I. Compte rendu TP1

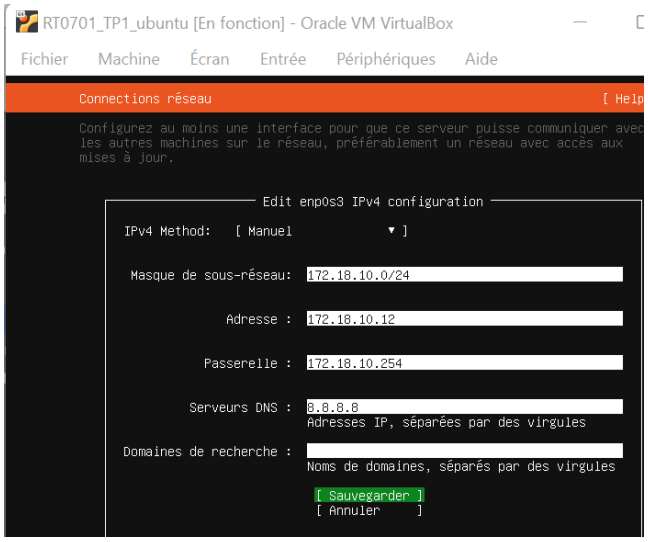
1. Architecture et configuration réseau

Créer 3 machines Linux sans interfaces graphiques qui ont pour adresses les suivantes :

Machine	Adresse	Passerelle
Debian	172.18.10.10	172.18.10.254
Alpine	172.18.10.11	172.18.10.254
Ubuntu	172.18.10.12	172.18.10.254

Il faut bien penser à mettre 2 processeurs à chaque machine sinon elles afficheront « kernel panic » et ne se lanceront pas.

a. Installation des systèmes d'exploitation



Dans un premier temps il a fallu récupérer les distributions, format iso, sur les sites respectifs de chacune des distributions. Puis j'ai configuré les adresses IP dans les invités lors de l'installation de la distribution.

J'ai eu comme difficulté l'installation de l'Alpine. En effet, à chaque redémarrage la machine perdait sa configuration précédente.

La solution que j'ai trouvée a été de l'installer sur un système d'exploitation Linux 2.6, avec comme choix de réseau le NAT et en DHCP. J'ai par la suite changé sa configuration réseau.

Mes hôtes avaient besoin d'accéder à internet pour pouvoir télécharger des paquets mais il m'était impossible d'accéder à internet. Il faut dire qu'au départ ils étaient sur un réseau privé.

Solutions trouvées :

- **Création d'un réseau NAT Virtualbox 172.18.10.0/24 qui a une gateway d'adresse 172.18.10.1**. Il n'était pas possible de la changer pour 172.18.10.254.
  Pour configurer ce réseau NAT VirtualBox, que j'ai nommé RT0701_TP1, j'ai cliqué sur Fichier > Paramètres > Réseau > Ajouter un réseau NAT. J'ai ensuite changé la configuration de mes machines virtuelles en leur passant comme mode d'accès réseau : Réseau NAT > RT0701_TP1. Enfin, puisque mon adresse de passerelle était 172.18.10.254 je l'ai changé pour 172.18.10.1. Les manipulations pour effectuer ce changement sont différentes selon la distribution.
  - Sous Ubuntu : N'ayant pas les commandes vi et nano disponibles pour changer le fichier /etc/natplan/00-installer-config.yaml j'ai utilisé la ligne de commande : `#ip`

`route replace default via 172.18.10.1.` Mon accès à Internet était ainsi établi jusqu'au moment où j'allais redémarrer ma machine. J'en ai donc profité pour télécharger nano. J'avais ainsi la possibilité de changer définitivement l'adresse de passerelle dans le fichier de configuration (00-installer-config.yaml) afin de rendre la modification permanente.

- Sous Debian : Il faut taper en super utilisateur : `nano /etc/network/interfaces`

Puis modifier ce fichier en rajoutant les lignes suivantes :

```
allow-hotplug eth0
iface enps0 inet static
      address 172.18.10.10
      broadcast 172.18.10.255
      netmask 255.255.255.0
      gateway 172.18.10.1
```

- Sous Alpine : Même principe que pour Debian.

