



# Configurer un serveur GNU/Linux Ubuntu-Server 8.04.1

## Sommaire :

- I) Téléchargement et installation du système
- II) Configuration initiale et mises à jours
  - 1. Configurer l'utilisateur root
  - 2. Configurer le réseau
  - 3. Faire les mises à jours
  - 4. Configurer SSH
  - 5. Configurer le firewall
  - 6. Manipulations diverses (NTP, ...)
- III) Configuration en serveur web Apache2 avec SSL
- IV) Configuration en serveur de partage de fichiers SAMBA
- V) Configuration en serveur FTP
- VI) Configuration en serveur Proxy Squid
- VII) Configuration en serveur de mails
- VIII) Configuration de Fail2ban
- IIX) Gérer les services du système

## Liens :

Page officielle Ubuntu-Server : <http://www.ubuntu.com/products/whatIsubuntu/serveredition>

Téléchargement : <http://www.ubuntu.com/getubuntu/download>

Documentation Ubuntu-Server : <https://help.ubuntu.com/8.04/serverguide/C/index.html>

Voici un tutoriel pour apprendre à configurer un serveur sous la distribution GNU/Linux Ubuntu-server 8.04.1. Cette version, qui dispose du « LTS » (Long Term Support), sera mise à jour pendant 3 ans, ce qui est un avantage pour un serveur.

Contrairement à la version Desktop, Ubuntu-Server est optimisé pour les architectures i686, processeurs 32 et 64bits, et peut supporter jusqu'à 64Go de ram grâce à l'activation de la PAE. Etant basée sur Debian, ce tutoriel devrait également fonctionner sur celui-ci, mais vous n'êtes pas à l'abri d'éventuels changement de noms sur les packages qui seront nécessaires. Cela dit, l'adaptation ne doit pas être difficile, car le tutoriel montre assez bien le chemin à suivre.

J'ai créé ce tutoriel en me basant sur la documentation officielle de la distribution, qui est cela dit extrêmement bien faite, mais en anglais. Ce tutoriel est donc une version light de cette documentation. Si vous avez des difficultés, des questions, n'hésitez pas à consulter la vraie documentation (lien au dessus), consulter les forums de la distribution, ou à m'envoyer un mail (allez sur mon site dont le lien est en pied de page, vous pourrez me contacter).

Pour faire fonctionner le serveur correctement sur cette distribution, avec tous les services décrits ci-après, il sera nécessaire de posséder au moins 256Mo de ram, 512Mo recommandés pour une meilleure rapidité, un disque-dur assez rapide (7200tr/min), environ 5Go seront nécessaires pour l'OS, et un processeur 32 ou 64bits, d'au moins 1Ghz (cela fonctionnerait aussi avec moins). Mettez la carte graphique la moins puissante que vous possédez, afin d'économiser de l'énergie qui serait gaspillée inutilement avec une carte graphique récente, et d'éviter un surplus de chaleur. Pour ma part j'utilise une machine qui possède cette configuration :

AMD Duron 1,8Ghz  
2x512Mo DDR400  
HDD 160Go  
CG Ati Rage 2Mo

Le système a également été testé sur une machine ancienne possédant cette configuration :

Intel Pentium III 600Mhz  
384 Mo SDRAM PC100  
HDD 40Go  
CG Ati Rage 8Mo

## I) Téléchargement et installation du système.

Tout d'abord, téléchargez la distribution si ce n'est pas déjà fait, ici :

<http://www.ubuntu.com/getubuntu/download>

Gravez l'image et bootez avec le CD sur votre machine. Pendant l'installation, il vous sera demandé :

- le partitionnement de disque à effectuer, c'est à votre choix, si vous ne savez pas, laissez par défaut.
- Le mot de passe root (très important, utilisez au moins 8 caractères, lettres, chiffres, majuscule et caractères spéciaux).
- Un utilisateur et son mot de passe.
- Les packages à installer (Serveur de mail, LAMP, DNS....) c'est à vous de voir en fonction de ce qui vous intéresse, ne vous inquiétez pas si vous oubliez un choix, on pourra y revenir plus tard une fois le système installé grâce à la commande **tasksel**.

