windows 10	Windows 10	Client Windows	Ethernet	172.21.10.6	255.255.255.0		Réseau Interne

Ensuite pour relier les machines virtuelles entre elles, j’ai utilisé le mode Réseau Interne de VirtualBox dans la configuration pour chaque machine. Ce qui fait que j’ai deux interfaces réseaux sur chaque machine, une qui permet d’aller sur Internet qui est mode NAT et la seconde qui permet aux quatres machines de communiquer entre elles de manière fiable.

Kabir AMOUSSA
1TSSIO-D GA



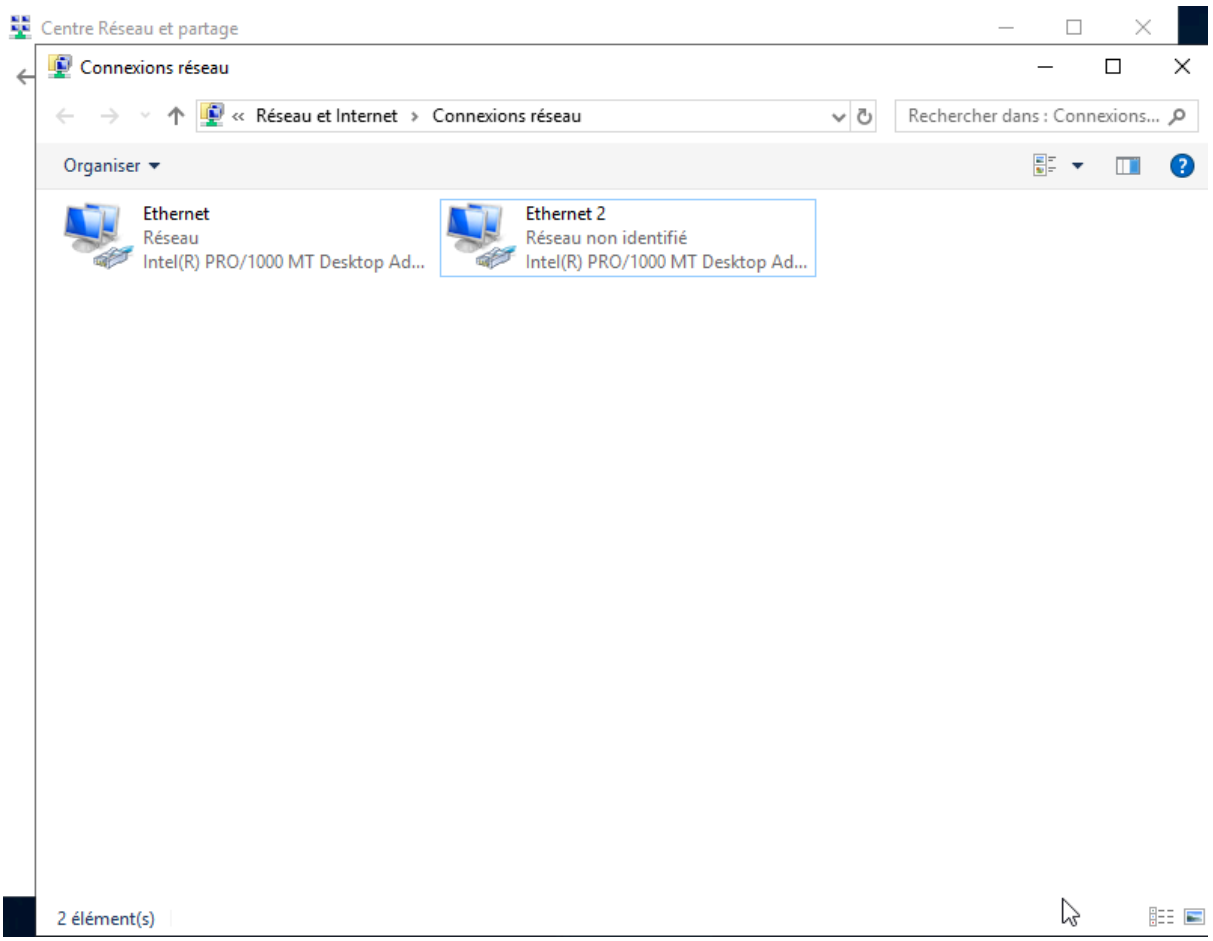
Puis, j'ai configuré les adresses IP fixes de chaque machine. Premièrement, sur Debian 11, j'ai effectué la configuration par un fichier (/etc/network/interfaces) depuis le terminal car je n'arrivais pas depuis l'interface graphique. Nous avons défini une adresse IP une adresse IP statique correspondant à notre plan d'adressage.

```
 GNU nano 5.4                  /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*
# The loopback network interface
auto lo
iface lo inet loopback
allow-hotplug enp0s3
iface enp0s3 inet dhcp
auto enp0s8
iface enp0s8  inet static
        address 172.21.10.2
        netmask 255.255.255.0
```

