Scripts : explications et intégration

Script 1 (log_ip.sh)

Le but était de récupérer les adresses IPs de tous les clients s'étant connectés sur le site durant l'heure précédente au format .csv. Ce script sera exécuté automatiquement toutes les heures sur le serveur HTTP dans les fichiers de log d'Apache.

Explications du script en détails :

Dans un premier temps, nous allons créer un dossier « Admin » et deux sous-dossiers : « log_ip » et « log_dns » qui permettront de sauvegarder les informations envoyées. Dans le cas où les dossiers existent déjà, nous allons tout simplement rediriger les messages d'erreurs dans la « poubelle ».

```
mkdir /home/gonfreecs/Bureau/Admin/ 2>/dev/null
```

```
mkdir /home/gonfreecs/Bureau/Admin/log_ip 2>/dev/null
```

```
mkdir /home/gonfreecs/Bureau/Admin/log_dns 2>/dev/null
```

Nous allons ensuite supprimer l'ancien fichier pour par la suite le remplacer par le nouveau.

```
rm /home/gonfreecs/Bureau/Admin/log_ip/log_ip.csv
```

Nous allons créer deux variables. L'une pour stocker l'heure actuelle et l'autre pour stocker le jour actuel.

```
heure_actu=`date +%H`
```

```
jour_actu=`date +%e`
```

Nous allons regarder ligne par ligne le fichier « /var/log/apache2/access.site.carnofluxe.domain.log ». C'est ici où sont stockées les IPs des clients s'étant connectés au serveur HTTP, la date et l'heure à laquelle ils se sont connectés, le système d'exploitation utilisé, le type de navigateur utilisé, ainsi que d'autres données moins importantes.

```
cat /var/log/apache2/access.site.carnofluxe.domain.log | while read
ligne;
```

Tant qu'il y a une ligne, nous allons récupérer le jour et la date, et les stocker dans deux variables : « jour_x » et « heure_x ».

```
do
```

```
	jour_x=`echo $ligne | cut -d" " -f4 | cut -d"/" -f1 | cut -d"[" -f2`

	heure_x=`echo $ligne | cut -d" " -f4 | cut -d":" -f2`
```

Sachant que le script sera lancé toutes les heures à xxh59, nous allons comparer le jour et l'heure de la ligne avec le jour et l'heure actuels.

```
	if [ $jour_x -eq $jour_actu ] && [ $heure_x -eq $heure_actu ]
```

Dans le cas où les deux jours et les deux heures sont identiques, nous récupérons l'IP et nous la stockons dans une variable.

```
	then

		ip=`echo $ligne | cut -d" " -f1`
```

Nous avons également essayé d'associer l'IP à son pays, mais le problème est qu'il n'est pas possible d'utiliser l'api faite pour (« http://ip-api.com/csv ») puisque nous n'avons plus accès à internet.

```
		#pays=`GET http://ip-api.com/csv/$ip | cut -d, -f2`
```

Nous allons ensuite stocker notre IP et notre pays dans un fichier temporaire. Dans notre cas, nous allons stocker uniquement l'IP (cf. à la deuxième ligne qui n'est pas commentée).

```
		#echo $ip, $pays >> log_ip_tempo.tmp

		echo $ip, >> log_ip_tempo.tmp

	fi
```

Dans le cas où il n'y aurait aucune IP dans le fichier « /var/log/apache2/ access.site.carnofluxe.domain.log », nous créons quand même un fichier malgré le fait qu'il soit vide.

```
	if [ -n "log_ip_tempo.tmp" ]

	then

		touch "log_ip_tempo.tmp"

	fi

done
```

Il faut ensuite enregistrer toutes nos données dans un fichier .csv, sans oublier de retirer les doublons. Pour cela, nous allons utiliser la fonction « uniq » qui permet de le faire.

```
cat log_ip_tempo.tmp | uniq >> /home/gonfreecs/Bureau/Admin/log_ip/log_ip.csv
```

Et pour finir, nous supprimons le fichier temporaire.

```
rm log_ip_tempo.tmp 2>/dev/null
```

Intégration au serveur HTTP :

Pour implémenter ce script, il suffit de créer un fichier dans « /bin » avec comme nom « log_ip.sh » et d'utiliser "crontab -e" pour l'automatiser. Sachant que nous exécutons ce script toutes les heures, il faut rajouter à la dernière ligne du fichier "crontab -e" :

```
59 * * * * /bin/log_ip.sh
```

Script 2 (log_dns.sh)

Le but était de générer un fichier au format csv qui remonte les informations suivantes :

- Ping sur le serveur HTTP
- Fonctionnement de la résolution de nom
- Accessibilité au site
- (Temps de réponse du site web)

Ce script sera exécuté toutes les 5 minutes sur le serveur DNS esclave.

Ce script recopiera le fichier .csv généré dans le répertoire souhaité sur le serveur HTTP en fin d'exécution grâce au protocole SSH et devra alerter l'administrateur en cas de problème.