Une fois l'installation terminée, la machine redémarre, on va pouvoir commencer à travailler sur la configuration de base du système.

## II) Configuration initiale et mises à jours

Pour commencer, il faut effectuer plusieurs tâches :

1. Configurer l'utilisateur root
2. Configurer le réseau
3. Faire les mises à jours
4. Configurer OpenSSH-Server
5. Configurer le firewall
6. Manipulations diverses (NTP, ..)

### 1. Configurer l'utilisateur root

Ubuntu-server a choisi comme tout système Ubuntu de ne pas autoriser par défaut la commande su. Il faut donc utiliser devant chaque commande nécessitant les droits super utilisateur la commande sudo.

Si vous souhaitez activer su, il faut activer root en lui donnant un mot de passe :

```
sudo passwd root
```

Donnez lui le même mot de passe qu'à l'installation. Maintenant, vous pourrez vous loguer en tant que root par la commande su.

Si vous souhaitez un jour désactiver root, exécutez cette commande :

```
sudo passwd -l root
```

*Loguez vous en tant que su si vous avez fait la manipulation, sinon pensez à rajouter sudo devant les commandes du tutoriel.*

## 2. Configurer le réseau

Par défaut le système a configuré les interfaces en DHCP. Si c'est ce que vous souhaitez, vous pouvez afficher l'ip actuelle par la commande :

```
ip a
```

Néanmoins, il est très fortement déconseillé d'utiliser une adresse DHCP sur un serveur, sauf si le DHCP du réseau a été configuré pour donner la même adresse IP au serveur (par le biais d'une restriction MAC).

Je vous conseille d'effectuer cette manipulation afin de donner une IP fixe :

Tout se passe dans le fichier **/etc/network/interfaces**.

```
nano /etc/network/interfaces
```

Repérez la partie concernant votre interface réseau. Ici c'est eth0. Les IP sont données en exemple, modifiez-les comme vous le souhaitez.

```
iface eth0 inet static
    address 192.168.0.2
    netmask 255.255.255.0
    gateway 192.168.0.1
```

Pour une configuration en DHCP, voici ce que contiendrait le fichier :

```
auto eth0
iface eth0 inet dhcp
```

Configurons maintenant le domaine, et les serveurs DNS. Ça se joue dans **/etc/resolv.conf**.

```
nano /etc/resolv.conf

search mydomain.example
nameserver 192.168.0.1
nameserver 4.2.2.2
```

Vous pouvez mettre autant de DNS que vous le souhaitez, les premiers sont les prioritaires. Conseil : Pour éviter les problèmes et les lenteurs de DNS, sauf si vous souhaitez que ce serveur soit un serveur DNS, donnez les IP des DNS de votre FAI. Vous les trouverez facilement sur google.

Si vous avez besoin de déclarer des hôtes, ça se passe ici **/etc/hosts**. Il suffit simplement de donner l'IP et sur la même ligne le nom d'hôte.

Une fois cette configuration effectuée, relancez-le réseau par la commande :

```
/etc/init.d/networking restart
```

Vérifiez ensuite que le réseau fonctionne en effectuant un ping :

```
ping www.google.fr
```

Si tout va bien, vous verrez que les pings sont « transmitted ».

### 3. Faire les mises à jours

Maintenant que le réseau est configuré, voici les commandes pour les mises à jours :

On rafraîchi les dépôts :

```
aptitude update
```

On lance la recherche et installation des mises à jours :

```
aptitude safe-upgrade
```

Si vous souhaitez valider par défaut l'installation des mises à jours, rajoutez le paramètre **-y** à la commande.

Si vous voyez que toutes les mises à jours ne sont pas passées (par exemple, une mise à jour de kernel), soit, lancez la commande

```
aptitude full-upgrade
```

ou utilisez le gestionnaire **aptitude**, et dans la rubrique Updates, appuyez sur « + » sur les mises à jours non effectuées, puis deux fois sur « g » pour valider.

