ESGI
école supérieure de génie informatique

06/07/2020

ADMINISTRATION LINUX

Notice d'utilisation

Joëlle CASTELLI - Noé LARRIEU-LACOSTE
2A2

Script 1 : création automatique de 10 utilisateurs utilisables nommés userX où X est un nombre aléatoire (le mot de passe est le nom de l'utilisateur), création de 5 à 10 fichiers d'une taille aléatoire entre 5Mo et 50Mo dans les répertoires personnels de chaque utilisateur.

```bash
#!/bin/bash

# On déclare les constantes
declare -r ROOT_UID=0
declare -r SUCCESS=0
declare -r ERROR=13
declare -r NB_USERS=10

# On vérifie que l'utilisateur soit root
if [ $UID -ne $ROOT_UID ]
then
   echo "Vous devez être root pour lancer ce script !"
   exit $ERROR
fi

i=1
while [ $i -le $NB_USERS ]
do
   # On génère un nom aléatoire qui sera également le mot de passe
   username=user$RANDOM
   password=$username

   # On vérifie si l'utilisateur existe déjà
   grep -q "$username" /etc/passwd
   if [ $? -eq $SUCCESS ]
   then
      # Si oui : on recommence le processus sans incrémenter le compteur
      echo "$i - L'utilisateur $username existe déjà : génération d'un nouveau nom"
   else
      # Si non : on crée un utilisateur avec un répertoire personnel et sans mot de passe
      useradd -m -g users -s /bin/bash "$username"
      if [ $? -eq $SUCCESS ]
      then
         echo -e "\n$i - Le compte de $username est créé !"

         # On set un mot de passe en renvoyant le prompt vers /dev/null pour ne pas polluer la console
         echo -e "$password\n$password" | passwd $username >& /dev/null

         # On génère aléatoirement le nombre de fichiers à créer
         nb_files=$((RANDOM%6+5))
         echo "On va créer $nb_files fichiers..."

         # Pour chaque fichier à créer
         for j in `seq 1 $nb_files`
         do
            # On définit le nom, l'utilité de taille et une taille aléatoire
            file_name=file$j
            size_unit=M
            file_size=$((RANDOM%46+5))

            # On alloue la mémoire nécessaire au fichier
            fallocate -l $file_size$size_unit /home/$username/$file_name
            if [ $? -eq $SUCCESS ]
            then
               echo "Le fichier $file_name de $file_size Mo a été créé !"
            else
               echo "Aïe, problème dans la création du fichier $file_name :("
            fi
         done
      else
         echo -e "\nAïe, problème dans la création de l'utilisateur $username :("
      fi

      # On incrémente le compteur pour créer l'utilisateur suivant
      i=$(($i + 1))
   fi
done

exit $SUCCESS
```

