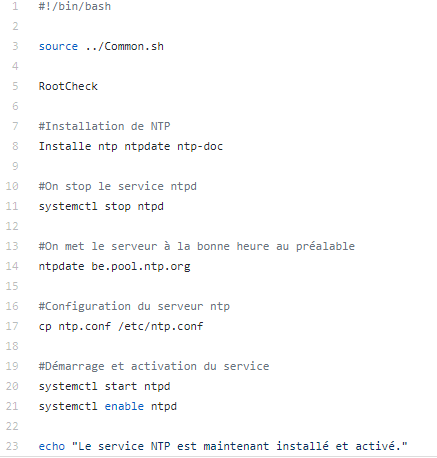
**NTP**

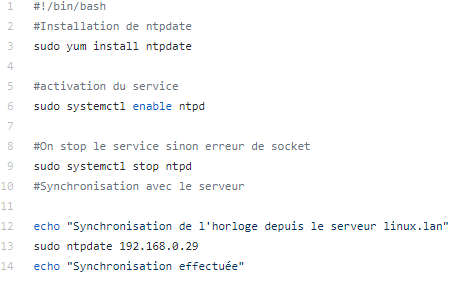
Le protocole NTP (Network Time Protocol) va permettre de synchroniser les horloges des ordinateurs connectés au même réseau local que celle du serveur de temps.  
Celui-ci devra synchroniser sa propre horloge en contactant un serveur de temps de référence à distance donc par internet.

Cette synchronisation des heures permettra entre autres de ne pas perturber certaines applications utilisant l’horloge du système mais aussi pour donner plus de cohérence en cas de comparaison des messages de « logs » de plusieurs ordinateurs sur le réseau.

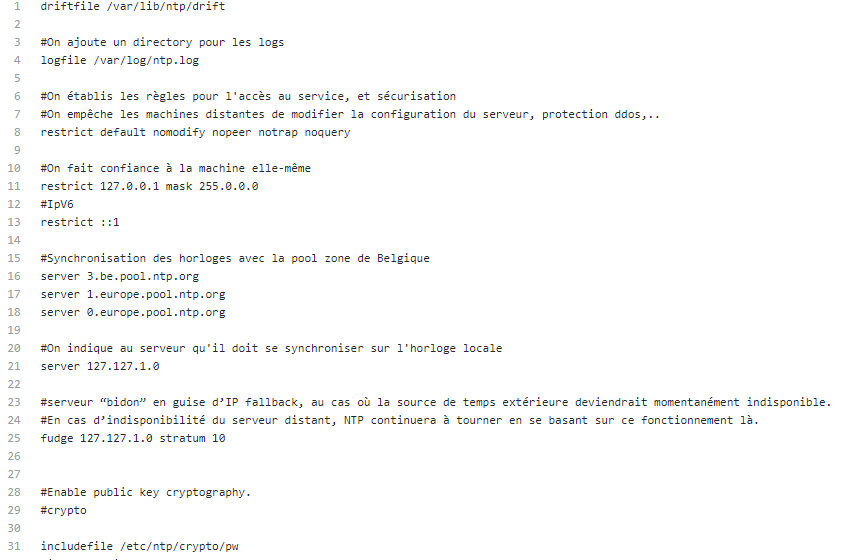
**Script côté serveur**



**Script côté client**



**Fichier de config ntpd**



On déclare d’abord le fichier de dérive « driftfile ». Il va permettre de corriger les dérives de l’horloge système en l’absence de connexion réseau au serveur de référence.

On déclare ensuite le répertoire et le fichier pour stocker les « logs » du service ntpd.

On permet la synchronisation avec notre source de temps mais on interdit à la source de modifier ou d’interroger le service sur ce système.

On autorise les accès sur l’interface de bouclage

Ensuite, on se synchronise avec les serveurs NTP belge de référence. Pour résoudre les problèmes de charge on entre plusieurs adresse. Il s’agit d’un groupement de serveur et la redistribution se fait à l’aide du Round Robin DNS (association de plusieurs adresses IP à un FQDN)

Enfin, on précise au serveur de se synchroniser sur l’ « Undisciplined Local Clock » et on indique un faux « pilote » destiné à la sauvegarde de l’heure dans le cas où aucune source externe d’heure synchronisée n’est disponible.