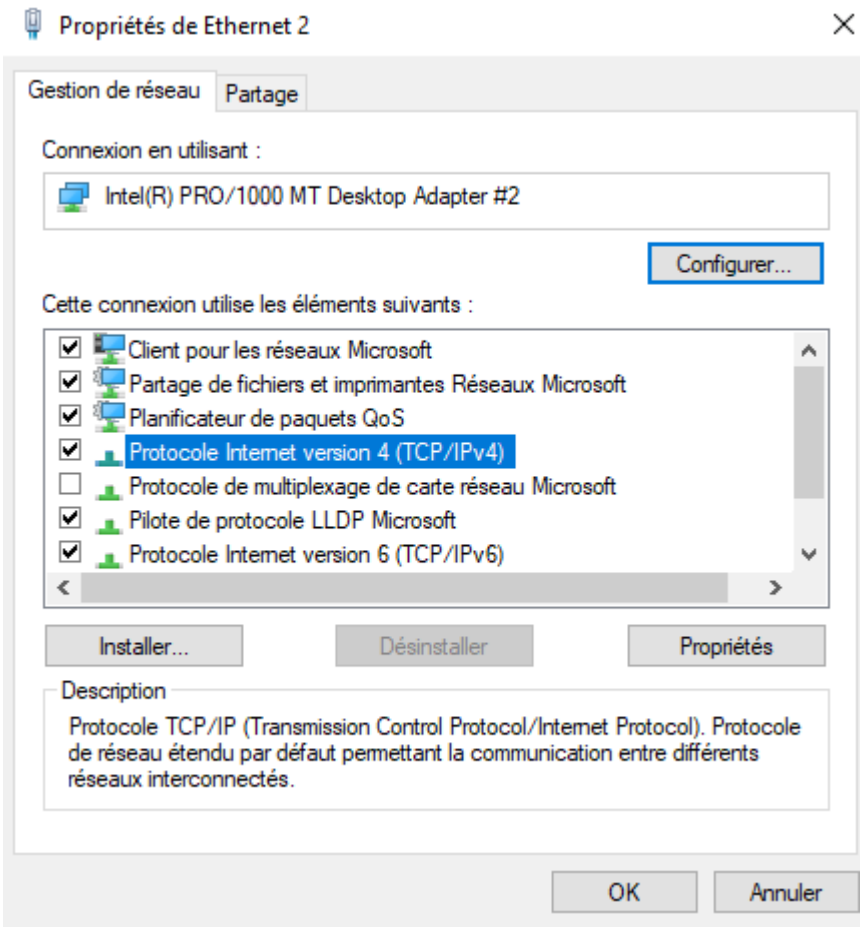
On peut voir qu'il y a la boucle locale, l'adresse qui permet à la machine de communiquer avec elle-même. Ensuite la première interface réseau nommée enp0s3 qui prend une adresse IPV4 depuis le serveur DHCP automatiquement pour se connecter à internet. Après il y a l'interface réseau pour les machines nommé enp0s8 qui prend une adresse IPV4 statique et le masque de sous-réseau. Cela permet de communiquer avec les autres machines du réseau interne.

