

# Mise en place d'un environnement de travail pour le développement sur une machine virtuelle Linux – Debian.

## *I – Création de la machine virtuelle.*

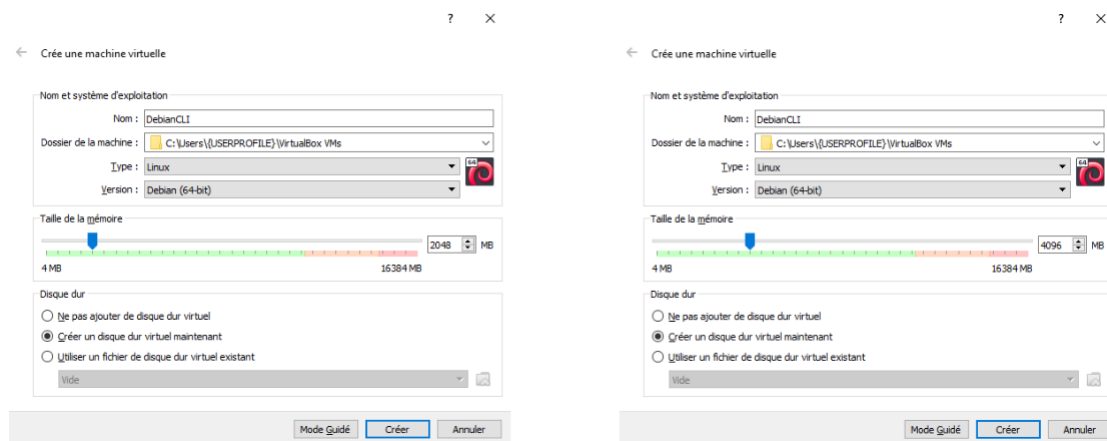
Tout d'abord, lancez [Oracle VM Virtual Box](#).

Nous allons ensuite ajouter une **nouvelle** machine virtuelle.

Selon la configuration de votre ordinateur, vous pouvez mettre plus ou moins de mémoire vive allouée à la machine virtuelle.

Mettez, au minimum, 2Go de mémoire vive.

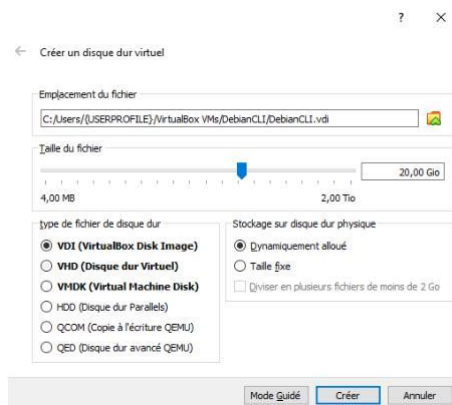
Si vous avez une configuration suffisante, 4Go de mémoire vive suffiront amplement pour cette machine virtuelle.



Ensuite, nous allons créer le disque dur virtuel.

De même que pour la mémoire vive, selon la configuration de votre ordinateur, vous pouvez créer un disque dur virtuel plus ou moins grand.

Dans tous les cas, ne créez pas un disque dur virtuel de moins de 20Go.



Téléchargez ensuite l'image système du système d'exploitation que nous allons utiliser, soit Debian. Le téléchargement se fera pendant que l'on configure la machine virtuelle.

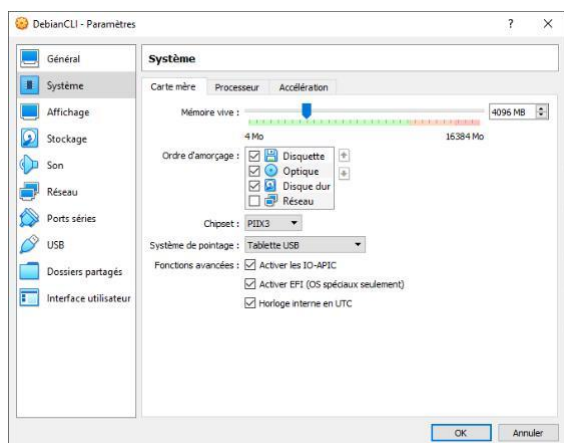
[Lien vers l'image système Linux - Debian](#)

