TP1 - Partie 1 et 2

 Sur une **Ubuntu server 20.04** ou une Debian server 11.

Si vous êtes sur VirtualBox (et non sur RemoteLabz) télécharger l'ISO Ubuntu Server 22.04.2 sur Get Ubuntu Server .

Faire l'installation nécessaire sur VirtualBox. Utiliser le réseau NAT. Pour faciliter les configurations penser à mettre en place la redirection de ports.

Pour permettre de faire de la virtualisation sur votre VM penser à activer l'option avec la commande « `C:\'Program Files'\Oracle\VirtualBox\VBoxManage.exe modifyvm` *`nom_VM`* `--nested-hw-virt on` ».

I. Script bash de configuration

1. Écrivez un script bash non interactif effectuant une configuration persistante d'une machine.

Ce script prendra en paramètre les informations suivantes :

- Nom à affecter à l'hôte
- Identifiant de la carte réseau à configurer
- Adresse de la carte réseau
- Adresse de la passerelle
- Adresse du DNS

Le script réalisera les opérations suivantes :

1. Affecte un nom à l'hôte
2. Vérifie l'existence de l'interface réseau
3. Désactive la carte réseau
4. Modifie l'adresse de la carte réseau
5. Active l'interface réseau
6. Active si nécessaire le service réseau
7. Modifie l'adresse du DNS
8. Teste la connexion au réseau

Des tests seront effectués à chaque étape, et une sortie d'erreur sera effectuée en cas de problème, vous différencierez les erreurs.

Attention, si vous faite ce TP depuis une connexion SSH les modifications réalisées ci-dessous vous déconnecteront.

Dans un premier temps créer le script qui contiendra les commandes :

```
touch config_reseau.sh
chmod u+x config_reseau.sh

nano config_reseau.sh
```

a. Un script qui respecte les consignes avec des lignes de commandes

Puis voici les commandes pour réaliser les opérations demandées :

Rappel : l'argument $0 est le nom du script.

```
#! /bin/bash
nom_machine=$1
nom_interface=$2
ip_interface=$3
ip_passerelle=$4
ip_dns=$5

function ip_incorrecte(){
        local  ip=$1
        local  stat=1
        if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]
        then
                OIFS=$IFS
                IFS='.'
                ip=($ip)
                IFS=$OIFS
                [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 && ${ip[2]} -le 255 &&
${ip[3]} -le 255 ]]
                stat=$?
        fi
        if [ $stat -eq 1 ]
        then
                return 0
        else
                return 1
        fi
}

function ip_masque_incorrecte(){
        ip_masque=$1
        OIFS=$IFS
        IFS='/'
        ip_masque=($ip_masque)
        IFS=$OIFS
        if  ip_incorrecte ${ip_masque[0]}
        then
                return 0
        else
                if [[ ${ip_masque[1]} =~ ([1-9]|[1-2][0-9]|3[0-2])$ ]]
                then
                        return 1
                fi
                return 0
        fi
}

#Vérification du type d'OS
if [[ "$OSTYPE" == "linux-gnu"* ]]
then
    type_os_ubuntu=$(cat /etc/os-release | grep -c Ubuntu)
    if [[ $type_os_ubuntu -gt 0 ]]
    then
          echo "La machine est une Ubuntu, installation par le fichier Netplan."
    else
          echo "La machine n'est pas Ubuntu, installation en ligne de commande."
    fi
else
      echo "La machine n'est pas une distribution Linux. Impossible de configurer
le réseau. Exit erreur 1."
```

```
        exit 1
fi

##############################
###Vérification des arguments
##############################

##Vérification du nombre d'argument
if [ $# -ne 5 ]
then
        echo il manque des arguments. Exit erreur 2.
        exit 2

##Vérfication de l'existance de l'interface rentrée par l'utilisateur
elif [ $(ip a show | cat | grep -c $nom_interface) -eq 0 ]
then
        echo "L interface rentrée n existe pas. Exit erreur 3."
        exit 3

##Vérification du troisième arguement : ip+masque de l interface
elif ip_masque_incorrecte $ip_interface
then
        echo "L ip de l interface n est pas valide. Exit erreur 4."
        exit 4

##Vérification du 4ieme arguement : ip passerelle
elif ip_incorrecte $ip_passerelle
then
        echo "L ip de la passerelle n est pas valide. Exit erreur 5."
        exit 5

##Vérification du 4ieme arguement : ip dns
elif ip_incorrecte $ip_dns
then
        echo "L ip du dns n est pas valide. Exit erreur 6."
        exit 6

else

##############################
###Debut des configurations
##############################
        #Affecte un nom à l hote
        $(hostname $nom_machine)

        type_os_ubuntu=$(cat /etc/os-release | grep -c Ubuntu)
        if [[ $type_os_ubuntu -gt 0 ]]
        then
          echo -e "network:\n  ethernets:\n    $nom_interface:\n      addresses:
[$ip_interface]\n      gateway4: $ip_passerelle\n      nameservers:\n
addresses: [$ip_dns]\n  version: 2" > /etc/netplan/*.yaml
          netplan apply

        else

##############################
###La machine est une Debian
##############################

        #Interface choisie mise à DOWN
          ip link set $nom_interface down

          #Nettoyage de l'interface : suppression de l ancienne IP si elle existe
```