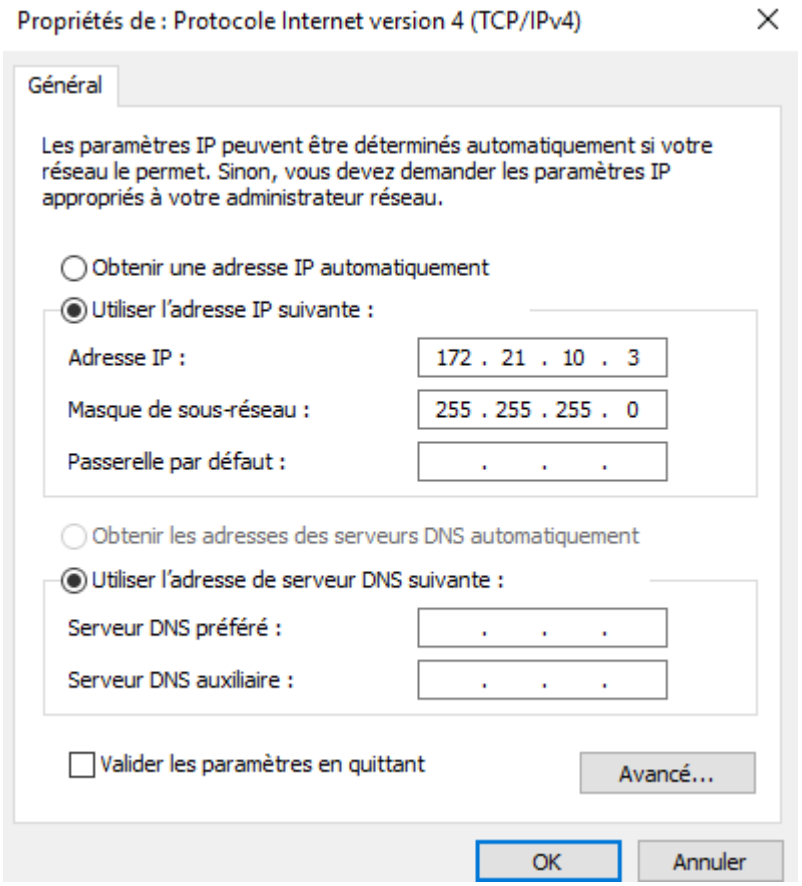
Sur Windows Server, la configuration s'effectue depuis l'interface graphique. Voici le chemin pour modifier les adresses IP :
Aller dans panneau de configuration → Réseau et Internet → Centre Réseau et partage.
Modifier les paramètres de la carte → Propriétés de la connexion Ethernet.
Sélection de Protocole Internet version 4 (TCP/IPv4)→ Cliquez sur Propriétés