## II – Configuration de la machine virtuelle.

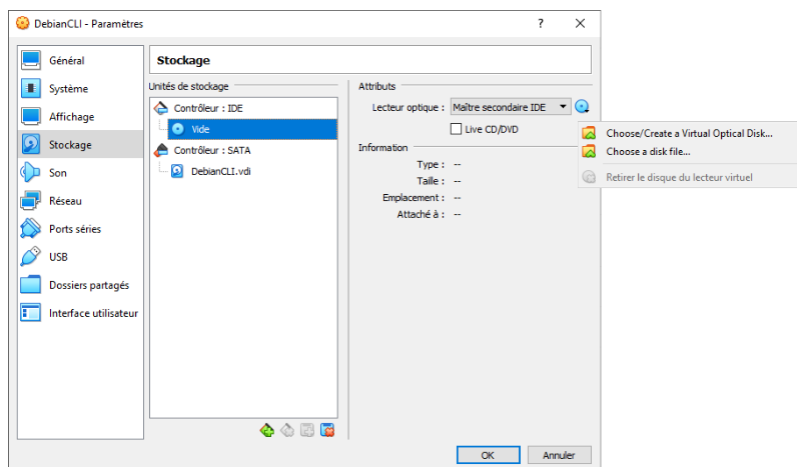
Nous allons donc faire la **configuration** de la machine virtuelle afin qu'elle puisse communiquer avec votre système d'exploitation principal.

Nous allons tout d'abord activer l'option « *Activer EFI (OS spéciaux seulement)* ».

Si vous le souhaitez, vous pouvez augmenter le nombre de cœurs de votre processeur qui seront alloués à votre machine virtuelle dans l'onglet *Processeur*.

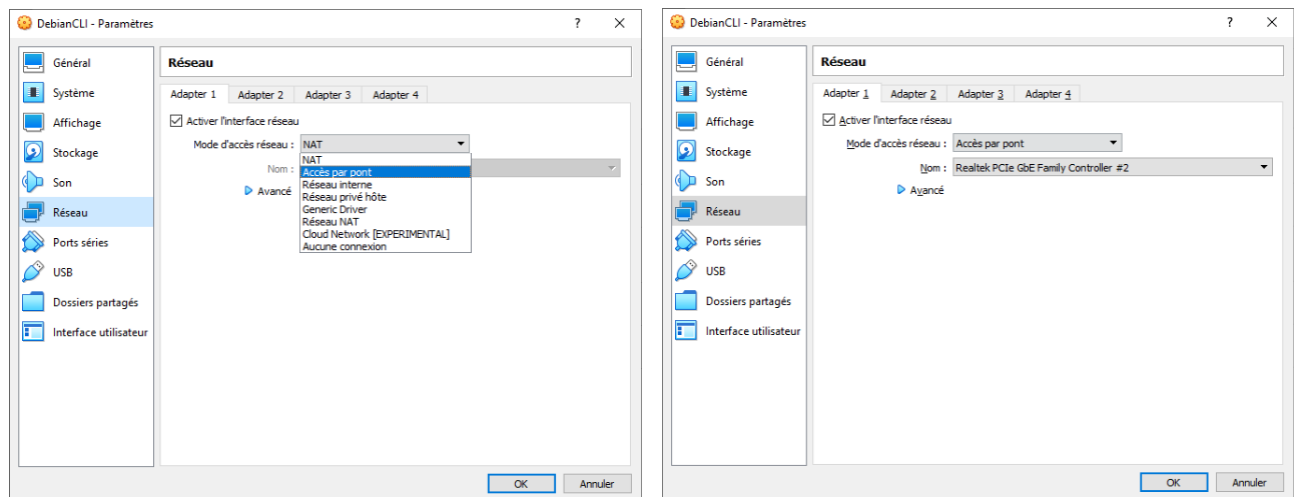


Nous allons ensuite monter l'image système de Debian, que nous avons téléchargé précédemment, afin que la machine virtuelle puisse démarrer sur le programme d'installation. Ici, vous allez cliquer sur « *Choose a disk file* » et sélectionner l'image système que vous avez téléchargé.