Script 2 : calcul de la taille des répertoires personnels de tous les utilisateurs humains du système, classement par ordre décroissant d'espace disque grâce à un tri shaker, affichage des 5 plus gros consommateurs d'espace à la connexion et affichage d'un avertissement pour chaque utilisateur lorsqu'il occupe plus de 100 Mo.

```bash
#!/bin/bash

# On déclare les constantes et on inclut le fichier de fonctions
declare -r TRUE=0
declare -r FALSE=1
declare -r ROOT_UID=0
declare -r SUCCESS=0
declare -r ERROR=13
source ./functions.sh

# On vérifie que l'utilisateur soit root
if [ $UID -ne $ROOT_UID ]
then
  echo "Vous devez être root pour lancer ce script !"
  exit $ERROR
fi

# On récupère les utilisateurs humains et leurs répertoires personnels
# humain =  UID supérieur ou égal à 1000 et username différent de "nobody"
usernames=(`awk -F: '$3 >= 1000 && $1 != "nobody" {print $1}' /etc/passwd`)
folders=(`awk -F: '$3 >= 1000 && $1 != "nobody" {print $6}' /etc/passwd`)

# On crée un tableau "human_users" avec des valeurs au format "taille(octets):répertoire:username"
nb_folders=$((${#folders[*]} - 1))
for i in `seq 0 $nb_folders`
do
    human_users+=("`du -sb ${folders[$i]} | cut -f1`:${folders[$i]}:${usernames[$i]}")
done

# On les classe par ordre décroissant de taille de répertoire personnel grâce au tri shaker
swapped=$TRUE
start=0
end=$((${#human_users[*]} - 2)) # -2 car on compare human_users[i] à human_users[i+1]
while [ $swapped -eq $TRUE ]
do
    swapped=$FALSE
    # Aller : recherche de la plus petite valeur
    for i in `seq $start $end`
    do
        # Si human_users[i] < human_users[i+1]
        value=`echo ${human_users[$i]} | cut -d: -f1`
        next_index=$(($i + 1))
        next_value=`echo ${human_users[$next_index]} | cut -d: -f1`
        if [ $value -lt $next_value ]
        then
            # On inverse les positions des valeurs
            temp=${human_users[$i]}
            human_users[$i]=${human_users[$next_index]}
            human_users[$next_index]=$temp
            swapped=$TRUE
        fi
    done
    # La valeur la plus petite est désormais au bout du tableau et n'a pas besoin d'être réévaluée
    # On décrémente la valeur de fin de la boucle pour optimiser le processus
    end=$(($end - 1))


    # Retour : recherche de la plus grande valeur
    for i in `seq $end -1 $start`
    do
        # Si human_users[i] < human_users[i+1]
        value=`echo ${human_users[$i]} | cut -d: -f1`
        next_index=$(($i + 1))
        next_value=`echo ${human_users[$next_index]} | cut -d: -f1`
        if [ $value -lt $next_value ]
        then
            # On inverse les positions des valeurs
            temp=${human_users[$i]}
            human_users[$i]=${human_users[$next_index]}
            human_users[$next_index]=$temp
            swapped=$TRUE
        fi
    done
    # La valeur la plus grande est désormais au début du tableau et n'a pas besoin d'être réévaluée
    # On incrémente la valeur du début de la boucle pour optimiser le processus
    start=$(($start + 1))
done


# On crée un nouveau module du message of the day auquel on donne les droits d'exécution
# On y écrit le résultat qu'on affiche aussi en console
motd_top5="/etc/update-motd.d/99-top5-disk-consumers"
echo "#!/bin/bash" > $motd_top5
chmod +x $motd_top5
echo 'echo -e "\e[36mLes plus gros consommateurs de disque sont :\e[0m"' >> $motd_top5
echo -e "\nLes plus gros consommateurs de disque sont :"

# S'il y plus de 5 utilisateurs, on fait un top 5
# Sinon, on affiche tous les utilisateurs
if [ ${#human_users[*]} -gt 5 ]
then
    end=4
else
    end=$((${#human_users[*]} - 1))
fi

# Pour chacun des plus gros consommateurs
for i in `seq 0 $end`
do
    # On récupère la taille du répertoire personnel en octets
    # grâce à la fonction get_readable_size qui retourne une chaîne de caractères
    folder_size_bytes=`echo ${human_users[$i]} | cut -d: -f1`
    folder_readable_size=`get_readable_size $folder_size_bytes`

    # On écrit sur le fichier du motd ainsi qu'en console
    username=`echo ${human_users[$i]} | cut -d: -f3`
    echo "echo -e '\e[36m- $username -> $folder_readable_size\e[0m'" >> $motd_top5
    echo "- $username -> $folder_readable_size"
done

echo -e "\nLe message d'accueil a été mis à jour !\n"


# On modifie le fichier bashrc de tous les utilisateurs humains
for j in "${human_users[@]}"
do
    # On localise le fichier bashrc
    repository=`echo $j | cut -d: -f2`
    filepath=$repository/.bashrc

    # On vérifie s'il existe déjà une règle sur le fichier
    grep -q "#Alerte : 100 Mo" $filepath
    if [ $? -eq $SUCCESS ]
    then
        # Si oui, on ne fait rien
        continue
    else
        # Sinon, on l'écrit dans le fichier

        # On récupère la taille du répertoire personnel
        rep_size=`du -sb $repository | cut -f1`
        readable_size=`get_readable_size $rep_size`

        # On écrit le commentaire qui identifie la présence de l'alerte
        echo -e "\n#Alerte : 100 Mo" >> $filepath
        # On écrit les variables utiles dynamique : la taille du répertoire est mise à jour à chaque ouverture du bashrc
        echo "repository=$repository" >> $filepath
        echo 'size=`du -sb $repository | cut -f1`' >> $filepath
        # Si la taille du répertoire est supérieure à 104857600 octets (100 Mo), une alerte s'affiche
        echo 'if [ $size -gt 104857600 ]' >> $filepath
        echo "then" >> $filepath
        echo "echo -e '\e[33mAlerte : vous avez dépassé la limite des 100 Mo !\e[0m'" >> $filepath
        echo "echo -e '\e[33mLa taille de votre répertoire est de $readable_size\e[0m'" >> $filepath
        echo "fi" >> $filepath

        username=`echo $j | cut -d: -f3`
        echo "Le fichier bashrc de $username a été mise à jour"
    fi
done

exit $SUCCESS
```