Kabir AMOUSSA
1TSSIO-D GA



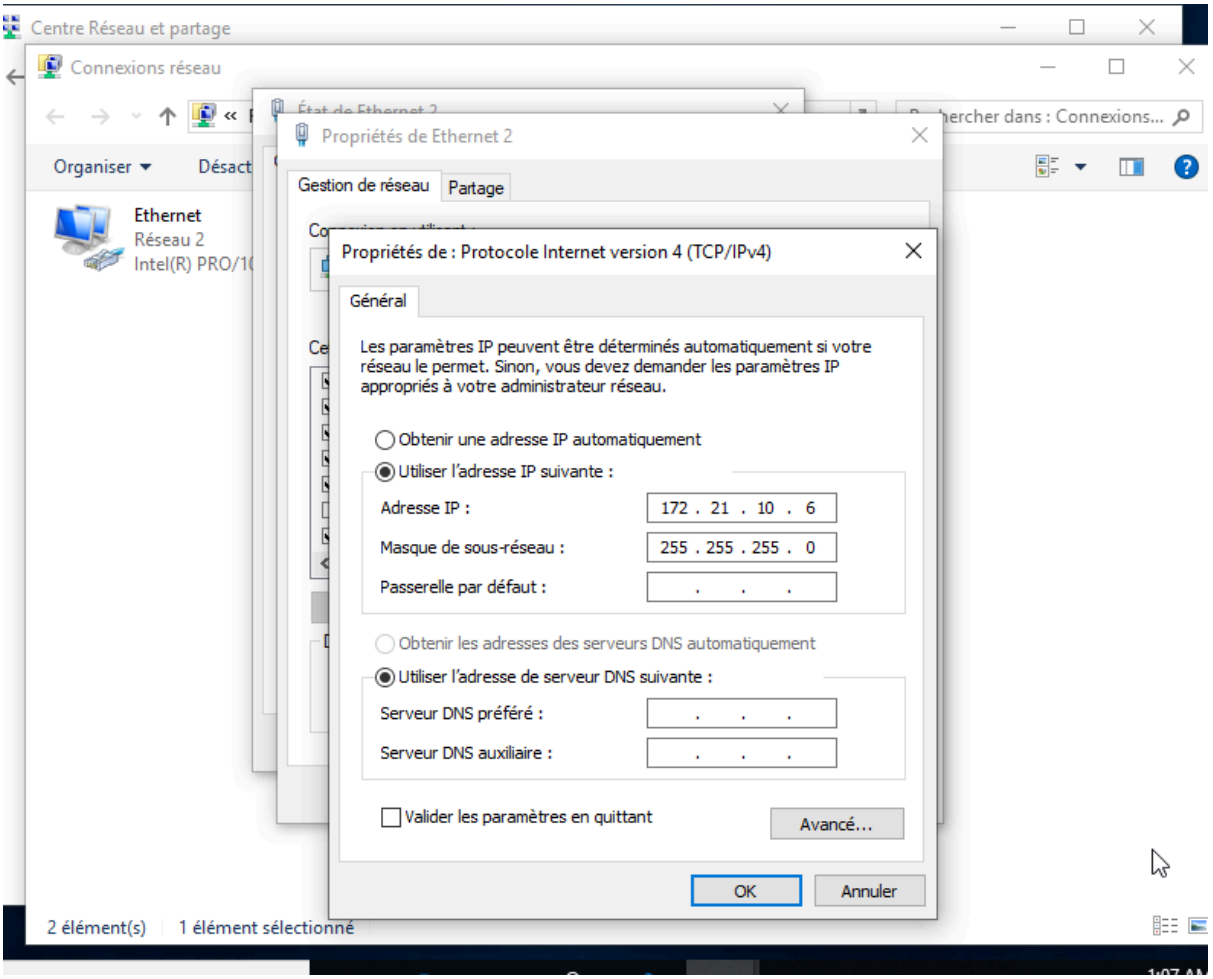
Kabir AMOUSSA
1TSSIO-D GA





Kabir AMOUSSA
1TSSIO-D GA

Pareil pour Windows 10, c'est le même processus.