Nous allons ensuite configurer la connexion réseau de notre machine virtuelle.

En effet, pour qu'elle puisse communiquer avec notre ordinateur, il va falloir passer l'interface réseau en mode « *accès par pont* » (*bridged en anglais*).

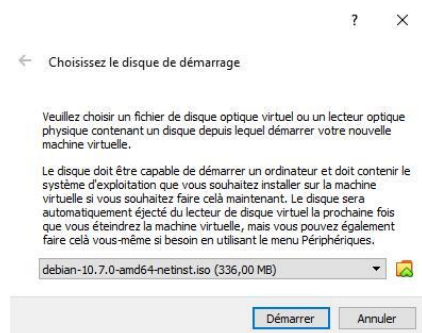


Nous avons donc fini la configuration de notre machine virtuelle.

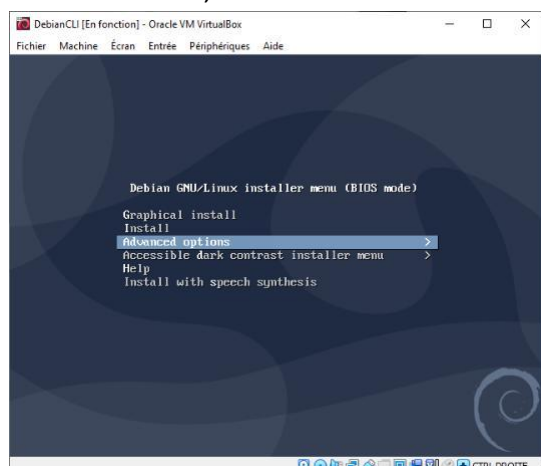
### *III – Installation du système d'exploitation.*

Nous allons maintenant installer le système d'exploitation Linux – Debian sur notre machine virtuelle.

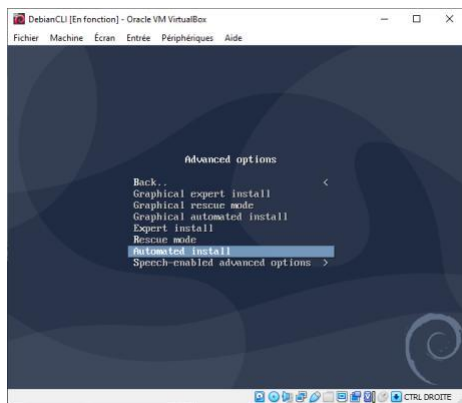
Nous allons donc démarrer notre machine virtuelle et démarrer sur l'image système téléchargée.



Une fois la machine virtuelle démarrée, nous allons, avec les flèches directionnelles et la touche *Entrée*, sélectionner « *Advanced options* ».



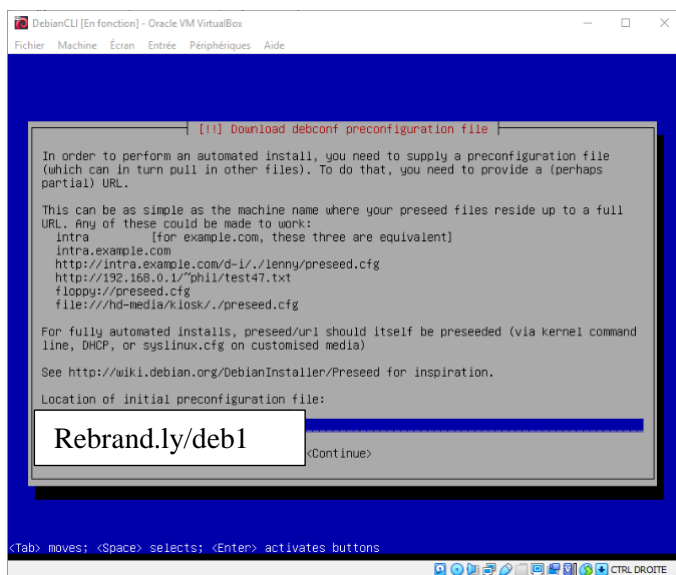
Une fois dans ce menu, nous allons sélectionner « *Automated install* ».



