Rapport de projet **Carnofluxe**

La société Carnofluxe nous a contacté afin de produire un site de supervision accessible en intranet avec tous les scripts et plans de sauvegardes allant avec, ainsi qu'un serveur DHCP pour distribuer automatiquement des adresses IPs sur le réseau. Ils souhaitent à terme héberger un site de e-commerce, et nous devons donc en monter l'infrastructure. La réputation de l'école en sera entachée si nous échouons, il nous faut donc réussir cela. Les objectifs étaient assez nombreux. Il nous fallait créer (et le tout sous Linux) un serveur DHCP, un serveur DNS, les scripts permettant de récupérer différentes informations nécessaires à la supervision, un serveur WEB et enfin un plan de sauvegarde efficace.

Au niveau des contraintes, on devait donc tout faire sur Linux, comme dit précédemment. On doit aussi faire en sorte qu'un client Windows 10 fonctionne avec notre infrastructure, notamment le DHCP. On devait aussi posséder un DNS esclave avec un DNS maître, afin de que l'esclave puisse prendre le relai si jamais le maître venait à connaître des soucis.

Pour commencer, nous avions des déviances pour quelle distribution de linux nous devions adopter. Nous avons dû nous mettre d'accord afin d'éviter tout conflit éventuel, et nous avons choisis d'adopter debian.

DHCP et DNS

Procédure d'installation et de configurations des serveurs DNS

Configuration du serveur DNS maitre :

Pour configurer ce serveur, nous avons utilisé un paquet qui s'appelle « bind9 » et pour le télécharger, nous avons utiliser la commande :

```
root@debian:/home/benjamin# apt-get install bind9
```

Une fois que le paquet « bind9 » est installer, nous avons donné au futur serveur DNS une adresse IP statique et pour cela nous sommes allés dans le fichier /etc/network/interfaces et taper plusieurs commandes permettant d'avoir une adresse IP statique :

```
  GNU nano 2.7.4          Fichier : /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 ens33
iface lo inet loopback

auto ens33
iface ens33 inet static
address 192.168.10.5
netmask 255.255.255.0

dns-nameservers 192.168.10.4 192.168.10.5
dns-search carnofluxe.domain
```

Maintenant que la machine possède une adresse IP statique (192.168.10.5) et un masque de sous réseau (255.255.255.0), la configuration du serveurs DNS peut commencer.

Pour commencer, nous sommes allés dans le fichier /etc/bind/named.conf.local pour définir les zones qui vont être utilisées. Une zone directe et une zone réverse. Pour la zone directe, nous avons entrés ces lignes dans le fichier :

```
zone "carnofluxe.domain" {
                type master;
                allow-transfer { 192.168.10.4;};
                file "/etc/bind/db.carnofluxe.domain";
        };
```

Une fois que cela est fait, la zone directe est configurée mais il reste encore la zone réverse. Cette zone est quasiment identique à la précédentes mais les noms de la zone et du fichier de configuration de la zone réverse ne sont pas les mêmes :

```
zone "10.168.192.in-addr.arpa" {
                type master;
                allow-transfer { 192.168.10.4;};
                file "/etc/bind/db.carnofluxe.domain.rev";
        };
```

Le nom de chaque zones est entre guillemets (carnofluxe.domain et 10.168.192.in-addr.arpa), le type de zone et « master » car elle ce trouve sur le DNS maitre et non sur le DNS esclave, « allow-transfer » permet détermine l'adresse IP du serveur DNS esclave (192.168.10.4) et « file » donne la position exacte de chaque fichier de configuration DNS.

Une fois que ce fichier a été configuré, nous avons créer 2 nouveaux fichiers, le fichier de configuration du DNS direct (db.carnofluxe.domain) et le fichier de configuration du DNS réverse (db.carnofluxe.domain.rev).

Dans le fichier db.carnofluxe.domain ce trouve la configuration du DNS direct :

```
 GNU nano 2.7.4       Fichier : /etc/bind/db.carnofluxe.domain

;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns1.carnofluxe.domain. admin.ns1.carnofluxe.domain. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@               IN      NS      ns1.carnofluxe.domain.
@               IN      NS      ns2.carnofluxe.domain.
ns1             IN      A       192.168.10.5
ns2             IN      A       192.168.10.4
carnofluxe.com  IN      A       192.168.10.10
supervision.com IN      A       192.168.10.10
```

$TTL permet de déterminer la durée pendant laquelle les informations concernant le serveur DNS lorsqu'un utilisateur l'interroge seront conservées. La durée actuelle de conservation est d'une semaine et quand ce délai sera écoulé, une nouvelle demande devra être envoyée.

La ligne d'en dessous représente un enregistrement. C'est un enregistrement de type SOA (Start Of Authority) et il est suivi de plusieurs informations. D'abord, le nom du DNS maitre, l'adresse mail de l'administrateur du domaine et à la fin de la ligne, une ouverture de parenthèses pour différents paramètres et différentes valeurs.

- « Serial » permet de donner une durée entre chaque mise à jour de zone et permet d'envoyer toutes les modifications effectuées depuis la dernière mise à jour aux serveur DNS esclave. Ici, la durée est de 2 secondes.