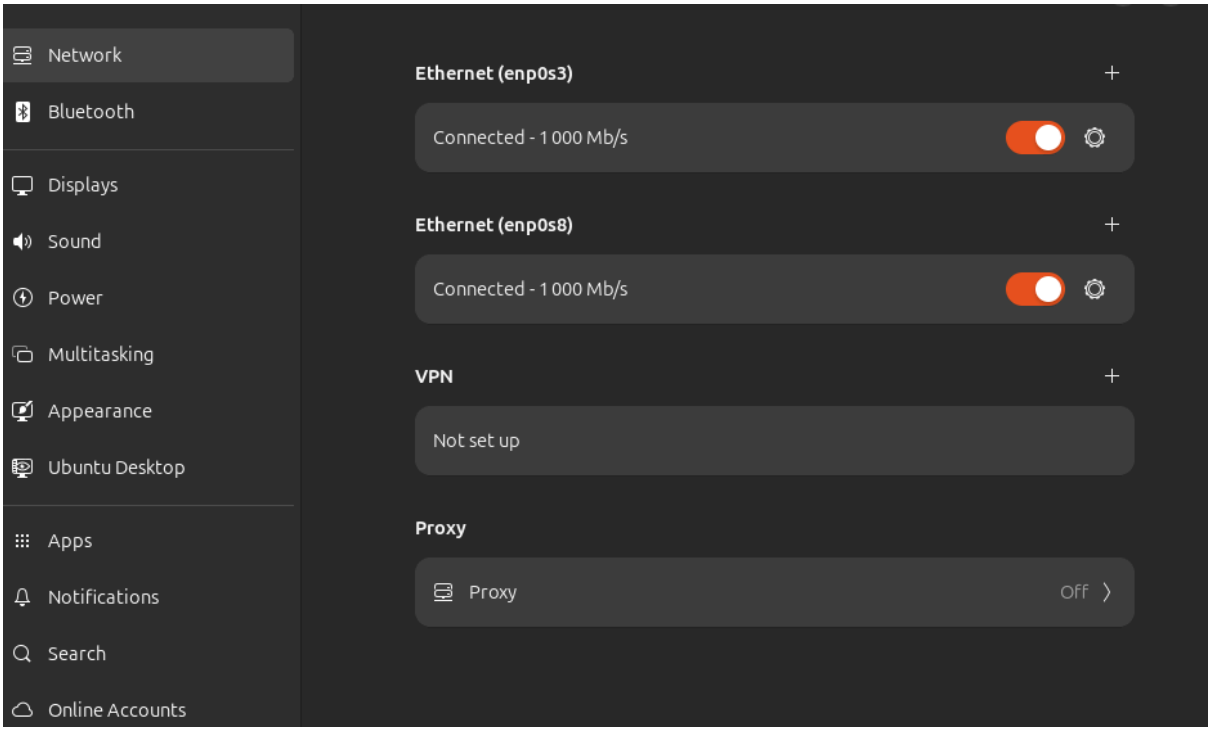
Enfin pour Ubuntu, j'ai pu ajouter l'adresse IP depuis l'interface graphique mais en passant par NetworkManager, un service qui simplifie la configuration des interfaces par une interface graphique. Mais les interfaces réseaux par défaut étaient non gérées ("unmanaged"). J'ai donc réalisé les étapes suivantes pour corriger cela.

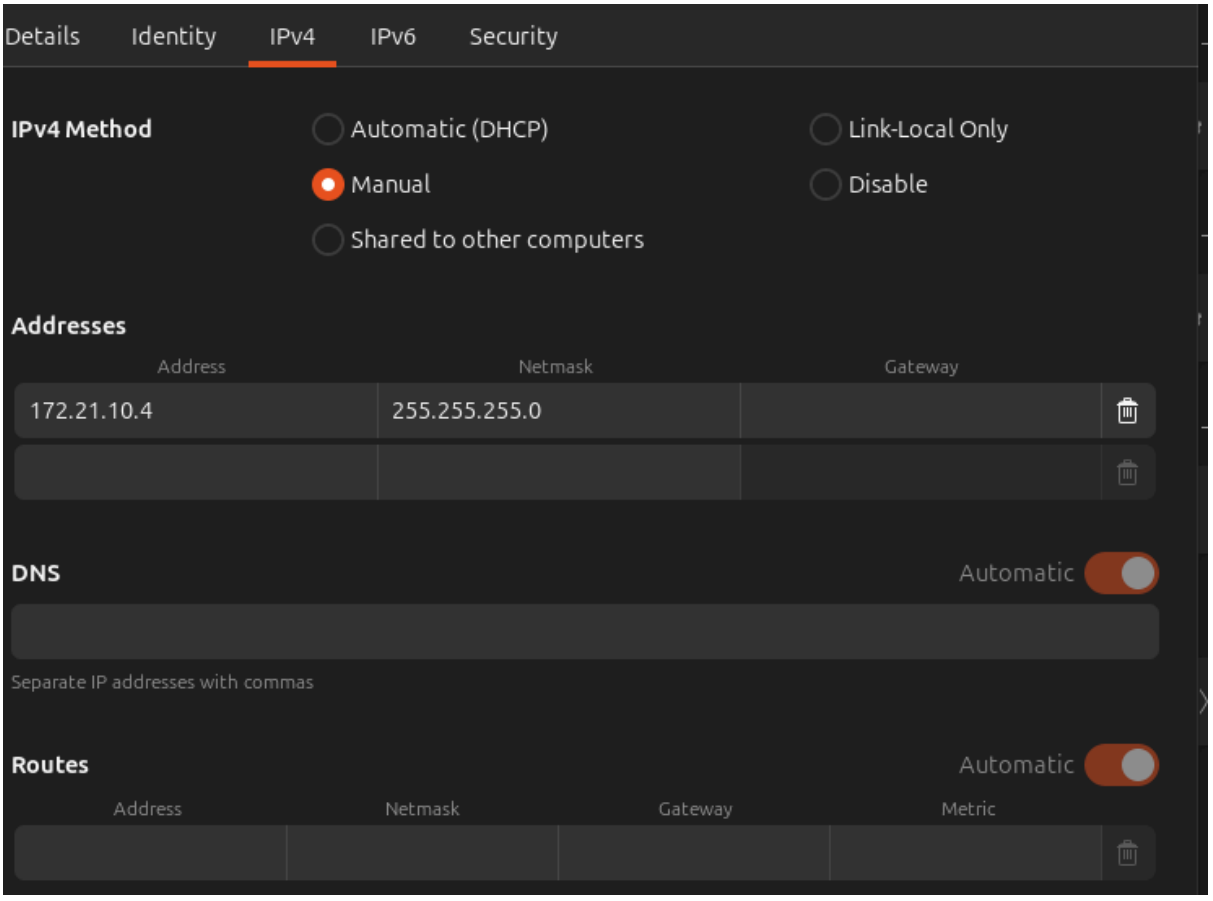
J'ai accédé aux fichiers NetworkManager.conf et changer le "managed=false en managed= true"

```
  GNU nano 7.2          /etc/NetworkManager/NetworkManager.conf
[main]
plugins=ifupdown,keyfile

[ifupdown]
managed=true

[device]
wifi.scan-rand-mac-address=no
```