- Une autre solution, si on souhaitait garder 172.18.10.254 comme passerelle, serait de créer **une nouvelle machine virtuelle, avec 2 cartes réseaux**, qui servirait de routeur. Une carte serait configurée dans le réseau 172.18.10.0/24 avec comme IP 172.18.10.254, et la deuxième carte réseau serait en mode bridge avec comme adresse IP celle qui me permet d'aller sur internet dans mon réseau local hôte. Utilisation du PFsense.

J'ai choisi d'opter pour la première solution, même si cela implique une légère entorse du cahier des charges.

2. Création d'utilisateurs et configuration des accès

b. Création des utilisateurs

```
#adduser superv
```

c. Donner le droit de redémarrer la machine à superv

J'ai fait la même configuration sur la machine Debian et Ubuntu :

```
#apt-get update
#apt-get upgrade
#sudo visudo
```

Puis il faut rajouter cette ligne dans le fichier pour donner les droits à superv de redémarrer la machine sans mot de passe :

```
#superv ALL=(ALL:ALL) NOPASSWD : /sbin/halt
```

J'ai testé la configuration que je venais de faire avec la commande suivante :

```
#sudo shutdown -r 0 superv
```

Pour la machine Alpine les commandes sont un peu différentes :

En effet, la distribution Alpine utilise une syntaxe légèrement différente, entre autre, pour l'installation des paquets :

```
#apk update
```

Le téléchargement de paquets posaient régulièrement des problèmes. Il m'était alors impossible de télécharger quoi que ce soit.

La première solution que j'ai trouvée a été de changer l'adresse du DNS pour mettre celle des serveurs Google.

```
#vi /etc/resolv.conf
nameserver 8.8.8.8
```

Puis j'ai décommenté les lignes du fichier repositories :
`#vi /etc/apk/repositories`

J'ai ensuite pu rentrer les commandes suivantes :

```
#apk add sudo
#echo '%wheel ALL=(ALL) ALL' > /etc/sudoers.d/wheel
#adduser superv wheel
#visudo
```

Où j'ai rajouté cette ligne :
`superv ALL=NOPASSWD : /sbin/halt`

Pour tester les modifications j'ai utilisé la commande suivante :
`sudo reboot superv`

d. Installer et configurer le serveur SSH

- Sur Debian :

La commande « `apt install openssh-server` » ne fonctionnait pas. Différentes erreurs s'affichaient en fonction des modifications apportées à source.list entre autre.

Pour qu'elle fonctionne j'ai dû supprimer le contenu du fichier source.list et le remplacer par un exemple de fichier source.list que j'ai récupéré directement sur le site de Debian. Les commandes ont donc été les suivantes :

`nano /etc/apt/source.list`

Je supprime son contenu et je le remplace par les lignes suivantes :

```
deb http://deb.debian.org/debian bullseye main
deb-src http://deb.debian.org/debian bullseye main

deb http://deb.debian.org/debian-security/ bullseye-security main
deb-src http://deb.debian.org/debian-security/ bullseye-security main

deb http://deb.debian.org/debian bullseye-updates main
deb-src http://deb.debian.org/debian bullseye-updates main
```

Le problème d'installation résolu, l'installation et la configuration du serveur ssh pouvaient commencer :

`apt-get update`

```
apt install openssh-server
sytemctl enable --now ssh.service
```

- Sur Alpine :

```
apk update
apk add openssh
rc-update add sshd
service sshd start
```

Le fichier de configuration du serveur ssh se trouve dans : `/etc/ssh/` sous le nom de `sshd_config`.

Le serveur doit refuser les connexions en tant que root

Dans le fichier « `/ect/ssh/sshd_config` » j'ai décommenté et changé le ligne suivante :

```
PermitRootLogin no
```

Il accepte seulement les connexions de superv

Pour autoriser ou interdire des utilisateurs à se connecter il faut modifier le fichier « `/ect/ssh/sshd_config` ». J'ai ensuite rentré la ligne suivante à la fin du document. Attention ce n'est pas un simple espace entre « AllowUsers » et le nom des utilisateurs mais une tabulation.

```
AllowUsers   superv
```

La machine Alpine n'accepte que les connexions SSH depuis la machine Debian, et uniquement de la part de l'utilisateur superv