Script 3 : génération d'une liste des exécutables pour lesquels le SUID et/ou le SGID est activé. A chaque lancement de script, comparaison avec la liste précédente si elle existe avec affichage des éventuelles différences.

```bash
#!/bin/bash

# On déclare les constantes
declare -r SUCCESS=0
declare -r ROOT_UID=0
declare -r ERROR=13

# On vérifie que l'utilisateur soit root
if [ $UID -ne $ROOT_UID ]
then
  echo "Vous devez être root pour lancer ce script !"
  exit $ERROR
fi

# On initialise les variables utiles et de mise en forme
list1=/home/suid_guid_exe
list2=/home/list2
zone="/"
format="%i:%a"
bold=$(tput bold)
normal=$(tput sgr0)

# On récupère les exécutables avec une autorisation au moins égale
# à 2000 ou 4000 au format "inode:permissions(octal)"
# On trie la liste, retire les doublons et stocke le résultat dans un fichier list2
find $zone -executable \( -perm -2000 -o -perm -4000 \) -exec stat --format $format {} ';' 2> /dev/null | sort -n -u > $list2

# On vérifie s'il existe déjà un fichier list1 pour comparer les données
if [ -f $list1 ]
then
    # Si le fichier existe : on compare les listes bit à bit
    cmp -s $list1 $list2
    if [ $? -eq $SUCCESS ]
    then
        echo -e "\nLes listes de fichiers sont identiques : aucune modification n'a été réalisée.\n"
    else
        # Si les listes sont différentes : on met les fichiers trouvés dans un tableau
        tab_list2=(`awk '{print $1}' $list2`)

        # On boucle sur tous les fichiers trouvés
        for i in "${tab_list2[@]}"
        do
            # On récupère les informations utiles
            id="`echo $i | cut -d: -f1`"
            current_rights="`echo $i | cut -d: -f2`"

            if grep -q "$i:$current_rights" $list1
            then
                # Si l'inode et les droits sont exactement les mêmes que dans la liste 1
                # Le fichier n'a pas été modifié : on retire la ligne de la liste 1
                sed -i "/^$id/d" $list1
            elif grep -q "^$id:" $list1
            then
                # Si l'inode est trouvé mais que les droits sont différents
                # On récupère le(s) nom(s) de fichier(s) associé(s) à l'inode et la date de dernière modification
                # On stocke le résultat dans une chaîne de caractères
                names=(`find $zone -inum $id`)
                file=`grep "^$id:" $list1`
                previous_rights=`echo $file | cut -d: -f2`
                for name in "${names[@]}"
                do
                    date="`date -r $name`"
                    updated_files="$updated_files$name"
                    updated_files="$updated_files\nDernière modification : $date"
                    updated_files="$updated_files\nDroits précédents : $previous_rights - Droits actuels : $current_rights\n"
                done
                # On retire la ligne de la liste 1
                sed -i "/^$id:/d" $list1
            else
                # Si l'inode n'a pas été trouvé dans la liste 1 : le fichier a gagné un droit SUID ou GUID
                # On récupère le(s) nom(s) de fichier(s) associé(s) à l'inode et la date de dernière modification
                # On stocke le résultat dans une chaîne de caractères
                names=(`find $zone -inum $id`)
                for name in "${names[@]}"
                do
                    date="`date -r $name`"
                    new_files="$new_files$name"
                    new_files="$new_files\nDernière modification : $date"
                    new_files="$new_files\nDroits actuels : $current_rights\n"
                done
            fi
        done

        # Si la liste initiale n'est pas vide : on récupère la liste des fichiers restant
        [ -s $file1 ]
        if [ $? -eq $SUCCESS ]
        then
            # On boucle sur chaque ligne du fichier
            for j in `cat $list1`
            do
                # On récupère l'inode, les droits précédents et le(s) nom(s) de fichier(s) associé(s) à l'inode
                id="`echo $j | cut -d: -f1`"
                previous_rights="`echo $j | cut -d: -f2`"
                names=(`find $zone -inum $id`)
                for name in "${names[@]}"
                do
                    # Pour chaque nom de fichier, on stocke les informations dans une chaîne de caractères
                    date="`date -r $name`"
                    current_rights=`stat $name --format "%a"`
                    deleted_files="$deleted_files$name"
                    deleted_files="$deleted_files\nDernière modification : $date"
                    deleted_files="$deleted_files\nDroits précédents : $previous_rights - Droits actuels : $current_rights\n"
                done
            done
        fi

        # Si la chaîne new_files n'est pas vide : on affiche la liste des fichiers
        [ -n "$new_files" ]
        if [ $? -eq $SUCCESS ]
        then
            echo -e "\e[92m\n${bold}Fichiers qui ont gagné un droit SUID et/ou GUID :${normal}\e[0m"
            echo -e $new_files
        fi

        # Si la chaîne updated_files n'est pas vide : on affiche la liste des fichiers
        [ -n "$updated_files" ]
        if [ $? -eq $SUCCESS ]
        then
            echo -e "\e[93m\n${bold}Fichiers modifiés :${normal}\e[0m"
            echo -e $updated_files
        fi

        # Si la chaîne deleted_files n'est pas vide : on affiche la liste des fichiers
        [ -n "$deleted_files" ]
        if [ $? -eq $SUCCESS ]
        then
            echo -e "\e[91m\n${bold}Fichiers qui ont perdu un droit SUID et/ou GUID :${normal}\e[0m"
            echo -e $deleted_files
        fi

        # Les nouvelles données deviennent la liste de référence
        cp $list2 $list1
        if [ $? -eq $SUCCESS ]
        then
            echo -e "La liste de référence a été mise à jour !\n"
        else
            echo -e "Aïe, problème dans la mise à jour de la liste de référence :(\n"
        fi
    fi
else
    # Si le fichier n'existe pas : on le crée à partir des données récupérées
    echo -e "\nIl n'existe pas de liste permettant de comparer les informations."
    echo "Création de la liste de référence en cours..."
    cp $list2 $list1
    if [ $? -eq $SUCCESS ]
    then
        echo -e "La liste de référence a été créée !\n"
    else
        echo -e "Aïe, problème dans la création de la liste de référence :(\n"
    fi
fi

# On supprime le fichier list2
rm $list2

exit $SUCCESS
```