- « Refrech » permet de donner une durée durant laquelle le serveur DNS slave stocke les enregistrements DNS sur son serveur. Ici, la durée est d'une semaine.
- « Retry » permet de donner une durée entre chaque tentative de contact du serveur DNS esclave avec le serveur DNS maitre lorsque ce dernier n'est pas joignable. Ici, la durée est de 1 jour.
- « Expire » permet de donner une limite de temps durant laquelle le serveur DNS esclave tentera de joindre le serveur DNS maitre. Ici, la durée est de 28 jours.
- « Negative » permet de donner une durée durant laquelle le serveur DNS maitre peut garder en mémoire cache les enregistrements. Ici, la durée est d'une semaine. Cette ligne est liée au TTL.

La suite de fichier sont les différents enregistrements. Pour chaque enregistrement, il y a 4 parties : l'hôte du domaine (@, ns1 et s2), la classe (IN = Internet), le type d'enregistrement (A et NS) et enfin la valeur de l'enregistrement (adresses IP et noms d'hôtes).

Nous avons ensuite configuré le DNS réverse dans le fichier db.carnofluxe.domain.rev :

```
  GNU nano 2.7.4     Fichier : /etc/bind/db.carnofluxe.domain.rev

;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA    ns1.carnofluxe.domain. admin.ns1.carnofluxe.domain. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL

@               IN      NS      ns1.carnofluxe.domain.
@               IN      NS      ns2.carnofluxe.domain.
5               IN      PTR     ns1.carnofluxe.domain.
4               IN      PTR     ns2.carnofluxe.domain.
```

Ce fichier est le même que le fichier précédant sauf en ce qui concerne la dernière partie. Les types d'enregistrements sont différents mais ils sont juste à l'envers du précédent fichier. Ns1 et ns2 ont été remplacés par le dernier chiffre de chaque adresses IP (5 et 4), les types d'enregistrements « A » ont été remplacé par PTR et toutes les valeurs d'enregistrements sont maintenant des noms d'hôtes.

Une fois que ces fichiers sont configurés, il ne reste plus qu'à configurer le fichier /etc/hosts :

```
 GNU nano 2.7.4                  Fichier : /etc/hosts

127.0.0.1       localhost
127.0.1.1       ns1.carnofluxe.domain
192.168.10.5    ns1.carnofluxe.domain

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Ce fichier permet d'identifier les différents hôtes présent sur la machine virtuel.

Configuration du serveur DNS esclave :

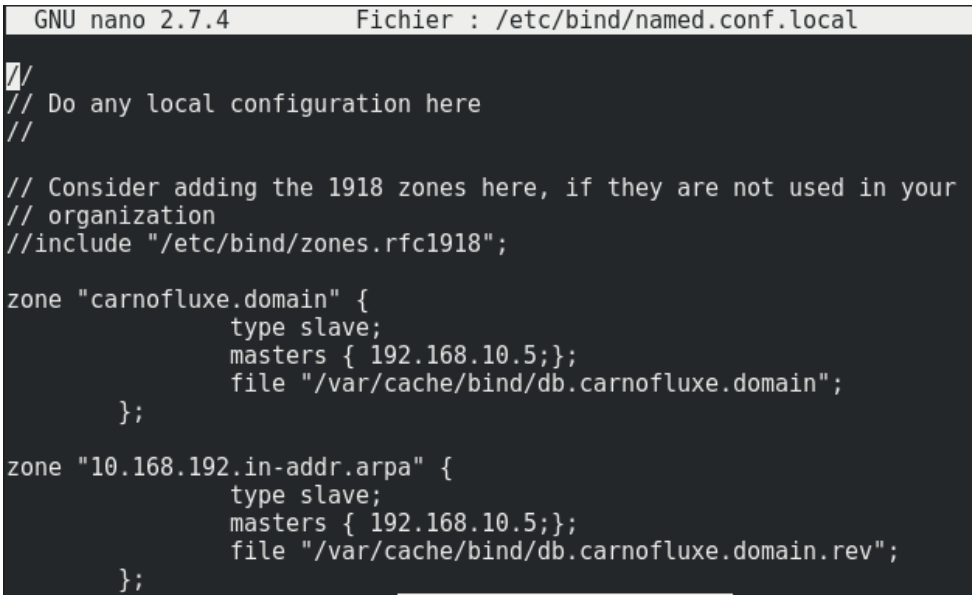
Comme pour le serveur principal, nous avons installé le paquet « bind9 » avec la commande :

```
root@debian:/home/benjamin# apt-get install bind9
```

Et comme pour le serveur principal, nous avons configuré une adresses IP statique dans le fichier /etc/nework/interfaces.

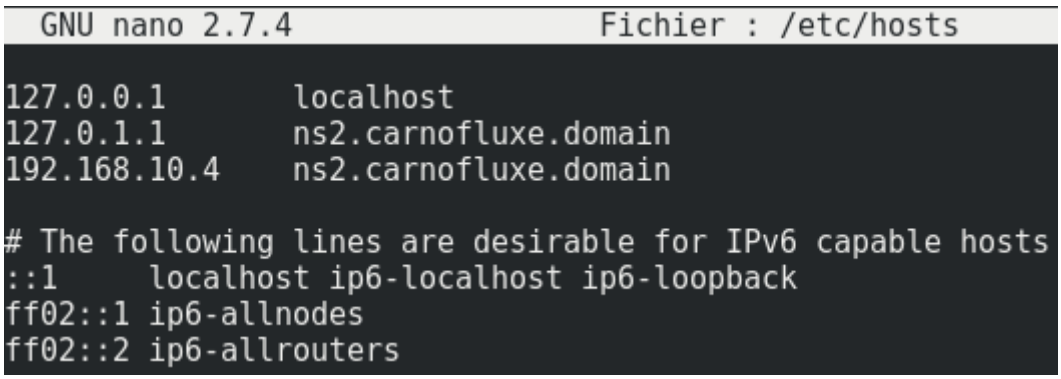
```
 GNU nano 2.7.4           Fichier : /etc/network/interfaces