La machine virtuelle va mettre à jour ses paramètres réseaux.

Ensuite, vous allez devoir entrer le lien du fichier de pré configuration suivant :

[Rebrand.ly/deb1](http://Rebrand.ly/deb1)



/!\ Le clavier est en QWERTY à ce moment de l'installation. Voici comment faire les caractères modifiés :

- Q = A
- : = .
- ) = -
- ! = /
- , = M
- MAJ+M =:

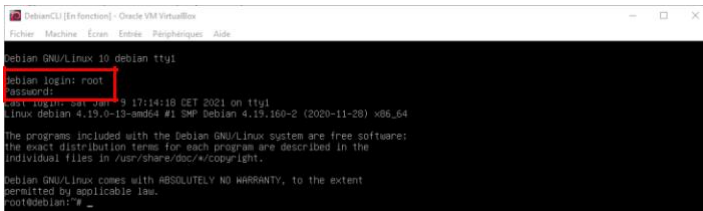
Maintenant, l'installation va se faire toute seule. Vous pouvez faire quelque chose à côté le temps que l'installation se termine.

## IV – Configuration de la connexion à distance.

Nous allons donc maintenant configurer la connexion à distance entre notre machine virtuelle Linux – Debian et notre système Windows.

Connectez-vous sur la machine virtuelle avec les identifiants suivants :

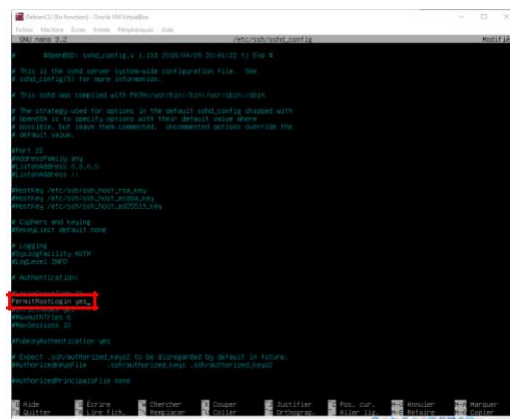
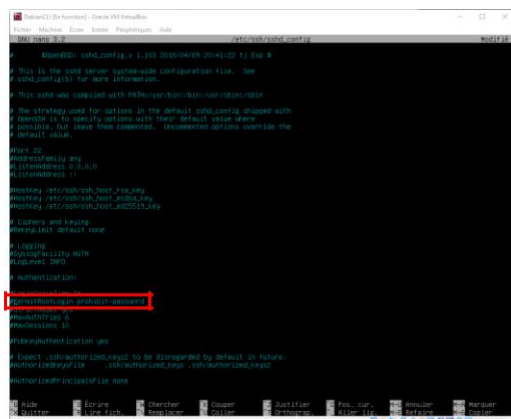
- Login : **root**
- Password : **root**



Nous allons ensuite permettre la connexion à distance via le compte « root ».