Cela permet à NetworkManager de gérer les interfaces détectées par le système. Je suis allé dans les paramètres (Settings) → Réseau (Network)→ Sélectionner l'interface réseau concerné qui est enp0s8 → Aller dans IPV4 et choisir le mode manuel.

Kabir AMOUSSA
1TSSIO-D GA





J'ai essayé cette méthode aussi avec la Debian 11 qui a parfaitement fonctionné aussi. Une fois les configurations terminées, j'ai effectué des tests de connectivité avec la commande ping entre toutes les machines.
Test Debian → Ubuntu

Kabir AMOUSSA
1TSSIO-D GA

```
root@debian11:~# ping -c 4 172.21.10.4
PING 172.21.10.4 (172.21.10.4) 56(84) bytes of data.
64 bytes from 172.21.10.4: icmp_seq=1 ttl=64 time=1.83 ms
64 bytes from 172.21.10.4: icmp_seq=2 ttl=64 time=1.33 ms
64 bytes from 172.21.10.4: icmp_seq=3 ttl=64 time=1.45 ms
64 bytes from 172.21.10.4: icmp_seq=4 ttl=64 time=1.62 ms

--- 172.21.10.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.330/1.558/1.831/0.189 ms
root@debian11:~#
```

Test Windows Server → Windows 10

```
C:\Windows\system32>ping 172.21.10.6

Envoi d'une requête 'Ping'  172.21.10.6 avec 32 octets de données :
Réponse de 172.21.10.6 : octets=32 temps=7 ms TTL=128
Réponse de 172.21.10.6 : octets=32 temps=4 ms TTL=128
Réponse de 172.21.10.6 : octets=32 temps=4 ms TTL=128
Réponse de 172.21.10.6 : octets=32 temps=1 ms TTL=128

Statistiques Ping pour 172.21.10.6:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
    Minimum = 1ms, Maximum = 7ms, Moyenne = 4ms

C:\Windows\system32>
```

Sous Windows, par défaut, le pare-feu bloque les paquets ICMP (ping) entrants.
Pour autoriser Windows à recevoir des requêtes ping, j'ai cherché et ajouté une règle dans le pare-feu en mode administrateur.
Voici la commande :
netsh advfirewall firewall add rule name="Autoriser ping IPv4" protocol=icmpv4:8,any dir=in action=allow

Ces deux tests confirment la bonne communication des machines sur le réseau interne VirtualBox.