# 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 ens33
iface lo inet loopback

auto ens33
iface ens33 inet static
address 192.168.10.4
netmask 255.255.255.0

dns-nameservers 192.168.10.4 192.168.10.5
dns-search carnofluxe.domain
```

Pour finir, nous avons configuré, dans le fichier /etc/bind/named.conf.local, les différentes zones utilisées. Une zone directe et une zone réverse.



Le fichier est le même que celui qui est sur le serveur DNS maitre, les seuls choses qui change sont : le type (slave = serveur esclave) et l'adresse IP qui est celle du serveur DNS maitre (192.168.10.5).

Comme pour le DNS maitre, nous avons configuré le fichier /etc/hosts pour indiquer les différents hôtes présents sur la machine virtuelle :



Il ne reste plus qu'à rallumer le service bind9 sur le serveur DNS maitre et le serveur DNS esclave avec la commande « /etc/init.d/bind9 restart » et les 2 serveurs DNS sont prêt à être utilisés.

Procédures d'installation et de configurations des serveurs DHCP

Pour configurer un serveur DHCP, nous avons d'abord télécharger le paquet « isc-dhcp-server » avec la commande :

```
root@debian:/home/benjamin# apt-get install isc-dhcp-server
```

Puis nous avons configuré une adresse IP statique pour le serveur dans le fichier /etc/network/interfaces :

```
  GNU nano 2.7.4          Fichier : /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 ens33
iface lo inet loopback

auto ens33
iface ens33 inet static
address 192.168.10.5
netmask 255.255.255.0

dns-nameservers 192.168.10.4 192.168.10.5
dns-search carnofluxe.domain
```

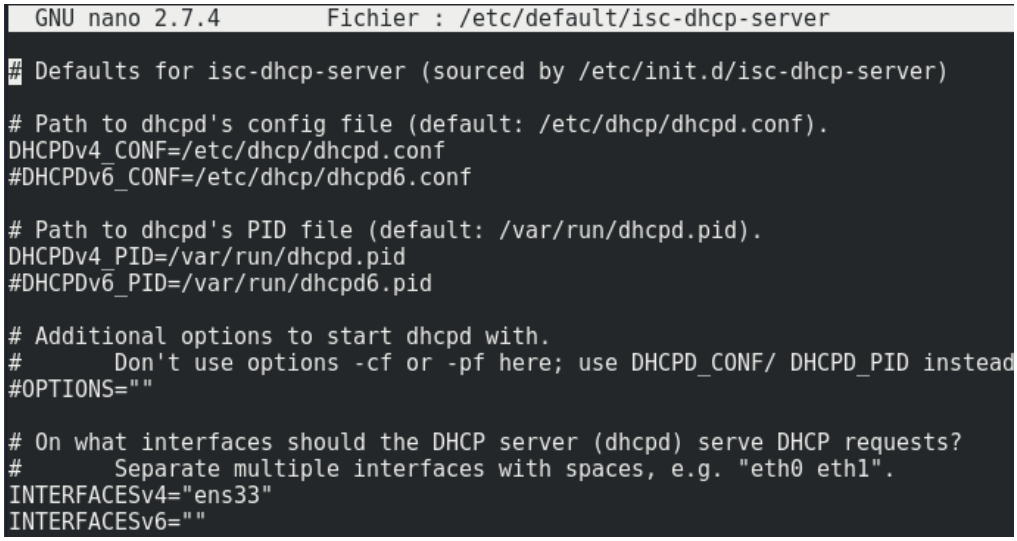
Après avoir fait cela, nous sommes ensuite aller dans les fichiers de configuration du dhcp. D'abord dans le fichier /etc/dhcp/dhcpd.conf pour configurer le DHCP :

```
  GNU nano 2.7.4            Fichier : /etc/dhcp/dhcpd.conf

# which we don't really recommend.

#subnet 10.254.239.32 netmask 255.255.255.224 {
#  range dynamic-bootp 10.254.239.40 10.254.239.60;
#  option broadcast-address 10.254.239.31;
#  option routers rtr-239-32-1.example.org;
#}