Vous allez donc écrire la commande suivante : **nano /etc/ssh/sshd\_config**

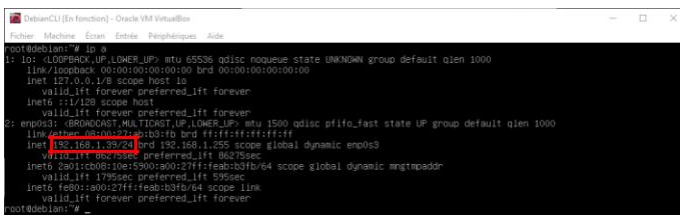
Une fois dans le fichier, vous allez modifier la ligne « **#PermitRootLogin prohibit-password** » en **PermitRootLogin yes**



Ensuite, appuyer sur **CTRL + X** puis **O** puis **Entrée** afin d'enregistrer le fichier.

Entrez ensuite la commande **systemctl restart sshd** afin de prendre en compte les modifications.

Nous allons ensuite écrire la commande **ip a** et noter l'adresse qu'elle nous renvoie.



Enfin, nous allons créer le fichier **.ssh** qui hébergera les clés en écrivant la commande suivante : **mkdir .ssh**

Nous allons maintenant réduire la fenêtre de notre machine virtuelle et ouvrir l'invite de commande Windows/Linux/MacOS.

## A – Configuration avec un système Windows.

Nous allons maintenant générer la clé de chiffrement asymétrique nous permettant une connexion sécurisée, et sans avoir à entrer le mot de passe à chaque connexion, entre notre machine virtuelle et notre ordinateur.

Nous allons donc écrire la commande suivante : **ssh-keygen -a 100 -t ed25519**

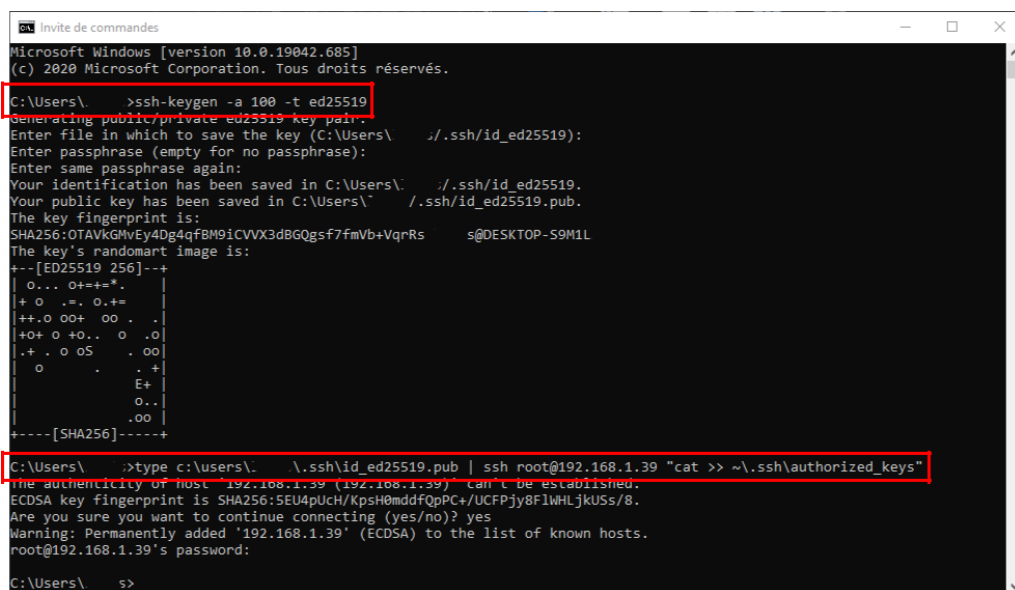
Appuyez sur la touche *Entrée* à chaque fois qu'il demande quelque chose.

La clé est maintenant créée, il faut maintenant l'envoyer à notre machine virtuelle afin de pouvoir communiquer avec elle.