Attention aux espaces ! Deux espaces sont équivalent à « une tabulation ». L'utilisation de la touche « Tab » ne fonctionne pas. Pour que le retour à la ligne soit interprété rajouter le paramètre « -e » après le « echo »

```
        echo $(ip a show $nom_interface) > interface.txt
        nbr_inet=$(cat interface.txt | grep -c inet)
        if [ $nbr_inet -gt 0 ]
        then
                ip_a_supp=$(awk '{print $19 }' interface.txt)
                ip addr del $ip_a_supp dev $nom_interface
        fi

        #Attribution de la nouvelle adresse IP à l'interface
        ip addr add $ip_interface dev $nom_interface
        ip link set $nom_interface up
        ip route add $ip_passerelle dev $nom_interface
        ip route add default via $ip_passerelle dev $nom_interface

        #Changement de l'adresse du DNS
        dns="nameserver $ip_dns"
        echo $dns > /etc/resolv.conf
    fi

################################
###Test du réseau
################################

        #Test de la connexion avec un ping
        echo $(ping -q -c 1 google.fr) > ping.txt
        paquet_perdu=$(awk '{print $18}' ping.txt)
        if [ $paquet_perdu = 0% ]
        then
                echo "Le réseau fonctionne"
        else
                echo "Le réseau ne fonctionne pas. Exit erreur 2."
                exit 2
        fi
fi
exit 0
```

Ne pas oublier de mettre les 2 lignes pour la passerelle.

b. La ligne de commande pour exécuter le script

La commande pour appeler le script est par exemple :

```
./config_reseau.sh ubuntu ens3 10.11.1.193/28 10.11.1.206 8.8.8.8
```

II. Reprises des scripts du cours

Vous trouverez à la fin des transparents du cours les 3 descriptions suivantes. Développez successivement les scripts correspondants en bash et en python.

1. Récupération d'informations sur un utilisateur

Description du script :

- Pas de paramètres
- Création d'un nouveau fichier à chaque utilisation, les anciens fichiers ne doivent pas être détruits !
- Affichage et enregistrement :
  - De l'utilisateur
  - De l'ID de l'utilisateur
  - Des groupes de l'utilisateur

a. BASH

```
touch info_utilisateur.sh
sudo chmod u+x info_utilisateur.sh

nano info_utilisateur.sh

#! /bin/bash

#user
nom_user=$(who -m | awk '{print $1}')

#UID du user
id_user=$(cat /etc/passwd | grep toto | awk '{split($0,a,":"); print a[3]}')

#Groupes du user
group_user=$(groups | awk '{for (i=2; i<=NF; i++) printf $i " "; print $NF}')
date_creation=$(date +"%d_%m_%Y_%H_%M")
FILE=$nom_user-$date_creation.txt

if [ -f "$FILE" ]
then
        echo "Impossible d executer ce script... attendez une minute"
else
        echo -e "User: $nom_user\nID: $id_user\nGroups: $group_user" > $FILE
fi
exit 0

./info_utilisateur.sh
```

b. Python

```
import os, pwd, grp, datetime

nom_user=pwd.getpwuid(os.getuid())[0]

id_user=pwd.getpwuid(os.getuid())[2]

tab_gid=os.getgroups()

date=datetime.datetime.now()
str_date=str(date.day)+ "_" + str(date.month)+ "_" + str(date.year)+ "_" +
str(date.hour)+ "_" + str(date.minute)
```

```
nom_fichier=nom_user + "_" + str_date

tab_nom_group=""
for gid in tab_gid :
        tab_nom_group = tab_nom_group + str(grp.getgrgid(gid).gr_name) + " "

contenu_fichier= "User: "  + str(nom_user) + "\nID: " + str(id_user) + "\nGroups:
" + str(tab_nom_group) + "\n"

if (os.path.isfile(nom_fichier)):
        print("Impossible d executer ce script... Attendez une minute")
else:
        Fichier=open(nom_fichier, "w+")
        Fichier.write(contenu_fichier)
        Fichier.close()
```