Script de fonctions : fonction qui récupère une taille en octets en paramètres et renvoie une chaîne de caractères au format « X Go, Y Mo, Z Ko et T octets »

```bash
#!/bin/bash

get_readable_size()
(
    # On récupère la taille à en paramètre
    nb_of_bytes=$1
    units=(" octets" " Ko et " " Mo, " " Go, ")

    # On boucle de 3 à 0 (1024³ = Go, 1024² = Mo...)
    for j in {3..0}
    do
        # Division entière de la taille en octets par 1024^$i
        unit_in_bytes=`echo "1024^$j" | bc`
        nb_of_units=$(($nb_of_bytes / $unit_in_bytes))

        # On concatène la chaîne avec "valeur_entiere unité_en_cours" (+ gestion du singulier)
        if [[ $j == 0 && $nb_of_units < 2 ]]
        then
            units[$j]=" octet"
        fi
        str=$str$nb_of_units${units[$j]};

        # On retire à la taille autant d'octets que la valeur trouvée (en octets)
        bytes_to_remove=$(($nb_of_units * $unit_in_bytes))
        nb_of_bytes=$(($nb_of_bytes - $bytes_to_remove))
    done

    # On renvoie la taille finale lisible
    echo $str
)
```

Configuration de serveurs DNS principal et secondaire

Contexte

Configuration DHCP

- **Adresse réseau** : `192.168.49.0/24`
- **Passerelle** : `192.168.49.1/24`
- **Distribution des adresses** : `192.168.49.128-254/24`
- **Adresse broadcast** : `192.168.49.255/24`

Machines

- **Client Ubuntu Desktop 20.04**
    - **Configuration IP** : DHCP avec DNS manuel
- **Client Ubuntu Server 20.04**
    - **Configuration IP** : statique (`192.168.49.133/24`) avec DNS manuel
    - **Domaine** : `ubuntu.uncle.esgi`
- **Serveur DNS primaire Debian 10**
    - **Configuration IP** : statique (`192.168.49.254/24`) avec DNS en localhost
    - **Domaine** : `ns1.uncle.esgi`
    - **Alias (CNAME)** : `server.uncle.esgi`
- **Serveur DNS secondaire Debian 10**
    - **Configuration IP** : statique (`192.168.49.253/24`) avec DNS sur le serveur principal
    - **Domaine** : `ns2.uncle.esgi`
- **Zone DNS** : `uncle.esgi`
- **Alias externes** :
    - `gogole.uncle.esgi` ⇒ `www.google.com`
    - `school.uncle.esgi` ⇒ `www.esgi.fr`

1. Configuration des serveurs DNS

1.1 Serveur primaire

Installation des ressources

Avant toute chose, nous devons installer les ressources nécessaires (l'option -y permettant de répondre oui par avance aux demandes de confirmation) :

```
apt install bind9 bind9utils bind9-doc dnsutils resolvconf -y
```

Configuration réseau

Notre serveur DNS doit avoir une configuration IP statique afin d'être configuré sur les postes clients. Une fois le nom de l'interface réseau récupéré grâce à la commande `ip -4 -o addr` (qui nous donne ici `ens33`), on peut alors modifier le fichier `/etc/network/interfaces` en se référant au contexte donné :

```
auto ens33
iface ens33 inet static
        address 192.168.49.254
        netmask 255.255.255.0
        gateway 192.168.49.1
        dns-nameservers 127.0.0.1
```

Il faut ensuite redémarrer le service avec la commande suivante :

```
systemctl restart networking
```

Nous vérifions alors que tout s'est bien déroulé en affichant la configuration IP grâce à la commande `ip a` et en vérifiant que le fichier `/etc/resolv.conf` contient bien uniquement la ligne `nameserver 127.0.0.1` :

```
user@dns-server:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:1e:06:45 brd ff:ff:ff:ff:ff:ff
    inet 192.168.49.254/24 brd 192.168.49.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe1e:645/64 scope link
       valid_lft forever preferred_lft forever
```

```
user@dns-server:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.0.1
```

Test du serveur cache

Désormais, notre serveur possède son propre DNS local : nous allons nous assurer qu'il fonctionne correctement et parvient à résoudre le nom de domaine www.debian.org grâce à la requête `dig www.debian.org`.

```
root@dns-server:/home/server dig www.debian.org
; <<>> DiG 9.11.5-P4-5.1+deb10u1-Raspbian <<>> www.debian.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64130
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.debian.org.                        IN      A

;; ANSWER SECTION:
www.debian.org.         168     IN      A       130.89.148.77

;; AUTHORITY SECTION:
www.debian.org.         300     IN      NS      geo1.debian.org
www.debian.org.         300     IN      NS      geo2.debian.org
www.debian.org.         300     IN      NS      geo3.debian.org

;; ADDITIONAL SECTION:
geo1.debian.org.        28799   IN      A       82.195.75.105
geo2.debian.org.        28799   IN      A       209.87.16.31
geo1.debian.org.        28799   IN      AAAA    2001:41b8:202:deb:1a1a:0:52c3:4b69

;; Query time: 568 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Jun 29 18:29:49 BST 2020
;; MSG SIZE  rcvd: 204
```

Le temps de résolution est de 568 ms mais en relançant la même requête, le temps de résolution est nul : l'adresse ayant été résolue et les informations correctement mises en cache, nous pouvons donc mettre en place le serveur primaire.