Nous allons donc écrire la commande suivante :

```
type c:\users\votre_nom_utilisateur_windows\.ssh\id_ed25519.pub |  
ssh root@ladresse_ip_de_votre_machine_virtuelle "cat >> ~/.ssh/authorized_keys"
```

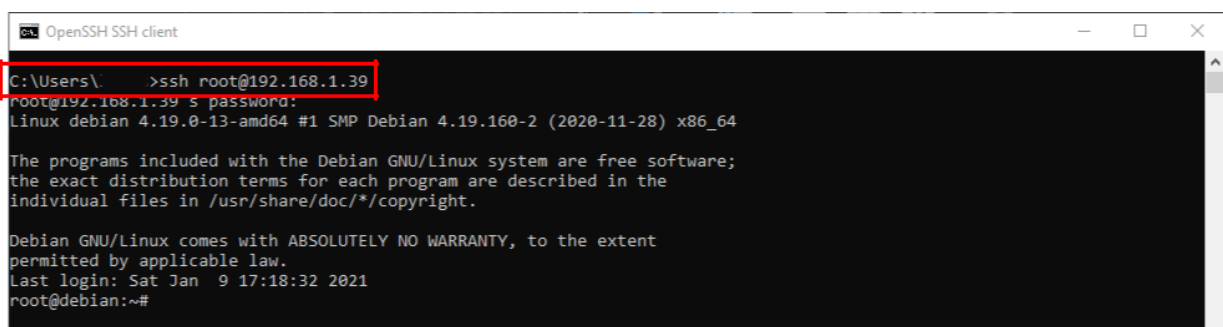
Vous répondrez **yes** à la question posée et vous entrerez le mot de passe de connexion à votre machine virtuelle, qui est **root**.



```
Microsoft Windows [version 10.0.19042.685]  
(c) 2020 Microsoft Corporation. Tous droits réservés.  
  
C:\Users\>ssh-keygen -a 100 -t ed25519  
Generating public/private ed25519 key pair.  
Enter file in which to save the key (C:\Users\> ./ssh/id_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in C:\Users\> ./ssh/id_ed25519.  
Your public key has been saved in C:\Users\> ./ssh/id_ed25519.pub.  
The key fingerprint is:  
SHA256:OTAVkGmEy4Dg4qfBM9iCVX3dBGQgsf7fmVb+VqrRs s@DESKTOP-S9M1L  
The key's randomart image is:  
+--[ED25519 256]--+  
|o... O+==+*|  
|+ O .=. O.+|  
|++o Oo+ Oo .|  
|+o+ O +O.. O .O|  
|.+. O oS . oo|  
|. O . . +|  
|. . . E+|  
|. . . O..|  
|.. oo|  
+----[SHA256]-----  
  
C:\Users\>type c:\users\> .\ssh\id_ed25519.pub | ssh root@192.168.1.39 "cat >> ~/.ssh/authorized_keys"  
The authenticity of host '192.168.1.39 (192.168.1.39)' can't be established.  
ECDSA key fingerprint is SHA256:5EU4pUcH/KpsH0mddfQpPC+/UCFPjy8f1WHLjkUss/8.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.1.39' (ECDSA) to the list of known hosts.  
root@192.168.1.39's password:  
C:\Users\>
```

Les clés sont maintenant échangées, nous allons maintenant tester la connexion SSH entre nos deux systèmes, en écrivant la commande suivante :

```
ssh root@ladresse_ip_de_votre_machine_virtuelle
```



```
OpenSSH SSH client  
  
C:\Users\>ssh root@192.168.1.39  
root@192.168.1.39's password:  
Linux debian 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sat Jan 9 17:18:32 2021  
root@debian:~#
```

Nous allons ensuite rectifier la sécurité de connexion sur notre machine virtuelle.