2. Un script de sauvegarde

Description du script :

- Paramètres
  - Le nom de l'archive
  - Le nom du répertoire à sauvegarder
  - L'adresse du serveur de sauvegarde
  - Le login et le mot de passe du compte
- L'archive est réalisée avec tar
- Après archivage, copie à travers du réseau de l'archive (vous installerez un serveur SFTP en local)

a. Mise en place du service SFTP sur une seconde machine Ubuntu

Pour cette partie il vous faudra une seconde machine Ubuntu. Après avoir paramétré son réseau, paramétrer le service SFTP sur cette dernière en suivant les étapes suivantes :

- Dans un premier temps créer un répertoire qui hébergera nos données FTP :

```
mkdir dossier_sftp
chmod 701 dossier_sftp
```

- Créer un groupe spécial pour les utilisateurs du service SFTP :

```
sudo groupadd sftp_users
```

- Créer un utilisateur qui n'a pas d'accès privilégiés. Cet utilisateur appartiendra au groupe créé à l'étape précédente :

```
sudo useradd -g sftp_users -d /upload -s /sbin/nologin sftp_user
```

- Donner un mot de passe au nouvel utilisateur grâce à la commande :

```
passwd sftp_user
```

- Créer un répertoire de téléchargement, spécifique au nouvel utilisateur :

```
sudo mkdir -p /dossier_sftp/sftp_user/upload
sudo chown -R root:sftp_users /dossier_sftp/sftp_user
sudo chown -R sftp_user:sftp_users /dossier_sftp/sftp_user/upload
```

- Configurer SSHD :
  - Ouvrir le fichier de configuration du démon SSH avec la commande :

```
sudo nano /etc/ssh/sshd_config
```

  - Rajouter en bas du fichier les lignes suivantes :

```
Match Group sftp_users
ChrootDirectory /dossier_sftp/%u
ForceCommand internal-sftp
```

- Enregistrer et redémarrer SSH avec la commande :

```
sudo systemctl restart sshd
```

Afin de tester les configurations réalisées retourner sur la première machine Ubuntu et tester la connexion : `sftp` `sftp_user@10.11.1.194`. Après avoir rentré le mot de passe de l'utilisateur le terminal affiche l'invite de commande SFTP. Le chemin de travail devrait être `/upload` quand vous tapez `pwd`.

b. Retour sur le script… BASH

```
touch sauvegarde.sh
sudo chmod u+x sauvegarde.sh

nano sauvegarde.sh

#! /bin/bash

nom_archive=$1
nom_repertoire=$2
ip_serveur_svg=$3
login=$4
mdp=$5

##Vérification du nombre d'argument
if [ $# -ne 5 ]
then
        echo il manque des arguments. Exit erreur 1.
        exit 1
else

    #On vérifie que le répertoire a archiver existe
    if [ -d "$nom_repertoire" ]
    then

            #Impossible d'archiver un dossier vide: vérifier qu'il ne le soit pas

            if [ "$(ls -A $nom_repertoire)" ]
            then
                tar -cvf $nom_archive.tar $nom_repertoire

                #On vérifie si le paquet sshpass est installé
                if [ $(which sshpass | grep -c sshpass) -eq 0 ]
                then
                    echo Installation du paquet SSHPASS nécessaire.
                    apt-get install sshpass
                fi
                echo "Le paquet sshpass est déjà installé. Transfert du
fichier..."
sshpass -p "$mdp" sftp "$login"@"$ip_serveur_svg" <<!
put $nom_archive.tar
exit
!
            else
                echo "Le dossier est vide il ne peut pas par être archiver. "
                exit 5
            fi
    else
            echo "Le repertoire a archiver n existe pas. Exit erreur 2."
            exit 2
```

Caractère de délimitation. Possibilité de mettre un autre caractère, au choix.

Il ne faut pas d'espaces à gauche ! Donc pas d'indentation.

```
    fi
fi

echo "Le dossier $nom_repertoire a bien été envoyé sous le nom de
$nom_archive.tar"
exit 0
```

c. Python

```
import sys, os, tarfile, paramiko

if (len(sys.argv) == 6):
        nom_archive=sys.argv[1]
        nom_repertoire=sys.argv[2]
        ip_serveur_svg=sys.argv[3]
        login=sys.argv[4]
        mdp=sys.argv[5]

        #On vérifie que le répertoire a archiver existe
        if (os.path.isdir(nom_repertoire)) :
                #impossible d archiver un dossier vide: verifier qu il ne le soit
pas
                if len(os.listdir(nom_repertoire)) == 0:
                        print(nom_repertoire + " est vide, impossible d archiver
un dossier vide")
                else:
                        nom_archive_tar=nom_archive+".tar"
                        with tarfile.open(nom_archive_tar, "w") as tar:
                                tar.add(nom_repertoire)

                        transport = paramiko.Transport((ip_serveur_svg,22))
                        transport.connect(None, login, mdp)
                        sftp = paramiko.SFTPClient.from_transport(transport)
                        sftp.put(nom_archive_tar,nom_archive_tar)
                        if sftp: sftp.close()
                        if transport: transport.close()

        else :
                print(nom_repertoire + " n existe pas.")
else:
        print("Le nombre d argument est incorrecte")
```