# A slightly different configuration for an internal subnet.
subnet 192.168.10.0 netmask 255.255.255.0 {
  range 192.168.10.100 192.168.10.200;
  option domain-name-servers 192.168.10.5, 192.168.10.4;
  option domain-name "carnofluxe.domain";
  option routers 192.168.10.254;
  option broadcast-address 192.168.10.255;
  default-lease-time 600;
  max-lease-time 7200;
}
```

Pour configurer correctement se fichier nous avons décommenter la ligne « authoritative » et un gros bloque de commande contenant la plage d'adresses IP

disponible, les adresses IP des serveurs DNS maitre et esclave, le nom de domaine, l'adresse IP du router, l'adresse IP de broadcast et l'adresse de sous réseau ainsi que le masque de sous-réseau.

Ensuite nous sommes allés dans le fichier /etc/default/isc-dhcp-server pour configurer la bonne interface réseau sur « ens33 »



Il ne reste plus qu'à rallumer le service avec la commande « /etc/init.d/isc-dhcp-server restart » et le serveur DHCP est prêt à délivrer des adresses IP.

Projet Système
Plan de sauvegarde

Pour plus de détails sur les scripts voir le document description des scripts

2- Plan de sauvegarde :

Il vous est demandé de mettre en place un plan de sauvegarde pour le serveur HTTP. Dans ce cadre, il faudra notamment proposer une stratégie de sauvegarde des sites et la mettre en pratique en définissant : Les fichiers à sauvegarder.

La fréquence de sauvegarde en fonction du type de fichier et de la fréquence des modifications pour ne pas perdre plus d'une journée de travail.

Le type de sauvegarde.

L'espace disque nécessaire sachant que l'on veut pouvoir revenir sur les 6 derniers mois de sauvegardes au cas où l'on détecte un problème tardivement. Les fichiers de sauvegarde de plus de 6 mois ne seront donc pas conservés.

Le tout sur un volume physique différent (une réplication via le réseau est prévue dans un autre lot) En cas de problème de sauvegarde, l'administrateur doit pouvoir être alerté.

Plan de sauvegarde	Un plan de sauvegarde du site WEB est proposé. La fréquence de sauvegarde est cohérente. La volumétrie est correctement estimée pour 6 mois. La sauvegarde se fait sur un autre volume physique. La tâche de sauvegarde est automatisée. L'administrateur reçoit une alerte en cas de problème de sauvegarde.

1/Quel type de sauvegarde utiliser ?

Stratégie de sauvegarde possible

- **Ordinateur + disque dur externe :**
  Un ordinateur unique avec sauvegarde automatisée de l'ensemble des données.
- **disque réseau :**
  Ordinateur(s) avec sauvegarde automatisée de l'ensemble des données et accès à distance aux fichiers.

(dans notre cas on optera pour un disque dur externe)

Sauvegarde incrémentielle

Contrairement aux sauvegardes complètes, les sauvegardes incrémentielles vérifient si le moment où le fichier a été modifié est postérieur au moment où ce dernier a été sauvegardé pour la dernière fois.

Si ce n'est pas le cas, le fichier n'a pas été modifié depuis la dernière sauvegarde et ne sera donc pas pris en compte lors de la sauvegarde actuelle.

En revanche, si la date de modification *est* postérieure à la date à laquelle la sauvegarde a été effectuée, le fichier a été modifié et devra donc être sauvegardé.

Les sauvegardes incrémentielles sont utilisées de concert avec une sauvegarde complète se déroulant régulièrement (par exemple, une sauvegarde hebdomadaire, avec des incréments quotidiens).

L'avantage essentiel de l'utilisation de sauvegardes incrémentielles est que l'opération est plus rapide que la création de sauvegardes complètes.