Vous allez donc entrer la commande suivante : **nano /etc/ssh/sshd\_config**

Nous allons modifier la ligne « *PermitRootLogin yes* » en **#PermitRootLogin prohibit-password**

```
OpenSSH SSH client
GNU nano 3.2

#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

^G Aide      ^O Écrire    ^W Chercher
^X Quitter   ^R Lire fich. ^_ Remplacer
```

```
OpenSSH SSH client
GNU nano 3.2

#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

^G Aide      ^O Écrire    ^W Chercher
^X Quitter   ^R Lire fich. ^_ Remplacer
```

Nous allons aussi modifier la ligne « *#PasswordAuthentication yes* » par **PasswordAuthentication no**

```
OpenSSH SSH client
GNU nano 3.2 /etc/ssh/sshd_config

#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no

^G Aide      ^O Écrire    ^W Chercher  ^N Couper    ^D Justifier  ^C
^X Quitter   ^R Lire fich. ^_ Remplacer  ^U Coller    ^T Orthograp. ^_
```

```
OpenSSH SSH client
GNU nano 3.2 /etc/ssh/sshd_config

#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no

^G Aide      ^O Écrire    ^W Chercher  ^N Couper    ^D Justifier  ^C
^X Quitter   ^R Lire fich. ^_ Remplacer  ^U Coller    ^T Orthograp. ^_
```

Appuyez ensuite sur **CTRL + X** puis **O** puis *Entrée* pour enregistrer le fichier.

Entrez ensuite la commande **systemctl restart sshd** afin de prendre en compte les modifications.

La configuration du SSH est donc terminée et fonctionnelle !

## B – Configuration avec un système Linux / MacOS.

Nous allons maintenant générer la clé de chiffrement asymétrique nous permettant une connexion sécurisée, et sans avoir à entrer le mot de passe à chaque connexion, entre notre machine virtuelle et notre ordinateur.

Nous allons donc écrire la commande suivante : **ssh-keygen -a 100 -t ed25519**

Appuyez sur la touche *Entrée* à chaque fois qu'il demande quelque chose.

La clé est maintenant créée, il faut maintenant l'envoyer à notre machine virtuelle afin de pouvoir communiquer avec elle.

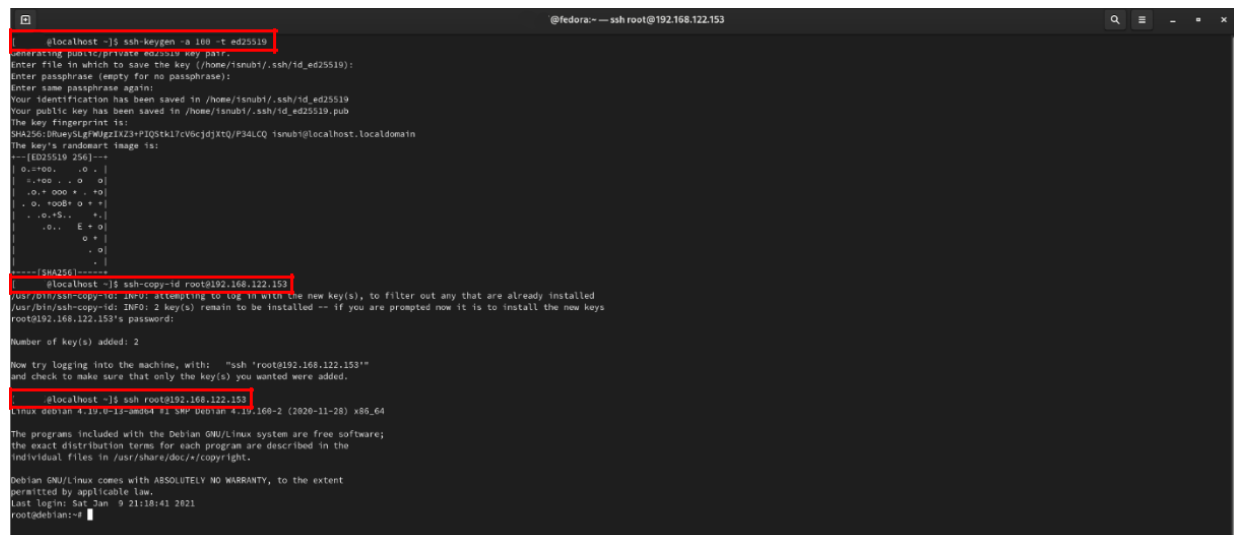
Nous allons donc écrire la commande suivante :

**ssh-copy-id root@adresse\_de\_la\_machine\_virtuelle**

Vous répondrez **yes** à la question posée et vous entrerez le mot de passe de connexion à votre machine virtuelle, qui est **root**.