Mise en place du serveur primaire

Définition des zones DNS et reverse DNS

Nous allons maintenant mettre en place nos zones DNS et reverse DNS selon le contexte défini initialement. Pour cela, nous allons modifier le fichier `/etc/bind/named.conf.local` et y ajouter les lignes suivantes :

```
zone "uncle.esgi" {
     type master;
     file "/etc/bind/db.uncle.esgi";
};

zone "49.168.192.in-addr.arpa" {
     type master;
     file "/etc/bind/db.49.168.192.in-addr.arpa";
};
```

Une fois chose faite, il faut alors créer les fichiers de zone `/etc/bind/db.uncle.esgi` (pour la zone DNS) et `/etc/bind/db.49.168.192.in-addr.arpa` (pour la zone reverse DNS).

Configuration de la zone DNS et reverse DNS

Remplissons d'abord le fichier `/etc/bind/db.uncle.esgi` contenant la définition de notre zone DNS ainsi que les domaines, les alias, etc. avec les informations suivantes :

```
$TTL 10800
$ORIGIN uncle.esgi.
@       IN SOA ns1.uncle.esgi. admin.uncle.esgi. (
        20200628; Serial
        3h; Refresh
        1h; Retry
        1w; Expire
        1h; Minimum
)
@       IN NS   ns1.uncle.esgi.
ubuntu  IN A    192.168.49.133
ns1     IN A    192.168.49.254
ns2     IN A    192.168.49.253
primary-server  IN CNAME        ns1
gogole          IN CNAME        www.google.com.
school          IN CNAME        www.esgi.fr.
```

Contenu de l'entête SOA (Start of Authority) :

- **@** : remplace le nom de la zone uncle.esgi
- **ns1.uncle.esgi.** : serveur primaire de la zone
- **admin.uncle.esgi.** : adresse e-mail de l'administrateur (l'@ est remplacée par un .)
- **Serial** : indique le numéro de version du serveur : il doit être incrémenté à chaque modification, ce qui indique aux autres serveurs que leurs données doivent être mises à jour.
- **Refresh** : indique le temps entre deux mises à jour des serveurs secondaires par rapport au serveur primaire.
- **Retry** : indique le temps entre 2 tentatives de Refresh s'il y a échec.
- **Expire** : indique le temps au bout duquel un serveur secondaire ne fait plus autorité s'il n'a pas réussi à contacter le serveur primaire.
- **Minimum** : indique le temps durant lequel une réponse négative doit être conservée en cache.

On définit après l'entête SOA le nom du serveur de la zone avec un enregistrement NS, puis on ajoute également les correspondances de nom de machine / adresses IP grâce aux enregistrements A ainsi que trois alias.

Il faut ensuite configurer la zone reverse DNS à l'aide du fichier `/etc/bind/db.49.168.192.in-addr.arpa` :

```
$TTL 10800
$ORIGIN 49.168.192.in-addr.arpa.
@       IN SOA ns1.uncle.esgi. admin.uncle.esgi (
        20200628;
        3h;
        1h;
        1w;
        1h);
@       IN NS   ns1.uncle.esgi.
133     IN PTR ubuntu.uncle.esgi.
254     IN PTR ns1.uncle.esgi.
253     IN PTR ns2.uncle.esgi.
```

De la même manière que pour la zone DNS, un enregistrement NS définit le nom du serveur DNS de la zone puis est suivi des enregistrements PTR permettant de faire la résolution inverse des adresses IP.

Une chose faite, il faut alors redémarrer le service bind9 avec la commande `systemctl restart bind9`

Test de la zone DNS

Si tout s'est bien déroulé, les zones DNS et reverse DNS devraient être reconnues par notre serveur. Pour s'en assurer, on va tout d'abord résoudre l'adresse `ubuntu.uncle.esgi` avec une requête `dig` :

```
user@dns-server:~$ dig ubuntu.uncle.esgi

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> ubuntu.uncle.esgi
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37398
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 16fab81d8a5925ff15021fbe5efa5d6242c49ab1ff5ea44b (good)
;; QUESTION SECTION:
;ubuntu.uncle.esgi.             IN      A

;; ANSWER SECTION:
ubuntu.uncle.esgi.      10800   IN      A       192.168.49.133

;; AUTHORITY SECTION:
uncle.esgi.             10800   IN      NS      ns1.uncle.esgi.

;; ADDITIONAL SECTION:
ns1.uncle.esgi.         10800   IN      A       192.168.49.254

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: lun. juin 29 23:30:10 CEST 2020
;; MSG SIZE  rcvd: 158
```

Le domaine est reconnu ! Vérifions alors la résolution inverse de ce domaine avec la requête `dig -x 192.168.49.133 +short` :

```
user@dns-server:~$ dig -x 192.168.49.133 +short
ubuntu.uncle.esgi.
```

On constate alors que la zone de reverse DNS fonctionne également.