**Serveur**

1. *#!/bin/bash*
2. **source** ..**/**Common.sh
4. RootCheck
5. *#Check installation du support EPEL*
6. Installe epel-release
8. *#Installation de tous les composants de ClamAV*
10. Installe clamav-server clamav-data clamav-update clamav-filesystem clamav clamav-scanner-systemd clamav-devel clamav-lib clamav-server-systemd
12. *#Configuration du daemon Clam*
13. *#Copie du template dans le cas où l'on a pas de fichier de configuration*
14. **cp** **/**usr**/**share**/**clamav**/**template**/**clamd.conf **/**etc**/**clamd.d**/**clamd.conf
16. *#Activation de Freshclam pour garder la DB à jour*
18. *#Création du service freshclam et configuration*
19. **echo** "# Run the freshclam as daemon
20. [Unit]
21. Description = freshclam scanner
22. After = network.target
23. [Service]
24. Type = forking
25. ExecStart = /usr/bin/freshclam -d -c 4
26. Restart = on-failure
27. PrivateTmp = true
28. [Install]
29. WantedBy=multi-user.target" **>** **/**usr**/**lib**/**systemd**/**system**/**clam-freshclam.service
31. *#Démarrage et activation du service au démarrage*
32. Service clam-freshclam.service
34. *#Changement des fichiers service autrement  clamd@.service ne démarre pas*
35. *#On renomme le fichier si on l'a pas déjà fais*
36. **if** **[** **!** -e "/usr/lib/systemd/system/clamd.service" **]**;**then**
37. **mv** **/**usr**/**lib**/**systemd**/**system**/**clamd**@**.service **/**usr**/**lib**/**systemd**/**system**/**clamd.service
38. **fi**
40. *#On modifie le fichier clamd@scan.service et on change la référence vers*
41. *# /lib/systemd/system/clamd.service*
42. **echo** ".include /lib/systemd/system/clamd.service
43. [Unit]
44. Description = Generic clamav scanner daemon
45. [Install]
46. WantedBy = multi-user.target" **>** **/**usr**/**lib**/**systemd**/**system**/**clamd**@**scan.service
48. *#On modifie le fichier /usr/lib/systemd/system/clamd.service et on configure*
49. **echo** "[Unit]
50. Description = clamd scanner daemon
51. After = syslog.target nss-lookup.target network.target
52. [Service]
53. Type = simple
54. ExecStart = /usr/sbin/clamd -c /etc/clamd.d/clamd.conf --foreground=yes
55. Restart = on-failure
56. PrivateTmp = true
57. [Install]
58. WantedBy=multi-user.target" **>** **/**usr**/**lib**/**systemd**/**system**/**clamd.service
60. *#Démarrage et automatistion des services*
61. Service clamd.service
62. Service [clamd**@**scan.service](mailto:clamd@scan.service)

Ce script va permettre d’installer l’antivirus sur le serveur pour amélioré sa sécurité

On vérifie d’abord que l’utilisateur exécute bien le script en tant que root, ce qui est nécessaire pour configurer les fichiers de ClamAV.

On installe ensuite EPEL (Extra Package for Entreprise Linux) qui est un repo fournissant des package additionnels pour les distributions de type RedHat/CentOs.

Ensuite, au cas où on ne possède pas de fichier de configuration, on copie le template clamd.conf. On peut créer le service freshclam et le configurer de sortie qu’il vérifie 4 fois par jour la précense de mise à jour. On le démarre et on l’active au démarrage ensuite.  
  
On renomme également le fichier /usr/lib/systemd/system/clamd@.service en /usr/lib/systemd/system/clamd.service autrement, par défaut le service ne fonctionne pas. Puis, il faut indiquer le bon chemin du clamd.service dans le fichier [/usr/lib/systemd/system/clamd@scan.service](mailto:/usr/lib/systemd/system/clamd@scan.service) **et on peut configurer le fichier clamd.service  
  
Les dernières lignes vont pemettre d’activer et de démarrer automatiquement les services clamd et clamd@scan au démarrage**

Test de l’antivirus

1. *#!/bin/bash*
3. date=$**(date** +**%**d\_**%**m\_**%**Y\_**%**H\_**%**M**)**
5. **echo** Arret **du** service freshclam et mise à jour
7. service clam-freshclam stop
9. **sudo** freshclam
11. service clam-freshclam start
13. **echo** Service  **red**émarré et mise à jour effectuée
14. **echo** Entrez le nom **du** repertoire ou fichier à analyser:
15. **read** nom
17. **if** **[** -e "$nom" **]**;**then**
18. **echo** Analyse en cours ....
20. **echo** "--------------------------------------------------------------------  ---**\n**" **>>** $HOME**/**analyseVirus.log
21. **echo** " " **>>** $HOME**/**analyseVirus.log
22. **echo** "Analyse du $date**\n**" **>>** $HOME**/**analyseVirus.log
23. **echo** " "  **>>** $HOME**/**analyseVirus.log
25. clamscan -l $HOME**/**analyseVirus.log -r $nom
26. **else**
27. **echo** Vous avez entré un nom de repertoire ou fichier inexistant
28. **fi**

Ce script va permettre d’exécuter une analyse virale d’un dossier ou fichier choisis par l’utilisateur qui lance le script.

On met d’abord à jour la base de données de l’antivirus et puis on demande à l’utilisateur de spécifier la cible. Si elle existe, on procède à l’analyse.

Rajouter fichier ?? Au choix