3. Un script d'installation

Description du script :

- Paramètres :
  - Le nom de l'archive
  - Le répertoire de base d'installation
- Vérification des existences et des droits
- Test du type d'archive (zip, tar, tgz) (Uniquement sur l'extension)
- Utilisation de l'archiveur correspondant

a. Bash

```
touch installation.sh
sudo chmod u+x installation.sh

nano installation.sh

#! /bin/bash

nom_archive=$1
```

```
nom_repertoire=$2

##Vérification du nombre d'argument
if [ $# -ne 2 ]
then
        echo il manque des arguments. Exit erreur 1.
        exit 1
else
    #On vérifie que l archive à installer existe
    if [ -f "$nom_archive" ]
    then
        proprio_archive=$(ls -l | grep $nom_archive | awk '{print $3}')
        user=$(who -m | awk '{print $1}')
        droits_autres=$(ls -l | grep $nom_archive | awk '{print $1}' | cut -b
8-10)

        #On vérfie les droits du répertoire
        #On vérifie si l'utilisateur connecté est la propriétaire du
répertoire
        #           ou les droits du groupe others
        #           ou si l'utilisateur est l utilisateur root
        if [ $proprio_archive = $user ] || [ $(echo droits_autres | grep r) ]
|| [ $proprio_archive = "root" ]
        then

            #On vérifie que le répertoire d extraction existe, sinon on le
crée
            if [ -d "$nom_repertoire" ] || [ "$nom_repertoire" = "." ]
            then
                echo Le répertoire existe déjà. Début de l extraction...
            else
                mkdir -p $nom_repertoire
            fi

            #On regarde le type d extension
            extension=$(echo $nom_archive | awk '{split($0,a,"."); print
a[2]}')

            if [ $extension = "tar" ]
            then
                tar xf $nom_archive -C $nom_repertoire

            elif [ $extension = "zip" ]
            then
                if [ $(which unzip | grep -c unzip) -eq 0 ]
                then
                    echo Installation du paquet UNZIP nécessaire.
                    sudo apt install unzip
                fi
                unzip  $nom_archive -d $nom_repertoire

            elif [ $extension = "tgz" ]
            then
                tar zxf $nom_archive -C $nom_repertoire

            else
                echo "L extension de votre archive ( $extension ) est inconnue
de ce script."
                exit 3
            fi

        else
```

Les archives ne sont pas des dossiers mais des fichiers. Donc ne pas tester avec « d ».

```
                echo "Vous n etes pas le proprietaire de '$nom_archive' et/ou vous
    n avez pas les droits d execution."
                exit 2
            fi
        else
            echo "L archive n existe pas."
            exit 1
        fi

    fi
    exit 0
```

b. Python

```
import sys, os, tarfile, zipfile

if (len(sys.argv) == 3):
        nom_archive=sys.argv[1]
        nom_repertoire=sys.argv[2]

        #On vérifie que le répertoire a archiver existe
        if os.path.isfile(nom_archive):
                #On vérifie les droits de l archive
                if os.access(nom_archive, os.R_OK):
                        #On vérifie que le répertoire d extraction existe
                        if (os.path.exists(str(nom_repertoire))) :
                                print("Le répertoire existe. Extraction...")
                        else :
                                os.makedirs(nom_repertoire)
                                print("Le répertoire : " + nom_repertoire + " a
été créé. Extraction...")
                        tab_extension=nom_archive.split(".")
                        extension=tab_extension[1]

                        if (extension == "tar") or (extension == "tgz"):
                                tar = tarfile.open(nom_archive)
                                tar.extractall(nom_repertoire)
                                tar.close()
                        elif (extension == "zip"):
                                zip = zipfile.ZipFile(nom_archive, mode="r")
                                zip.extractall(nom_repertoire)
                                zip.close()
                        else:
                                print("L extension n est pas gérée dans ce
script.")
                else:
                        print("Vous n avez pas les droits")
        else:
                print("Votre archive" + nom_archive + "n existe pas")
else :
        print("Le nombre d argument n est pas bon.")
```