Toutefois, l'inconvénient majeur des sauvegardes incrémentielles est que la restauration de tout fichier donné nécessitera peut-être des recherches dans une ou plusieurs sauvegarde(s) incrémentielle(s) jusqu'à ce que le fichier en question puisse être localisé.

Lors de la restauration d'un système de fichiers complet, il est nécessaire de restaurer la dernière sauvegarde complète et toute sauvegarde incrémentielle postérieure.

1 sauvegarde par jours pendant 6 mois= 183 sauvegardes au maximum.

Le mieux pour sauvegarder serait de faire une sauvegarde complète, puis chaque jours des sauvegardes incrémentielles , au bout d’une semaine on ferait une autre sauvegarde complète avant de supprimer les données ayant plus de 6 mois

Nous allons sauvegarder sur un disque physique (disque dur externe) de 1To les fichiers via un dossier partagé entre la machine virtuelle est Windows

Pourquoi 1To ?Prenons le cas ***extrême***(**improbable**)où chaque incrémentielle devra sauvegarder le maximum, ma vm pesant 6,3Go on

fait 6,3*7*4*6=1,058To on n’atteindra jamais ce maximum car il est absurde de modifier l’intégralité des fichiers et après 6 mois les données sont effacées donc 1To est suffisant.

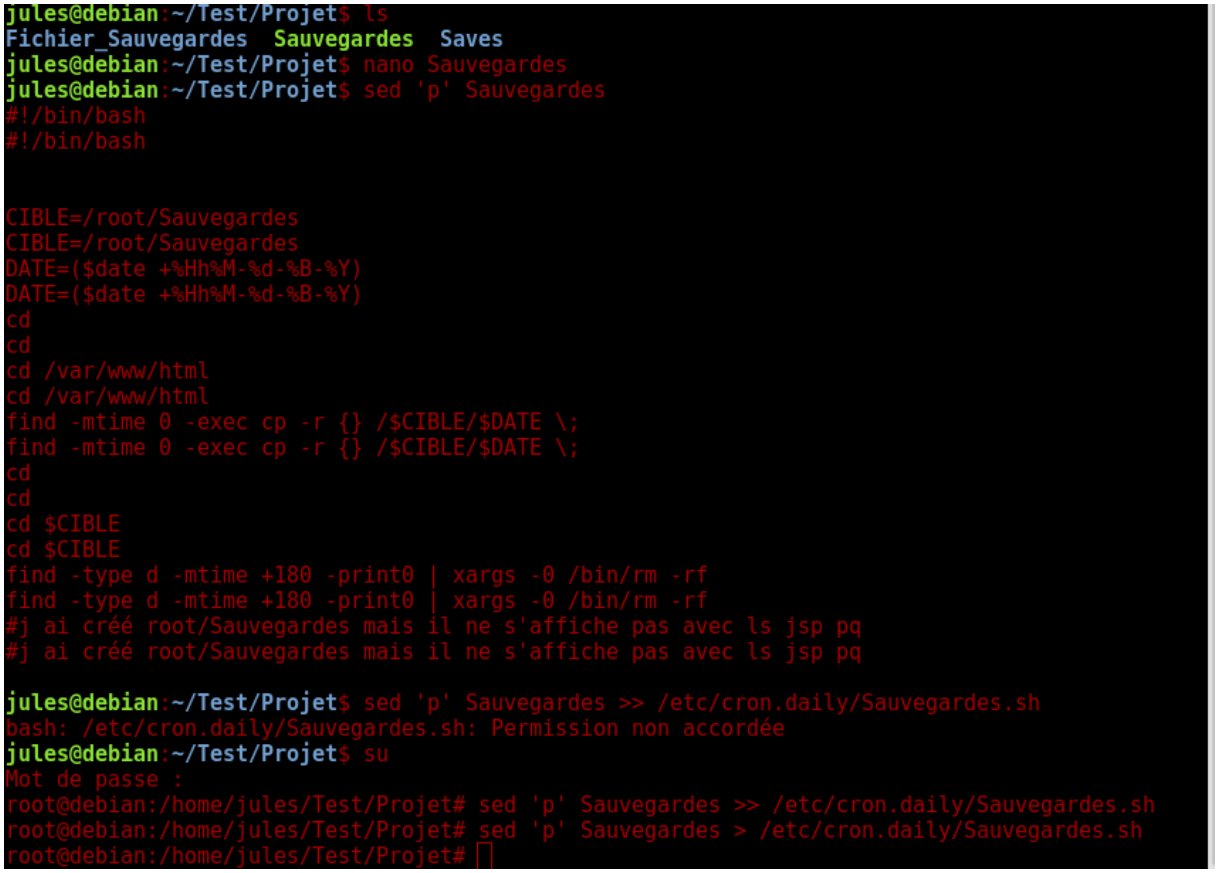
Les Outils de sauvegarde

En ligne de commande

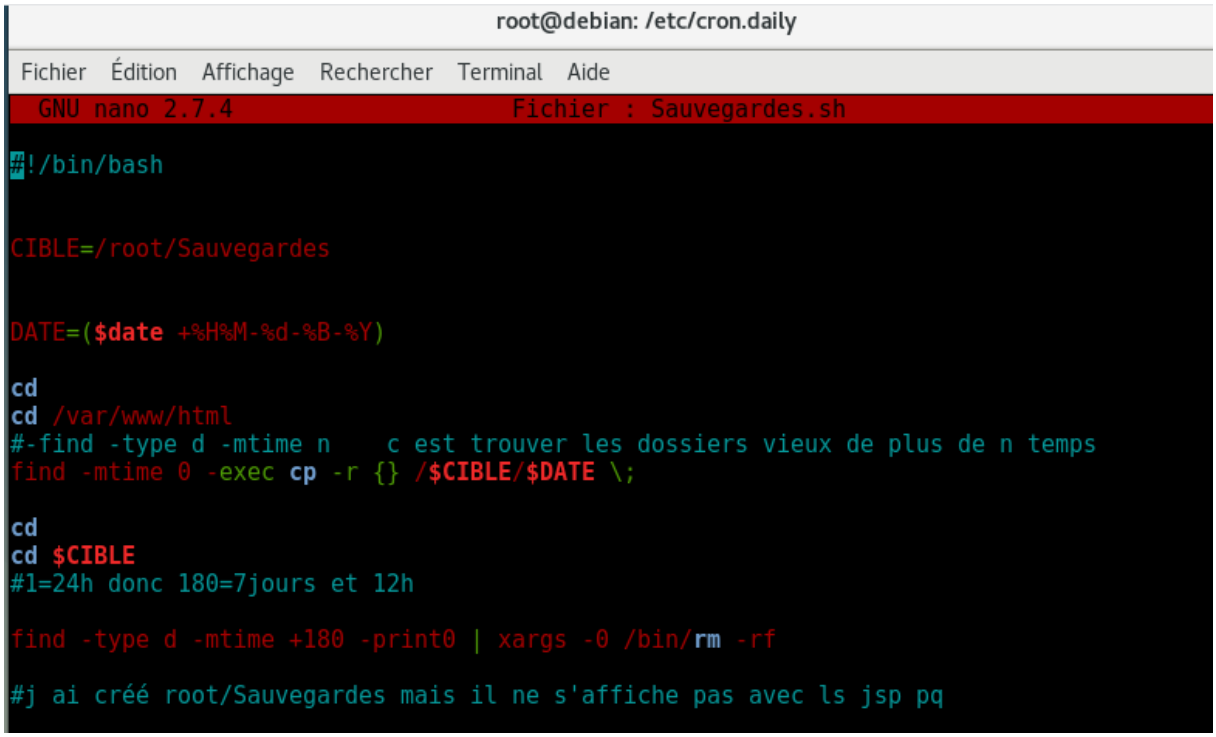
- cp: Copiez tous vos fichiers ou tous vos répertoires.
- dd : Sauvegardez vos partitions et restaurez les.
- rsync: Outil de sauvegarde incrémentale des données.
- unison: Outil de synchronisation bidirectionnelle.
- rdiff-backup: outil de sauvegarde incrémentale des données
- dar:Outil de sauvegarde et d'archivage des données.

07 /02/2019

Prise en main des commandes linux et de la manière dont une sauvegarde peut-être mise en place



Voici un script de sauvegarde incrémentielle pas encore bien au point :



Le $date c est une commande avec laquelle le + %H… définit le format de la date à afficher dans la variable DATE

On utilise la commande find pour trouver les fichiers/dossiers(nature précisée par -type) ainsi que -mtime pour dire « tous les --- vieux de plus de »

08/02/19

Après de nouveaux essais et restructurations, voici le script Sauvegardes.sh

final, localisé dans /etc/cron.daily via le super utilisateur :

Donc comment ce script fonctionne ?

Il est articulé en deux parties : une partie sauvegarde et une partie suppression

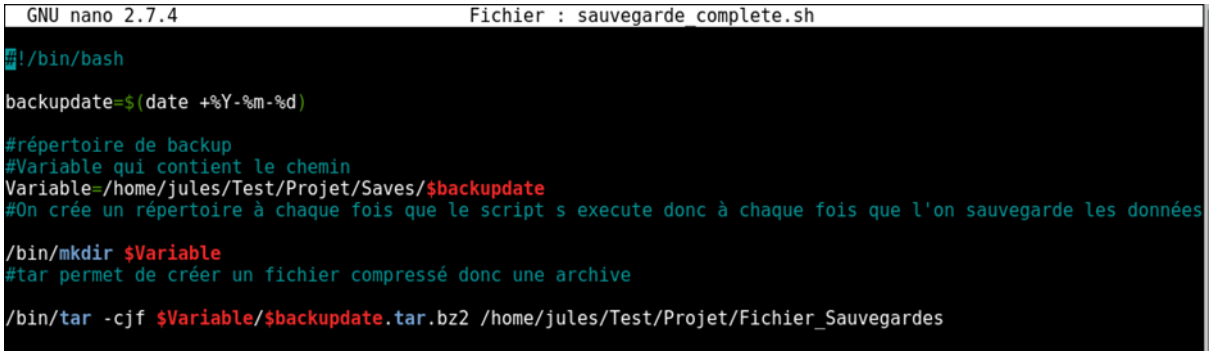
Pourquoi supprimer ? car on doit sauvegarder les modifications mais supprimer les fichiers vieux de plus de 6 mois.

Tout d'abord, on indique le langage c'est-à-dire bash

Ensuite on crée une variable contenant le chemin pour trouver le dossier qui contiendra les sauvegardes.

Puis on crée une variable contenant la fonction date et ses arguments %Hh(heure) %M(mois) %d (jour du mois) %B (Mois complet, cad prend en compte le fait que le mois comporte plus ou moins 30 jours) %Y (année format aaaa)

Note :Le type de sauvegarde mélangera deux scripts avec une sauvegarde complète toutes les semaines puis entre chaque sauvegardes complètes des sauvegardes incrémentales journalières



Ici le code pour la sauvegarde complète, relativement simple. (voir description des scripts si besoin)

Pour automatiser les tâches de la meilleure façon, je vais utiliser crontab -e pour lancer automatiquement les scripts qui seront pour rappel le script sauvegarde_complete.sh et Sauvegardes.sh.

L'utilisation de crontab nous évite de créer un script notifiant l'utilisateur.

En effet, j'ai d'abord pensé une ébauche de script comme suit :