1.2 Serveur secondaire

Mettons maintenant en place le serveur DNS secondaire qui assurera le relais si le serveur DNS principal tombe en panne. On utilisera une relation "maitre - esclave" entre les serveurs primaire et secondaire : le serveur secondaire sera la copie confirme du premier et ne pourra se mettre à jour que si le principal est actif.

Installation des ressources

Il faut tout d'abord installer les mêmes outils que sur le premier serveur avec la commande suivante :

```
apt install bind9 bind9utils bind9-doc dnsutils resolvconf -y
```

Configuration réseau

De même que pour le serveur principal, le serveur secondaire doit avoir une configuration IP statique afin d'être configuré sur les postes clients. C'est encore une fois le fichier `/etc/network/interfaces` qui sera modifié en se référant au contexte donné et au retour de la commande `ip -4 -o addr` :

```
auto ens33
iface ens33 inet static
        address 192.168.49.253
        netmask 255.255.255.0
        gateway 192.168.49.1
        dns-nameservers 192.168.49.254
```

Il faut ensuite redémarrer le service avec la commande :

```
systemctl restart networking
```

Nous vérifions alors que tout s'est bien déroulé en affichant la configuration IP grâce à la commande `ip a` et en vérifiant que le fichier `/etc/resolv.conf` contient bien uniquement la ligne `nameserver 127.0.0.1` :

```
user@dns-server-2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:36:81:c4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.49.253/24 brd 192.168.49.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe36:81c4/64 scope link
       valid_lft forever preferred_lft forever
user@dns-server:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.49.254
search uncle.esgi
```

Mise en place du serveur secondaire

Déclaration du serveur secondaire sur le serveur primaire

Il faut ensuite configurer le serveur principal pour qu'il interagisse avec le serveur secondaire. Tout d'abord, il faut lui dire de notifier le serveur secondaire lorsqu'un des fichiers de zone est modifié : pour ce faire, on ajoute la directive `notify yes` pour chaque zone qu'on souhaite reproduire. On ajoute ensuite la liste des serveurs secondaires autorisés à effectuer un transfert de zone avec la directive `allow-transfer` :

Serveur DNS principal : fichier `/etc/bind/named.conf.local`

```
zone "uncle.esgi" {
     type master;
     file "/etc/bind/db.uncle.esgi";
     notity yes;
     allow-transfer { 192.168.49.253; };
};

zone "49.168.192.in-addr.arpa" {
     type master;
     file "/etc/bind/db.49.168.192.in-addr.arpa";
     notity yes;
     allow-transfer { 192.168.49.253; };
};
```

Il faut également déclarer le serveur secondaire `ns2` dans les zones DNS et reverse DNS :

Serveur DNS principal : zone DNS `/etc/bind/db.uncle.esgi`

```
$TTL 10800
$ORIGIN uncle.esgi.
@       IN SOA ns1.uncle.esgi. admin.uncle.esgi. (
        20200628;
        3h;
        1h;
        1w;
        1h);
@       IN NS   ns1.uncle.esgi.
@       IN NS   ns2.uncle.esgi.
ubuntu  IN A    192.168.49.133
ns1     IN A    192.168.49.254
ns2     IN A    192.168.49.253
primary-server  IN CNAME        ns1
gogole          IN CNAME        www.google.com.
school          IN CNAME        www.esgi.fr.
```

Serveur DNS principal : zone reverse DNS `/etc/bind/db.49.168.192.in-addr.arpa`

```
$TTL 10800
$ORIGIN 49.168.192.in-addr.arpa.
@       IN SOA ns1.uncle.esgi. admin.uncle.esgi (
        20200628;
        3h;
        1h;
        1w;
        1h);
@       IN NS   ns1.uncle.esgi.
@       IN NS   ns2.uncle.esgi.
133     IN PTR ubuntu.uncle.esgi.
254     IN PTR ns1.uncle.esgi.
253     IN PTR ns2.uncle.esgi.
```

Une fois chose faite, il faut redémarrer le service bind9 avec la commande `systemctl restart bind9`.

Si tout s'est bien déroulé, le serveur secondaire devrait être reconnu par les zones DNS et reverse DNS. Pour s'en assurer, on va tout d'abord résoudre l'adresse `ns2.uncle.esgi` avec la une requête `dig` :

```
dig ns2.uncle.esgi

; <<>> DiG 9.11.5-P4-5.1+deb10u1-Debian <<>> ns2.uncle.esgi
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22898
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 954560d971df8139d8d9adc55efa63fd94049bcab5a7c36f (good)
;; QUESTION SECTION:
;ns2.uncle.esgi.                        IN      A

;; ANSWER SECTION:
ns2.uncle.esgi.         10800   IN      A       192.168.49.253

;; AUTHORITY SECTION:
uncle.esgi.             10800   IN      NS      ns2.uncle.esgi.
uncle.esgi.             10800   IN      NS      ns1.uncle.esgi.

;; ADDITIONAL SECTION:
ns1.uncle.esgi.         10800   IN      A       192.168.49.254

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: lun. juin 29 23:58:21 CEST 2020
;; MSG SIZE  rcvd: 135
```