Si un kernel a été mis à jour, redémarrez votre machine.

```
reboot
```

### 4. Configurer OpenSSH-Server

Configurer le SSH est important pour éviter de laisser une porte ouverte sur le monde vers votre serveur. Par défaut SSH est configuré pour accepter les connexions d'un utilisateur du système par

simple demande de mot de passe.

La meilleure solution reste de changer le port du SSH, ne pas autoriser le login de root, ni la connexion par mot de passe, mais uniquement pour un utilisateur donné avec une clé privée.

Tout d'abord, si à l'installation vous n'aviez pas coché Serveur OpenSSH, il va falloir le faire maintenant :

```
apt-get install openssh-server
```

Pour se connecter par ssh sur un serveur, vous pouvez utiliser le client Putty sous Windows, ou la commande **ssh** sous Linux.

Pour éviter les problèmes, laissez ouverte une deuxième console sous SSH au cas où.

Sauvegardons l'original de la configuration et changeons les droits pour éviter les problèmes :

```
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.original  
chmod a-w /etc/ssh/sshd_config.original
```

Maintenant, ouvrez le fichier de configuration :

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

Repérez ces lignes pour pouvoir modifier la configuration comme indiqué :

```
Port 2222 //Le port par défaut est le 22, c'est conseillé de changer.  
PermitRootLogin no //root ne pourra pas se connecter  
RSAAuthentication yes  
PubkeyAuthentication yes  
AuthorizedKeysFile      %h/.ssh/authorized_keys  
PermitEmptyPasswords   no  
PasswordAuthentication no //connexion par simple mot de passe désactivée
```

Avec cette configuration, la seule solution de connexion sera d'utiliser une clé privée, fichier crypté qui vous permettra à la fois d'être le seul à pouvoir ouvrir une connexion, et à crypter les échanges du SSH vers le serveur sur le réseau.

Ce système fonctionne avec deux clés, une publique (.pub), une privée (sans extension). La privée permet de générer la publique, la publique ne permet pas de générer la privée. La clé publique doit être copiée dans un fichier du système pour SSH, la privée doit être conservée précieusement en dehors du serveur, et ne doit pas traîner n'importe où !

**Tout d'abord, créons les clés – vous devez faire cette manipulation en étant logué avec l'utilisateur qui vous intéresse, surtout pas en root :**

```
ssh-keygen -t rsa -b 2048
```

Une passphrase vous sera demandée, elle n'est pas obligatoire. Un nom de clé vous sera aussi demandé, ici on prendra par exemple key-rsa. La clé sera créée dans le répertoire courant.

Copiez la clé publique dans le fichier d'autorisation de SSH :

```
mkdir /home/votrelogin/.ssh  
cat key-rsa.pub >> /home/votrelogin/.ssh/authorized_keys  
chmod 644 /home/votrelogin/.ssh/authorized_keys
```

Maintenant, déplacez la clé privée créée sur votre pc client, ne la laissez pas sur le serveur. Utilisez une clé usb par exemple, qu'il faudra monter manuellement via la commande (remplacez les étoiles par le chemin de votre clé, en général /dev/sda1 ou /dev/sdb1...) :

```
mount /dev/sd** /media/floppy  
cp key-rsa /media/floppy
```

N'oubliez pas de démonter la clé avant de la débrancher :

```
umount /media/floppy
```

Revenez en root. Redémarrons maintenant SSH pour valider la configuration :

```
/etc/init.d/ssh restart
```

Avec une nouvelle console, essayez de vous connecter. Voici quelques explication :

Pour utiliser la clé privée sous Windows, il vous faudra la convertir au format clé privée Windows (.ppk) avec puttygen (téléchargeable gratuitement sur internet). Sous Putty, on donne le chemin de la clé ppk dans la catégorie **Connection/SSH/Auth**. N'oubliez pas d'indiquer comme connexion **votre-login@ip-de-votre-serveur**

La clé privée sans extension servira pour une machine cliente linux. Voici la syntaxe pour ssh :

```
ssh votre-login@ip-serveur -p 2222 -X -i chemin_vers_clé_privée
```

L'option X permet l'export du serveur X chez le client, vous permettant d'ouvrir un logiciel graphique du serveur sur votre machine linux cliente.