```
  GNU nano 2.7.4              Fichier : Check_Sauvegardes.sh

#!bin/bash
CIBLE=/home/jules/Documents/Sauvegardes1
test=find $CIBLE type -d -mtime -1

if[ -d $CIBLE ]
then
     echo "il n'y a pas d'erreurs"

else

        echo "La sauvegarde ne s'est pas faite correctement"
fi
```

Seulement même si l'idée semble pertinente, après réflexion je me suis tourné vers le crontab et la redirection qui me semblait plus simple d'utilisation

```
* * * * 5 /mnt/hgfs/sauvegarde_complete.sh 2>>/mnt/hgfs/Sauvegarde erreur.log
* 22 * * * /mnt/hgfs/Sauvegardes.sh 2>> /mnt/hgfs/Sauvegarde erreur.log
```

La syntaxe particulière de crontab -e est détaillée dans la description des scripts mais pour rappel l'ordre des nombres est : Minute- Heure - JourDuMois - Mois – JoursDeLaSemaine commande

Donc d'après la syntaxe tous les vendredis on fait une sauvegarde complète et tous les jours à 22h on fait une sauvegarde incrémentielle

Enfin on peut s'intéresser à la structure de la suite du code : 2>>CheminDuFichierErreur.log

Qu'avons-nous fait ? On a redirigé les erreurs dans un fichier, ici le fichier erreur.log ce qui nous permet de savoir si une erreur est survenue

Evidemment on aura crée le fichier dans la destination.
Donc finalement on n'utilisera pas la façon automatique dans cron.daily et cron.weekly mais directement crontab.

Pour automatiser les tâches de la meilleure façon, je vais utiliser crontab -e pour lancer automatiquement les scripts qui seront pour rappel le script sauvegarde_complete.sh et Sauvegardes.sh.

L'utilisation de crontab nous évite de créer un script notifiant l'utilisateur.

En effet, j'ai d'abord pensé une ébauche de script comme suit :