Explications du script en détails :

Dans un premier temps nous allons calculer la date actuelle.

```
today=`date '+%Y-%m-%d_%Hh%M'`
```

Puis nous allons indiquer l'IP de la machine et l'URL du site.

```
ip_serveur="192.168.10.10"
```

```
url_site="site.carnofluxe.domain"
```

Nous essayons de récupérer l'IP du site à partir de l'URL (pour savoir si le DNS fonctionne).

```
ip_site=`dig $url_site | sed -n '15p' | cut -c36-48`
```

Et nous commençons à remplir notre fichier .csv. Nous essayons de faire un ping et de récupérer son temps de réponse. Nous récupérons par la suite ce temps dans une variable.

```
ping -c 1 $ip_serveur | cut -d"=" -f4 | sed -n '2p' > "log_dns.csv"
```

```
status=`cat "log_dns.csv"`
```

Dans le cas où la variable n'est pas vide (c'est-à-dire qu'elle a réussi à faire un ping), nous rentrons dans notre boucle, nous indiquons que la connexion est établie entre notre machine et notre serveur HTTP et nous stockons cette information dans notre fichier .csv. Le paramètre « </br> » présent dans la fonction « echo » permettra par la suite de pouvoir faire des retours à la ligne dans notre fichier .html.

```
if [ -n "$status" ]
```

```
 then
```

```
        echo "Connexion établie entre votre machine et le serveur HTTP !</br>" > "log_dns.csv"
```

Nous allons maintenant essayer de tester le DNS. Si nous pouvons récupérer une IP à partir de l'URL donnée, cela signifie que notre DNS est fonctionnel. Dans notre condition, nous en profiterons donc d'indiquer l'IP trouvée (elle doit normalement correspondre à l'IP de la machine), ainsi que le temps de réponse du ping ; que nous stockerons à la suite dans notre fichier .csv.

```
        if [ -n "$ip_site" ]
        then
                echo "Adresse ip du site : $ip_site </br>" >> "log_dns.csv"
                echo "Temps de connexion avec le site : $status </br>" >> "log_dns.csv"
```

Nous allons à présent essayer de télécharger la page du site et de calculer son temps de chargement. Pour ce faire, nous utilisons la fonction « wget » qui permet de télécharger la page et de nous donner beaucoup d'informations dont notre temps de chargement. Nous avons donc fait un « wget » de notre site et stocké toutes les informations dans un fichier temporaire (« log_dns_tempo.tmp ») ; puis récupérer l'information qui nous intéresse.

```
                wget $url_site 2> "log_dns_tempo.tmp"
                temps_charge=`sed -n '8p' "log_dns_tempo.tmp" | cut -c74-75`
```

Et si tout se passe bien, nous avons récupérer ce temps de chargement et nous l'avons ajouté dans notre fichier .csv.

```
                if [ -n "$temps_charge" ]
                then
                        echo "Temps de chargement de la page : $temps_charge </br>" >>
"log_dns.csv"
```

Nous en profitons ensuite pour supprimer la page téléchargée (« index.html »), sachant que nous la téléchargeons toutes les 5 minutes, elle prendrait de la place inutilement sur notre machine.

```
                        rm index.html
```

Viennent ensuite les cas qui ont pu rencontrer des problèmes. Dans chacun des cas, nous stockons les erreurs rencontrées dans notre fichier .csv et nous envoyons un mail pour signaler l'incident. Le mail aura comme sujet le type de problèmes rencontrés avec la date et l'heure actuels ; et comme texte l'erreur précise rencontrée. Dans notre situation, les mails sont envoyés uniquement sur la machine où le script a été exécuté.

```
                else
```

```
                    echo "ERREUR : impossible d'accéder au site via son nom de domaine !
</br>" >> "log_dns.csv"

                    echo "ERREUR : impossible d'accéder au site via son nom de domaine !" |
mail -s "Alerte_DNS_$today" admin@carnofluxe.com

                fi

            else

                echo "ERREUR : impossible d'accéder au site !</br>" >> "log_dns.csv"

                echo "ERREUR : impossible d'accéder au site !" | mail -s "Alerte_DNS_$today"
admin@carnofluxe.com

            fi

        else

        echo "ERREUR : aucune connexion entre votre machine et le serveur HTTP !</br>" >>
"log_dns.csv"

        echo "ERREUR : aucune connexion entre votre machine et le serveur HTTP !" | mail -s
"Alerte_DNS_$today" admin@carnofluxe.com

fi
```

Une fois que toutes nos informations ont été stockées dans le fichier .csv (dont les erreurs qui ont pu être rencontrées), il faut maintenant envoyer notre fichier sur le serveur HTTP. Pour ce faire, nous utilisons le protocole SSH ainsi qu'une extension qui se nomme « sshpass » et qui permet d'indiquer le mot de passe pour accéder à la machine sans que nous ayons besoin de le renseigner. À savoir que cette méthode n'est pas très recommandée puisqu'elle affiche très clairement le mot de passe de notre machine dans le script. Il faut donc veiller à ce que le script ne puisse pas être lisible ou modifiable par une tierce personne en configurant les permissions de notre fichier.