Déclaration du serveur secondaire

Sur le serveur secondaire, il faut alors modifier le fichier `/etc/bind/named.conf.local` afin de le déclarer comme serveur "esclave".

```
zone "uncle.esgi" {
     type slave;
     file "/var/lib/bind/db.uncle.esgi";
     notity yes;
     masters { 192.168.49.254; };
};

zone "49.168.192.in-addr.arpa" {
     type master;
     file "/var/lib/bind/db.49.168.192.in-addr.arpa";
     notity yes;
     masters { 192.168.49.254; };
};
```

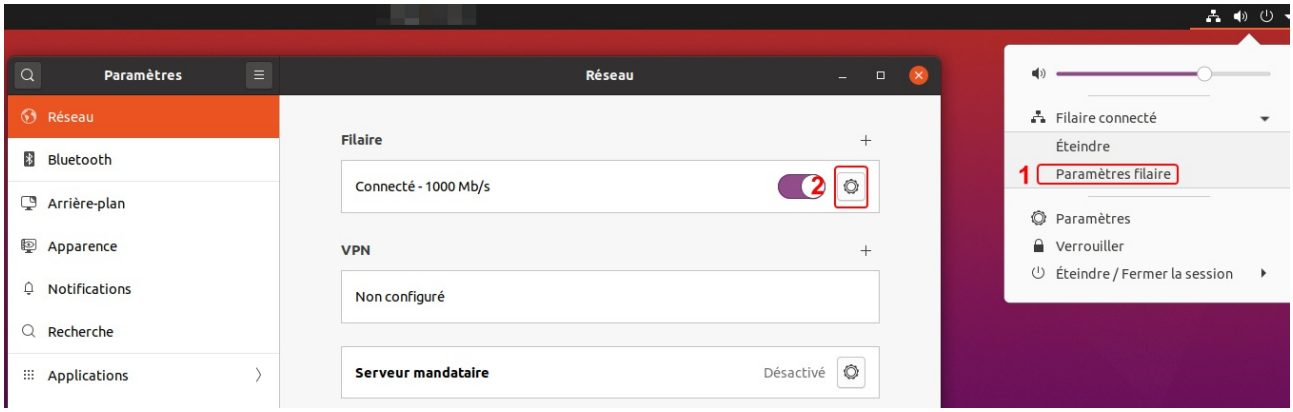
Pour avoir les droits nécessaires, les fichiers téléchargés sur le serveur secondaire depuis le serveur principal seront stockés dans le repertoire `/var/lib/bind/`.

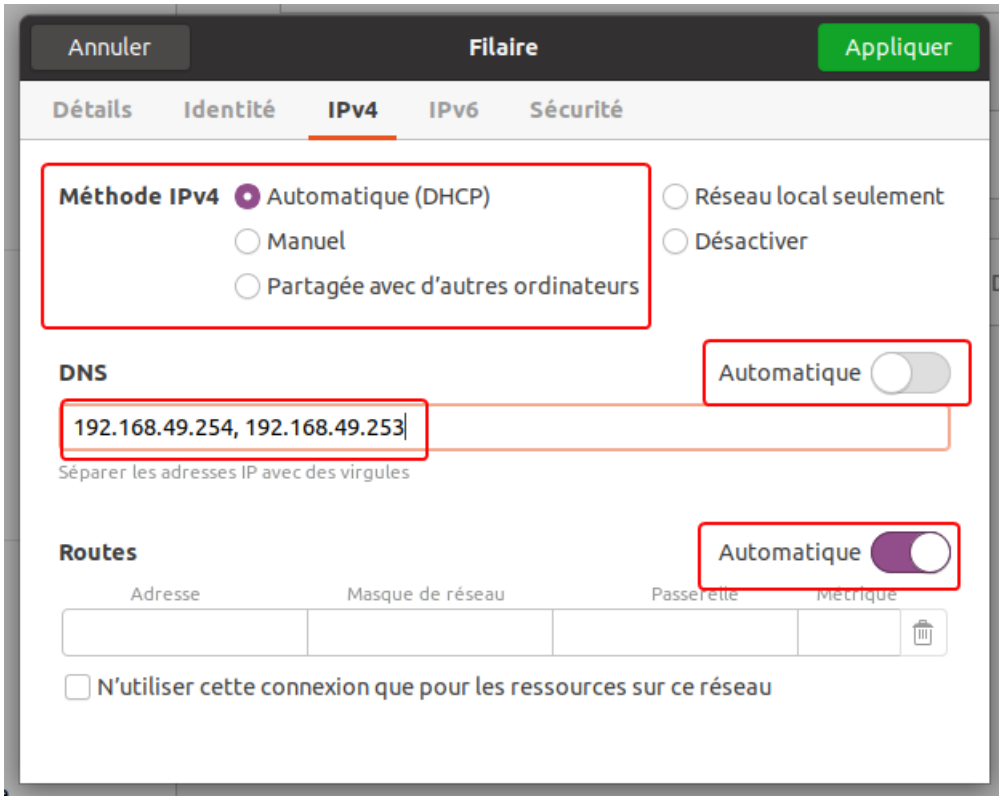
1.3 Configuration du client Ubuntu Desktop 20.04

Comme indiqué dans le Contexte, la configuration IP du client Ubuntu Desktop est obtenue par DHCP et le DNS est configuré manuellement.