```
 GNU nano 2.7.4              Fichier : Check_Sauvegardes.sh

#!bin/bash
CIBLE=/home/jules/Documents/Sauvegardes1
test=find $CIBLE type -d -mtime -1

if[ -d $CIBLE ]
then
     echo "il n'y a pas d'erreurs"

else

        echo "La sauvegarde ne s'est pas faite correctement"
fi
```

Seulement même si l'idée semble pertinente, après réflexion je me suis tourné vers le crontab et la redirection qui me semblait plus simple d'utilisation

```
* * * * 5 /mnt/hgfs/sauvegarde_complete.sh 2>>/mnt/hgfs/Sauvegarde erreur.log
* 22 * * * /mnt/hgfs/Sauvegardes.sh 2>> /mnt/hgfs/Sauvegarde erreur.log
```

La syntaxe particulière de crontab -e est détaillée dans la description des scripts mais pour rappel l'ordre des nombres est : Minute- Heure - JourDuMois - Mois – JoursDeLaSemaine commande

Donc d'après la syntaxe tous les vendredis on fait une sauvegarde complète et tous les jours à 22h on fait une sauvegarde incrémentielle

Enfin on peut s'intéresser à la structure de la suite du code : 2>>CheminDuFichierErreur.log

Qu'avons-nous fait ? On a redirigé les erreurs dans un fichier, ici le fichier erreur.log ce qui nous permet de savoir si une erreur est survenue

Evidemment on aura crée le fichier dans la destination.
Donc finalement on n'utilisera pas la façon automatique dans cron.daily et cron.weekly mais directement crontab.

Scripts de supervisions

1er script :

```
 GNU nano 2.7.4                                          Fichier : CopieHTTP

#!/bin/bash
# Copie depuis Apache.log
Ip1=$(cut -d ' ' -f1 /var/log/apache2/access.log) 2> /home/gaetan/logs/Copiehttp/error.log
Ip2=$(cut -d ' ' -f1 /var/log/apache2/access.log.1) 2>> /home/gaetan/logs/Copiehttp/error.log
Ip3=$(cut -d ' ' -f1 /var/log/apache2/access.supervision.com.log) 2>> /home/gaetan/logs/Copiehttp/error.log
echo "$Ip1" > /home/gaetan/scripts/Listeips.csv
echo "$Ip2" >> /home/gaetan/scripts/Listeips.csv
echo "$Ip3" >> /home/gaetan/scripts/Listeips.csv
cat /dev/null > /var/log/apache2/access.log
cat /dev/null > /var/log/apache2/access.log.1
cat /dev/null > /var/log/apache2/access.supervision.com.log
```

Pour ce qui est du 1er script, il utilise la commande cut principalement.

On peut voir que j'ai pris le principe de mettre le résultat des commandes dans des variables. D'abord je vais chercher les IP dans différents fichiers log d'Apache me permettant de voir les IP s'étant connecté au site Web. Je ne prends que la première colonne, soit celle où sont les adresses IP, sans rien de plus. Je vais donc ensuite les afficher avec la commande echo, puis en les redirigeant. La première va clean le fichier afin de n'avoir que les IPs de la dernière heure, et les autres se mettront à la suite. Enfin je vais faire cat pour lire un fichier vide et le rediriger dans les logs utilisés au départ afin de les clean aussi, pour que la prochaine heure, je n'ai que les IP s'étant connectée pendant celle-ci.

Les erreurs du script seront redirigées dans un fichier log.

2e script :