NB : le mot de passe utilisé est « » (3 espaces).

NB² : pour utiliser une première fois « shpass » dans notre script, il faut d'abord approuver la connexion entre nos deux machines. Pour le faire, il suffit de lancer la commande sans « sshpass -p " " » (cf. à la troisième ligne qui est commentée par un #), de rentrer manuellement le mot de passe et de rentrer « yes » dans le terminal au moment où il vous sera demandé si vous souhaitez approuver la connexion.

```
sshpass -p "   " scp "log_dns.csv"
gonfreecs@192.168.10.10:/home/gonfreecs/Bureau/Admin/log_dns

#scp "log_dns.csv" gonfreecs@192.168.10.10:/home/gonfreecs/Bureau/Admin/log_dns
```

Et nous terminons par supprimer le fichier temporaire.

```
rm log_dns_tempo.tmp 2>/dev/null
```

Intégration au serveur DNS esclave :

Voici la liste des paquets à installer pour que le script fonctionne correctement :

- apt-get install mailutils
- apt-get install mutt
- apt-get install ssh
- apt-get install openssh-server
- apt-get install sshpass

Pour implémenter ce script, il suffit de créer un fichier dans « /bin » avec comme nom « log_dns.sh » et d'utiliser "crontab -e" pour l'automatiser. Sachant que nous exécutons ce script toutes les 5 minutes, il faut rajouter à la dernière ligne du fichier "crontab -e" :

```
*/5 * * * * log_dns.sh
```

Script 3 (regen.sh)

Le but était de créer un script qui sera exécuté toutes les 5 minutes et régénèrera la page du site web de supervision à partir des fichiers csv. Si un problème devait survenir, la page doit indiquer l'erreur et idéalement montrer l'état du site au cours du temps.

Explications du script en détails :

Dans un premier temps nous allons calculer la date actuelle.

```
today=`date '+%Y-%m-%d_%Hh%M'`
```

Puis nous stockons nos deux fichiers précédemment générés dans deux variables. La première variable (« CSV_client ») stocke les IPs des clients s'étant connectés la dernière heure. La seconde (CSV_infos) stocke les différentes informations sur l'état de la connexion.

```
CSV_client=`echo "/home/gonfreecs/Bureau/Admin/log_ip/log_ip.csv"`
CSV_infos=`echo "/home/gonfreecs/Bureau/Admin/log_dns/log_dns.csv"`
```

Nous stockons les IPs et les informations dans deux autres variables. Nous calculons ensuite le nombre de clients (en calculant le nombre d'IPs différentes) et nous stockons ce résultat dans une variable.

```
IPs=`cat $CSV_client`
Infos=`cat $CSV_infos`
NB_IP=`wc -l $CSV_client | cut -d' ' -f1`
```

Nous vérifions si le fichier envoyé par le serveur DNS esclave existe et dans le cas où il n'existe pas, nous envoyons une alerte par mail.

```
if [ -z $CSV_infos ]
then
        echo "ERREUR : il n'y a pas de fichier : log_dns.csv"
        echo "ERREUR : il n'y a pas de fichier : log_dns.csv" | mail -s "Alerte_Regen_$today" admin@carnofluxe.com
fi
```

Nous mettons en forme le code .html de manière à ce qu'il soit reconnu ; sans oublier d'ajouter nos données. Et à la fin, nous créons notre fichier au format .html.

```
echo "<html>
 <head>
  <meta charset=utf-8>
  <title>Adresses clients</title>
 </head>
 <body>
  <h1>Nombre de clients s'étant connectés durant la dernière heure : $NB_IP<br/>$IPs</h1>
 <body>
  <h1>$Infos</h1>
 </body>
</html>
" > /var/www/superv.carnofluxe/index.html
```

Et nous finissons par renommer le fichier « log_dns.csv » en ajoutant la date actuelle pour que nous puissions garder une trace du fichier dans le cas où le fichier aurait stocké des erreurs.

```
mv /home/gonfreecs/Bureau/Admin/log_dns/log_dns.csv
/home/gonfreecs/Bureau/Admin/log_dns/log_dns_$today.csv
```

Intégration au serveur HTTP :

Voici la liste des paquets à installer pour que le script fonctionne correctement :

- apt-get install mailutils
- apt-get install mutt
- apt-get install ssh
- apt-get install openssh-server
- apt-get install sshpass

Pour implémenter ce script, il suffit de créer un fichier dans « /bin » avec comme nom « regen.sh » et d'utiliser "crontab -e" pour l'automatiser. Sachant que nous exécutons ce script toutes les 5 minutes, il faut rajouter à la dernière ligne du fichier "crontab -e" :

```
*/5 * * * * regen.sh
```