Dans un premier temps j'ai seulement autorisé l'adresse de la machine Debian avec iptables sur l'Alpine:

```
iptables -I INPUT -s 172.18.10.10 -p tcp --dport ssh -j ACCEPT
iptables -I INPUT -s 172.18.10.12 -p tcp --dport ssh -j REJECT
iptables -L
service iptables save
```

Puis j'ai rajouté la condition sur l'utilisateur en rajoutant dans le fichier « `/etc/ssh/sshd_config` » la ligne suivante :

```
AllowUsers   superv
```

J'ai vérifié mes configurations depuis les machines Debian et Ubuntu :

Commande faite	Depuis la Debian	Depuis l'Ubuntu
`ssh superv@172.18.10.11`	Accès autorisé	Permission refusée
`ssh jade@172.18.10.11`	Le mot de passe n'est jamais validé.	

La machine Ubuntu n'accepte que les connexions depuis la machine Debian, et ceci sans passage de mot de passe, en utilisant un échange de clef

Tout d'abord en suivant le même principe qu'au-dessus :

```
iptables -I INPUT -s 172.18.10.10 -p tcp --dport ssh -j ACCEPT
iptables -I INPUT -s 172.18.10.11 -p tcp --dport ssh -j REJECT
```

Le serveur ssh était déjà installé sur la machine Ubuntu. Afin de mettre en place l'échange de clefs pour la connexion j'ai suivi les étapes suivantes :

- Générer des clefs SSH :
  `ssh-keygen -t rsa -b 4096`

Laisser la clef à son emplacement par défaut.
Rentrer une phrase secrète qui sert de mot de passe.

- Copier la clef publique sur le serveur distant :
  Pour établir la connexion SSH par clés SSH il faut que notre clef publique soit présente dans le fichier « `~/.ssh/authorized_keys` » de la machine Debian. Transmettre la clef à la machine Debian avec la commande : `ssh-copy-id superv@172.18.10.10`
- Tester la connexion :
  `ssh superv@172.18.10.10`

Puis, après avoir rentré la phrase de vérification, j'ai pu remarquer que j'étais bien connectée à la machine Debian.

3. Commande de contrôle d'exécution

e. Commande permettant de récupérer…

Dans un premier temps il a fallu trouver les commandes qui nous permettent d'afficher l'endroit où la donnée souhaitée est stockée :

L'adresse IP d'une machine	`ip addr show enp0s3`
L'espace disque disponible	`df -h /`
La charge CPU instantanée	La commande « `top` » ne suffit pas car elle ne donne pas la charge CPU instantanée mais seulement de la dernière minute, des 5 dernières minutes ou des dernières 15 minutes. Je télécharge donc la bibliothèque « `sysstat` » pour avoir accès à la commande « `mpstat` ».

J'ai donc utilisé les 3 commandes précédentes pour extraire les informations demandées. Afin d'extraire uniquement ce qui nous intéresse j'ai utilisé le pipe « | » et l'instruction « `awk` ». Le pipe permet d'envoyer le résultat à une seconde instruction. L'instruction « `awk` » est principalement utilisé pour la manipulation de fichiers textuels, des opérations de recherches, de remplacements et de transformations complexes.

Exemple : `awk '/motif/ {action}'` où l'action pourrait être de récupérer la valeur de la 2ième colonne par exemple.

L'adresse IP d'une machine	`ip addr show enp0s3 | awk '/inet / {print $2}'`
L'espace disque disponible	La commande « `df -h /` » permet de trouver le motif afin de récupérer la ligne puis la colonne souhaitée. La commande finale est donc : `df -h / | awk '/dev/ {print $5}'`
La charge CPU instantanée	`mpstat | awk '/all / {print $3}'`

f. Collecter depuis la machine Debian les différentes informations sur la machine Ubuntu. Il faut pouvoir déclencher les commandes depuis la machine Debian et sauvegarder les informations dans un fichier présent dans le répertoire /tmp

J'ai crée sur la machine Ubuntu un script qui m'a permis de rassembler les 3 commandes précédentes.

```
nano script.sh
```

```
#!/bin/bash
ip=$(ip addr show enp0s3 | awk '/inet / {print $2}')
use=$(df -h / | awk '/dev/ {print $5}')
cpu=$(mpstat | awk '/all / {print $3}')
heure=$(date +%H%M)
echo "$heure : machine $ip , espace disque $use , charge cpu $cpu "
```

J'ai rajouté les droits d'exécution au fichier puis je teste :

```
chmod +x script.sh
./script.sh
```

Je souhaitais me connecter à la machine Ubuntu par Debian mais ma connexion était toujours refusée.