L'option p change le port, et le i donne le chemin vers la clé privée.

Si la connexion donne un message d'avertissement pour les droits de la clé, changez les :

```
chmod 500 chemin_vers_clé_privée
```

Pour vérifier la configuration, vous pouvez également essayer de vous connecter en tant que root, avec ou sans clé, vous serez refusé.

## 5. Configurer le firewall

Ubuntu-Server utilise par défaut le firewall UFW, qui fonctionne vraiment bien et qui est assez simple. Il n'est pas activé par défaut.

Activons le :

```
ufw enable
```

Pour autoriser un port, par exemple le 2222 pour le SSH on utilise la commande (à faire obligatoirement !) :

```
ufw allow 2222
```

Pour bloquer un port, c'est presque pareil :

```
ufw deny numéro_port
```

Si vous voulez supprimer une règle créée par exemple un blocage du port 1234 :

```
ufw delete deny 1234
```

Pour afficher les ports bloqués/autorisés, le statut du firewall :

```
ufw status
```

Vous pouvez obtenir des commandes plus détaillées sur la documentation anglaise d'Ubuntu-Server, ou sur google (c'est l'ami de tout informaticien).

## 6. Manipulations diverses

Une chose est très importante sur un serveur, mais fréquemment oublié...l'horloge. Pour éviter les erreurs, rien ne vaut un bon NTP qui mettra votre horloge à jour tout seul !

```
apt-get install ntp
```

Ensuite, ajoutons les serveurs NTP pour la France :

```
nano /etc/ntp.conf
```

repérez la ligne

```
server ntp.ubuntu.com
```

et rajoutez au dessus la ligne :

```
server fr.pool.ntp.org
```

Enregistrez les modifications, et redémarrez le service NTP :

```
/etc/init.d/ntp restart
```

Vérifiez l'heure et la date :

```
date
```

Autre manipulation intéressante dans certains cas, la désactivation du raccourci clavier Ctrl+Alt+Suppr.

Si vous souhaitez le désactiver :

```
nano /etc/event.d/control-alt-delete
```

Commentez la ligne (avec un # devant)

```
exec /sbin/shutdown -r now "Control-Alt-Delete pressed"
```

Sauvegardez.

### III) Configuration en serveur web Apache2 avec SSL

Dans cette partie nous verrons comment mettre en place un Apache2, avec SSL actif, et comment le forcer sur certaines pages web et pas d'autres.

Tout d'abord, il faut installer apache2 et ce qui va servir plus tard :

```
apt-get install apache2 libapache2-mod-auth-mysql openssl
```

Vous pouvez aussi lancer la commande **tasksel**, et cocher "Serveur LAMP". Ce choix installera Apache2, Mysql et PHP avec pas mal de packages qui pourraient servir.

En ce qui concerne les sites que vous installerez, je vous conseille fortement de les mettre dans un répertoire uniquement prévu à cet effet, et sur une partition différente du système, ça peut aider en cas de besoin de tout réinstaller, ça vous évitera de formater la partition de données.

Ici, on utilisera par exemple le répertoire **/sites** comme répertoire de base pour les sites.

Je ne détaillerai pas le fonctionnement des virtualHosts sur Apache, ni les alias, mais si vous suivez bien ce tuto, vous comprendrez comment ça marche, c'est assez simple ;)

Tout ce passe dans le fichier **default**, que vous trouverez dans le répertoire de votre apache2, soit **/etc/apache2/site-available**. Ce fichier contient les déclarations des sites, les chemins d'accès, et les liens accessibles par le web.

Le but est de déclarer deux « VirtualHost », un pour le http sur port 80, un pour le https sur port 443.

A la base un Virtualhost permet de contrôler un site pour un domaine, mais on peut aussi en créer un

qui les contrôle tous. Chaque virtualhost donne le site par défaut du serveur, les alias, et les autorisations de chacun. Attention, si vous avez plusieurs domaines à gérer sur le même serveur, la configuration ne sera pas la même pour vous.