Configuration réseau

Pour ce faire, il faut se rendre dans les paramètres réseau de la machine et ajouter les informations suivantes :





Test du serveur DNS

Pour tester les zones DNS et reverse DNS, on utilise une requête `dig` et l'alias "primary-server" du serveur principal :

```
dig primary-server.uncle.esgi

; <<>> DiG 9.16.1-Ubuntu <<>> primary-server.uncle.esgi
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27593
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;primary-server.uncle.esgi. IN A

;; ANSWER SECTION:
primary-server.uncle.esgi. 7092 IN CNAME ns1.uncle.esgi.
ns1.uncle.esgi.  7092 IN A 192.168.49.254

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: lun. juin 29 15:19:09 PDT 2020
;; MSG SIZE  rcvd: 88
```

La zone DNS étant correcte, on vérifie alors la zone reverse DNS :

```
user@dns-server:~$ dig -x 192.168.49.133 +short
ubuntu.uncle.esgi.
```

1.4 Configuration du client Ubuntu server 20.04

Comme indiqué dans le Contexte, la configuration IP du client Ubuntu Server est statique (`192.168.49.133/24`) et le DNS est configuré manuellement.

Configuration réseau

Pour ce faire, nous allons utiliser l'utilitaire réseau installé par défaut sur Ubuntu Server : netplan. Pour cela, il faut modifier le fichier `/etc/netplan/00-installer-config.yaml` de la façon suivante :

```
network:
  ethernets:
    ens33:
      addresses: [192.168.49.133/24]
      gateway4: 192.168.49.1
      nameservers:
        addresses: [192.168.49.254,192.168.49.253]
  version: 2
```

Pour appliquer cette configuration, il faudra lancer la commande `netplan apply`. Si aucune erreur n'a été détectée, l'adresse IP sera alors correctement définie :

```
user@dnsclient:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:09:70:42 brd ff:ff:ff:ff:ff:ff
    inet 192.168.49.133/24 brd 192.168.49.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe09:7042/64 scope link
       valid_lft forever preferred_lft forever
```

Test du serveur DNS

Nous testons alors la zone DNS grâce à la requête `dig` :

```
user@dnsclient:~$ dig ns2.uncle.esgi

; <<>> DiG 9.16.1-Ubuntu <<>> ns2.uncle.esgi
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33076
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;ns2.uncle.esgi.                        IN      A

;; ANSWER SECTION:
ns2.uncle.esgi.         10800   IN      A       192.168.49.253

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Mon Jun 29 22:29:55 UTC 2020
;; MSG SIZE  rcvd: 59
```

Puis la zone reverse DNS :

```
user@dnsclient:~$ dig -x 192.168.49.133 +short
ubuntu.uncle.esgi.
```

2. Test de panne du serveur DNS primaire

Lorsque les deux serveurs DNS sont actifs, le statut de résolution DNS sur le client **Ubuntu Server 20.04** affiche les informations suivantes :

```
systemd-resolve --status
Global
       LLMNR setting: no
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
          DNSSEC NTA: 10.in-addr.arpa
                      ...
                      corp
                      d.f.ip6.arpa
                      home
                      internal
                      intranet
                      lan
                      local
                      private
                      test

Link 2 (ens33)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 192.168.49.254
         DNS Servers: 192.168.49.254
                      192.168.49.253
```

Nous constatons que le **Current DNS server** est notre serveur DNS principal.

Lorsqu'on éteint le serveur principal, on constate alors que le **Current DNS server** est désormais notre serveur DNS secondaire :

```
systemd-resolve --status
Global
       LLMNR setting: no
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
          DNSSEC NTA: 10.in-addr.arpa
                      ...
                      corp
                      d.f.ip6.arpa
                      home
                      internal
                      intranet
                      lan
                      local
                      private
                      test

Link 2 (ens33)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 192.168.49.253
         DNS Servers: 192.168.49.254
                      192.168.49.253
```

On vérifie cela grâce à une requête `dig` puis un `ping` vers le domaine `ns2.uncle.esgi` :

```
user@dnsclient:~$ dig www.google.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56632
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         243     IN      A       216.58.213.132

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Mon Jun 29 22:39:05 UTC 2020
;; MSG SIZE  rcvd: 59
```

```
user@dnsclient:~$ ping ns2.uncle.esgi
PING ns2.uncle.esgi (192.168.49.253) 56(84) bytes of data.
64 bytes from 192.168.49.253 (192.168.49.253): icmp_seq=1 ttl=64 time=0.088 ms
64 bytes from 192.168.49.253 (192.168.49.253): icmp_seq=2 ttl=64 time=0.173 ms
64 bytes from 192.168.49.253 (192.168.49.253): icmp_seq=3 ttl=64 time=0.151 ms
64 bytes from 192.168.49.253 (192.168.49.253): icmp_seq=4 ttl=64 time=0.161 ms
--- ns2.uncle.esgi ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3035ms
rtt min/avg/max/mdev = 0.088/0.143/0.173/0.032 ms
```