J'ai donc vérifié l'état du ssh sur la machine Ubuntu avec la commande «`sudo service ssh status` » qui était en erreur. Pour trouver l'origine du problème j'ai utilisé la commande « `/usr/sbin/sshd -T`» et il m'a suffi de taper ces deux lignes pour que la connexion soit rétablie :

```
ssh-keygen -A
/etc/init.d/ssh start
```

Je me suis donc connectée à la machine Ubuntu par Debian afin d'exécuter le script et de stocker son résultat dans un fichier :

```
ssh superv@172.18.10.12 './script.sh' >> /tmp/ubuntu.log
cat /tmp/ubuntu.log
```

Attention à bien mettre le script.sh dans le home de superv et que ce même utilisateur ait les droits sur le fichier script.sh.

Il fallait que la machine Debian puisse se connecter sans mot de passe sur la machine Ubuntu pour activer le script. Pour cela j'ai généré une clef RSA depuis la machine Debian en suivant la même méthode que précédemment :

```
su superv
ssh-keygen -t rsa -b 4096
ssh-copy-id superv@172.18.10.12
ssh superv@172.18.10.12
```

J'ai ensuite changé les droits du fichier où l'on stocke les données :

```
chow superv /tmp/ubuntu.log
```

Enfin, j'ai testé si le scripte fonctionnait quand je lançais la commande manuellement :

```
ssh superv@172.18.10.12 './script.sh' >> /tmp/ubuntu.log
```

J'ai donc pu passer à « l'automatisation » de la collecte des données grâce à l'utilisation de Cron.

g. L'opération de collecte doit s'exécuter à intervalle régulier, toutes les 5 minutes, depuis la machine Debian sur la machine Ubuntu

Cron permet d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiée précise. Chaque utilisateur a un fichier crontab qui lui permet d'indiquer les actions à exécuter. Dans un premier temps j'ai téléchargé Cron sur la machine Debian. Cela m'a permis d'éditer la table des tâches en ajoutant une ligne qui permet d'exécuter toutes les 5 minutes en second script. J'appelle ce second scripte « script2.sh » qui permet de lancer le commande ssh vue au-dessus.

```
su superv
cd ~
pwd
nano script2.sh
ssh superv@172.18.10.12 './script.sh'>> /tmp/ubuntu.log

chmod +x script2.sh

apt-get install cron
crontab -e
5 * * * * superv /home/superv/script2.sh
ls /etc/init.d/
/etc/init.d/cron start
```

Malheureusement la planification des tâches ne fonctionne pas. L'exécution des scripts en lignes de commandes fonctionne, ./script2.sh exécute bien ./script.sh qui se trouve sur la machine Debian et rajoute les données récoltées au fichier ubuntu.log. Le problème semble donc provenir de la configuration du cron et non des scripts. Même en redémarrant la machine le problème persiste.

II. Création d'un répertoire partagé sur la machine Debian pour les utilisateurs superv grâce au protocole samba

```
apt-get update
apt-get install -y samba
systemctl enable smbd
nano /etc/samba/smb.conf
```

J'ai rajouté ces lignes à la fin du fichier `smb.conf` :

```
[partage]
       comment = Partage d un dossier
       path = /srv/partage
       read only = no
       browseable = yes
       valid users = @superv

systemctl restart smbd
```

J'ai ensuite rajouté l'utilisateur superv au dossier partagé
`smbpasswd -a superv`

J'ai créé le dossier partagé :
```
mkdir /srv/partage
chown superv partage
```

Sur les autres machines il a fallu installer la bibliothèque cliente :

```
apt-get install smbclient cifs-utils
```

Puis, la commande suivante a permis de créer le lien avec le répertoire partagé :

```
smbclient -U superv //172.18.10.10/partage
```

Enfin, j'ai testé en ajoutant un fichier dans le dossier partagé :

```
cd /srv/partage
touch fichier_test.txt
```

Je l'ai bien retrouvé sur les autres machines.

Pour plus d'information sur Samba : https://www.it-connect.fr/serveur-de-fichiers-debian-installer-et-configurer-samba-4/