Au passage, avant d'oublier, ouvrez les ports 80 et 443 sur votre machine :

```
ufw allow 80
ufw allow 443
```

Pour mieux comprendre, voici des exemples de sites qui sont installés sur le serveur et leur chemin d'accès :

- Dotclear en défaut, accessible sur <http://monserveur>
- L'administration de dotclear sur <https://monserveur/admin>
- Un blog phpwebgallery sur <http://monserveur/blog>
- Et un forum phpbb2 sur <http://monserveur/forum>

Le but est de faire ça pour chacun :

- Dotclear accessible en http et https (pas obligatoire pour ce dernier, mais ça coûte rien donc je le laisse)
- Administration de dotclear autorisée en https, refusée en http (pour ne pas diffuser le login/mdp en clair sur le net).
- Le blog en http et https (l'administration du blog se fait par la même page que l'accueil, je passerai en https manuellement quand j'aurais besoin, sinon l'accès au public se fait par http).
- Le forum accessible en http et https, à configurer dans son administration pour autoriser l'https à la connexion d'un utilisateur.

Sauvegardez le fichier original, et ensuite ouvrez le :

```
cp /etc/apache2/site-available/default /etc/apache2/site-available/default-
original
nano /etc/apache2/site-available/default
```

Voici le contenu du fichier qu'il faudra obtenir :

```
#On declare le virtualhost http, qui concerne l'ensemble du domaine
NameVirtualHost *:80

#de meme pour https
NameVirtualHost *:443

#Début du virtualHost http
<VirtualHost *:80> #

#Déclarez ici l'admin du site avec son mail, qui sera accessible sur les pages
#d'erreurs.
```

```
ServerAdmin admin@votredomaine.com

#Ci après le chemin de base vers le site web par défaut - /sites/www où est
#installé mon dotclear, qui doit être le site par défaut.
    DocumentRoot /sites/www

#La declaration d'après était sur apache par défaut, j'ai pas touché :D
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>

#Déclarez les droits et options de la racine du site:
    <Directory /sites/www>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all #on autorise tout le monde
    </Directory>

#De meme que tout à l'heure, la suite j'ai pas touché xD
    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
    <Directory "/usr/lib/cgi-bin">
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>

    ErrorLog /var/log/apache2/error.log

    LogLevel warn

    CustomLog /var/log/apache2/access.log combined
    ServerSignature On

Alias /doc/ "/usr/share/doc/"
<Directory "/usr/share/doc/">
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Allow from 127.0.0.0/255.0.0.0 ::1/128
</Directory>

#Passons maintenant aux déclarations des alias
#Dans cet exemple, le dotclear installé dans /sites/www, a un panneau
#d'administration accessible dans /sites/www/admin, à la base on n'a pas
#besoin de le déclarer, mais pour bloquer son accès en http il faut le faire.
#On l'autorisera uniquement en https pour ne pas transférer les mots de passe
#en clair. Vous pouvez utiliser cette même méthode pour un autre site qui
#possède un panneau de configuration via le web, adaptez juste le chemin ;)

Alias /admin "/sites/www/admin/"
<Directory "/sites/www/admin">
    deny from all
```

```
</Directory>
#Voilà, votre http://ip-serveur/admin est bloqué ;)
#Je possède également un blog, je veux autoriser son accès en http.
#Déclarons-le comme tel. Il est installé dans le répertoire /sites/blog.
    Alias /blog "/sites/blog/"
    <Directory "/sites/blog">
        allow from all
    </Directory>
#J'ai également un forum dans /sites/forum, je veux qu'il soit autorisé en
#http.
    Alias /forum "/sites/forum/"
    <Directory "/sites/forum">
        allow from all
    </Directory>
#J'aurais pu continuer avec les autres sites que je possède, mais c'est la
#même chose que précédemment. Terminons donc la configuration de ce
#virtualhost, en fermant les balises.
</VirtualHost>
#Créons le deuxième virtualhost pour https :
<VirtualHost *:443>
#Activons pour celui-ci le SSL, on configurera le module apache et les
#certificats après.
SSLEngine On
SSLOptions +StrictRequire
#vous pourrez changer les chemins si nécessaire – voir fin tuto
SSLCertificateFile /etc/ssl/certs/server.crt
SSLCertificateKeyFile /etc/ssl/private/server.key
#Comme pour tout à l'heure, on configure le virtualhost :
    ServerAdmin admin@domaine.com
    DocumentRoot /sites/www/
#On donne les options à la racine du site
    <Directory /sites/www/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all #On pourrait ne pas l'autoriser, moi je l'autorise.
    </Directory>
#Créons les mêmes alias que tout à l'heure, mais changeons les droits comme
#nécessaire
#On avait bloqué l'admin du dotclear en http, autorisons le en https.
```

```
Alias /admin "/sites/www/admin/"
<Directory "/sites/www/admin">
    allow from all
</Directory>

#J'autorise également le blog en https, celui ci possède un compte admin pour
#le configurer, mais sur la même page que le blog, je passerais donc
#manuellement en https lorsque j'aurais besoin de l'administrer pour crypter
#le mot de passe sur le net.

Alias /blog "/sites/blog/"
<Directory "/sites/blog">
    allow from all
</Directory>

#Pour le forum, activons le https, à vous après de configurer le forum pour
#qu'il puisse passer en https automatiquement à la connexion d'un utilisateur.

Alias /forum "/sites/forum/"
<Directory "/sites/forum">
    allow from all
</Directory>

</VirtualHost>
```