```
GNU nano 2.7.4                                  Fichier : etatHttpDNS

#!/bin/bash
# 2e script
# On va chercher le pourcentage de ping perdu lors de la commande httping et on le met dans la variable Ping
Ping=$(httping -g http://192.168.10.10 -c 4 | grep "connects" | cut -d " " -f5-6) 2> /home/benjamin/logs/error.log
# On va chercher le nom de domaine et on le met dans la variable Domain
Domain=$(nslookup 192.168.10.5 | grep 'name' | cut -d ' ' -f3)
# On passe le ssh automatiquement et on se connecte au serveur HTTP pour y créer des fichier contenant erreur/status du serveur apache
sshpass -p '1892' ssh gaetan@192.168.10.10 'systemctl status apache2 |grep "Active" | cut -d " " -f5-6 > /home/gaetan/scripts/Infos.csv 2> /home/gaetan/logs/error.log'
# On copie le fichier créé dans le serveur HTTP et contenant le status du serveur HTTP dans le serveur DNS esclave
sshpass -p '1892' scp gaetan@192.168.10.10:/home/gaetan/scripts/Infos.csv /home/benjamin/scripts/Apache2status.csv
# On copie le fichier créé dans le serveur HTTP et contenant les erreurs dans le serveur DNS esclave
sshpass -p '1892' scp gaetan@192.168.10.10:/home/gaetan/logs/error.log /home/benjamin/logs/Apache2statuserror.log
# On met le contenu du fichier récupéré dans la variable Status
Status=$(cat /home/benjamin/scripts/Apache2status.csv)
# On supprime le fichier désormais inutile
rm /home/benjamin/scripts/Apache2status.csv
# On copie le contenu du fichier erreur récupéré et on le met dans le fichier error.log
cat /home/benjamin/logs/Apache2statuserror.log >> /home/benjamin/logs/error.log
# On supprime le fichier inutilisé
rm /home/benjamin/logs/Apache2statuserror.log
# On récupère le min/moyenne/max de temps de réponse du site, et on le met dans la variable Temps
Temps=$(httping -g http://192.168.10.10 -c 4 | grep 'round-trip' | cut -d ' ' -f2-5)  2>> /home/benjamin/logs/error.log
# On affiche les textes, avec les contenus des variables
echo "Pourcentage de ping perdu, $Ping" > Infos.csv
echo "Nom de domaine, $Domain" >> Infos.csv
echo "Status du serveur Apache, $Status" >> Infos.csv
echo "Temps de réponse du serveur, $Temps" >> Infos.csv
# On copie le fichier final sur le serveur HTTP
sshpass -p '1892' scp -r -p /home/benjamin/scripts/Infos.csv gaetan@192.168.10.10:/home/gaetan/scripts/Infos.csv 2>>/home/benjamin/logs/error.log
```

Pour le deuxième script, on a quelque chose de plus long. Tout d'abord, je vais faire la commande **httping** qui va me permettre de ping l'adresse ip du site. Grâce à grep je vais donc chercher la ligne avec l'information qui m'intéresse. Je vais ensuite utiliser le cut pour récupérer le tout et le mettre dans la variable **Ping**, tout en redirigeant les erreurs dans un log.

Ensuite je vais utiliser nslookup sur l'adresse ip du DNS pour récupérer le nom de domaine. J'ai ensuite fait grep et cut pour stocker le tout dans la variable **Domain**

Pour le status Apache, on se connecte automatiquement via SSH au serveur HTTP, on récupère les informations et les stocke dans un fichier, et les erreurs seront misent dans un log. Ensuite on récupère les deux fichiers et les copie dans le serveur DNS esclave. On copie le contenu du fichier status dans la variable **Status** et le contenu de l'erreur dans un log puis on supprime les fichiers intermédiaires inutiles.

Pour le temps de réponse, j'utilise à nouveau **httping** et **grep, cut**, etc… Pour finalement stocker le résultat dans la variable **Temps**.

On affiche du texte allant avec les contenus des variables.

Puis on copie le fichier final du serveur DNS au serveur HTTP.

3e script :

```
 GNU nano 2.7.4                                          Fichier : csvtohtml

#!/bin/bash
# convertir fichier csv en html

csv2html -o /var/www/supervision/Infos.html  /home/gaetan/scripts/Infos.csv 2>/home/gaetan/logs/errorcsv2html.log
csv2html -o /var/www/supervision/Listeips.html /home/gaetan/scripts/Listeips.csv 2>> /home/gaetan/logs/errorcsv2html.log
```

Pour le 3e script, je vais l'exécuter en crontab root, car sinon il ne fonctionne pas. Le principe est simple, je convertis simplement les fichiers csv obtenus par mes 2 premiers scripts en fichiers html que je vais mettre avec la page d'accueil du site, afin de pouvoir y accéder depuis l'accueil du site de supervision.

Serveur HTTP

Procédure d'installation d'un serveur HTTP

Configuration du serveur HTTP :

Pour configurer ce serveur, nous avons installé un paquet s'appelant « apache2 » avec la commande :

```
root@debian:/home/gaetan# apt-get install apache2
```

Une fois apache2 installé, nous avons donné une adresse ip statique au futur serveur HTTP

```
# The primary network interface
auto ens33
iface ens33 inet static
address 192.168.10.10
netmask 255.255.255.0
```

Maintenant que la machine possède une adresse IP statique (192.168.10.10) et un masque de sous réseau (255.255.255.0), la configuration du serveur DNS peut commencer.

```
root@debian:/# nano /var/www//html/index.html
```

Index.html est la page de base du serveur http, c'est celle qui s'affiche lorsque l'on tape l'adresse IP du serveur dans un navigateur

Le serveur n'abrite qu'un seul site web, pour avoir deux serveurs Web, il faut faire un virtual host

```
root@debian:/etc/apache2/sites-available# nano supervision.com.conf
```

On créée un .conf du nom du site web que l'on souhaite

Et l'on rentre ceci

```
<VirtualHost *:80>
        ServerName supervision.com
        ServerAlias www.supervision.com
        DocumentRoot "/var/www/supervision"
        <Directory "/var/www/supervision">
                Options +FollowSymLinks
                AllowOverride all
                Require all granted
        </Directory>
        ErrorLog /var/log/apache2/error.supervision.com.log
        CustomLog /var/log/apache2/access.supervision.com.log combined
</VirtualHost>
```

Une fois la configuration créée il faut l'activer avec :

```
root@debian:/etc/apache2/sites-available# a2ensite supervision.com
```

Puis redémarrer apache2(systemctl reload apache2)

Ce virtual host utilise la même adresse IP que l'autre serveur web et le dossier avec index.html est dans /var/www/supervision

Conclusion

Pendant le projet nous avons bien déterminer nos tâches tout en prenant un temps pour essayer de comprendre et partager ses connaissances. L'ambiance était bonne et nous nous sommes entraidés tant au niveau conceptuel qu'au niveau technique.