Les clés sont maintenant échangées, nous allons maintenant tester la connexion SSH entre nos deux systèmes, en écrivant la commande suivante :

**ssh root@adresse\_ip\_de\_votre\_machine\_virtuelle**



```
@localhost:~$ ssh-keygen -a 100 -t ed25519
Generating public/private ed25519 key pair
Enter file in which to save the key (/home/ismubi/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ismubi/.ssh/id_ed25519
Your public key has been saved in /home/ismubi/.ssh/id_ed25519.pub
The key fingerprint is:
04256:0b0e54a6d9123+PJ0tk17cv6cjdtQ/P34LQ 1smubi@localhost.localdomain
The key's randomart image is:
+--[ED25519 256]--+
|  o==+o==  o  =+ |
| ..+o= .. o  o |
| .o+ .ooo + . +o |
| . o .+oo+ o + + |
| . .+15.. +. |
| .o.. E + o |
| . o + |
| . o |
+-----[SHA256]-----+

@localhost:~$ ssh-copy-id root@192.168.122.153
/usr/bin/ssh-copy-id: warning: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 2 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@192.168.122.153's password:

Number of key(s) added: 2

Now try logging into the machine, with: "ssh 'root@192.168.122.153'"
and check to make sure that only the key(s) you wanted were added.

@localhost:~$ ssh root@192.168.122.153
Linux debian 4.19.0-13-amd64 #1 SMP Debian 4.19.168-2 (2020-11-28) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan  9 21:18:43 2021
root@debian:~$
```



Nous allons ensuite rectifier la sécurité de connexion sur notre machine virtuelle.

Vous allez donc entrer la commande suivante : **nano /etc/ssh/sshd\_config**

Nous allons modifier la ligne « *PermitRootLogin yes* » en **#PermitRootLogin prohibit-password**

```
OpenSSH: sshd_config v 8.100 2020/04/09 20:41:22 tj Exp 5
# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.
#
# To disable tunnelled client port forwarding, change to no here!
# ForwardingAgent no
#
# Authentication:
#
# PermitRootLogin yes
#
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
# AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
```

```
OpenSSH: sshd_config v 8.100 2020/04/09 20:41:22 tj Exp 5
# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.
# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.
#
# To disable tunnelled client port forwarding, change to no here!
# ForwardingAgent no
#
# Authentication:
#
# PermitRootLogin prohibit-password
#
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
# AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
```

Nous allons aussi modifier la ligne « *#PasswordAuthentication yes* » par **PasswordAuthentication no**

```
Authentication:
LoginGraceTime 3m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
#PubkeyAuthentication yes
#
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
# AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
#
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
# HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
#
# To disable tunnelled client port forwarding, change to no here!
# ForwardingAgent no
#
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
#ChallengeResponseAuthentication no
#
# Kerberos options
#KerberosAuthentication no
#KerberosLocalizedName yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
```

```
Authentication:
LoginGraceTime 3m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
#PubkeyAuthentication yes
#
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
# AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
#
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
# HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
#
# To disable tunnelled client port forwarding, change to no here!
# ForwardingAgent no
#
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
#ChallengeResponseAuthentication no
#
# Kerberos options
#KerberosAuthentication no
#KerberosLocalizedName yes
#KerberosTicketCleanup yes
#KerberosGetAFSToken no
```

Appuyez ensuite sur **CTRL + X** puis **O** puis *Entrée* pour enregistrer le fichier.

Entrez ensuite la commande **systemctl restart sshd** afin de prendre en compte les modifications.

La configuration du SSH est donc terminée est fonctionnelle !

Votre environnement de travail pour le développement est donc prêt à être utilisé !

Documentation réalisée par  
Aaron PESCASIO