Et voilà, le deuxième virtualhost est terminé. Vous voyez un peu comment ça fonctionne ?

En fonction de vos sites, adaptez les alias, supprimez ceux inutiles, ceci n'est que pour illustrer les possibilités en fonction des sites.

Voici les commandes pour créer les certificats et clés pour SSL.

Tout d'abord, activons le module SSL pour apache2 (en root ou par sudo).

```
a2enmod ssl
```

Voici les étapes :

1. On crée la clé privée et publique pour l'encryptage.
2. On crée le certificat basé sur la clé publique.
3. On auto-signe notre certificat, ce n'est pas recommandé, il faudrait normalement l'envoyer au CA pour validation et signature. Le fait qu'il soit auto-signé nécessitera un ajout d'exception dans firefox ou IE.

Créons les clés :

```
openssl genrsa -des3 -out server.key 1024
```

Donnez une passphrase !

Quand le service apache sera lancé, il aura besoin de la passphrase, puisqu'on ne peut pas se le permettre (car il faudrait la donner à chaque boot) on va faire une petite modification. Lorsque l'on vous le demandera, donnez la passphrase précédemment choisie :

```
mv server.key server-old.key
openssl rsa -in server-old.key -out server.key
```

Voilà. Maintenant on crée le CSR, grâce à la clé :

```
openssl req -new -key server.key -out server.csr
```

On vous demande de compléter certains champs :

```
Country name : FR
State or Province Name (full name) [Some-State]: France
Locality Name (eg, city) []: votre ville
Organization Name (eg, company) [Internet Widgits Pty Ltd]: nom d'organisation
Organizational Unit Name (eg, section) []: facultatif
Common Name (eg, YOUR name) []: votre nom
Email Address []: votre email
```

Le reste est facultatif.

On l'auto-signe :

```
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

On les mets dans les répertoires inscrits dans la configuration SSL d'apache :

```
sudo cp server.crt /etc/ssl/certs
sudo cp server.key /etc/ssl/private
```

Relancez votre apache :

```
/etc/init.d/apache2 reload
```

Tout devrait fonctionner. Vérifiez sur votre navigateur l'accès aux pages :

<http://ip-serveur>

<https://ip-serveur> celui-ci vous demandera d'ajouter une exception, faite le.

#### IV) Configuration en serveur de partage de fichiers SAMBA

Si vous n'aviez pas installé samba à l'installation, nous allons corriger ça. Lancez la commande :

```
tasksel
```

Cochez **Samba-file-server** et validez. Vous auriez également pu l'installer en effectuant la commande :

```
apt-get install samba
```

Une fois installé, passons à sa configuration. Tout ce passe dans le fichier **/etc/samba/smb.conf**. Dans ce fichier, on gère les partages, les droits de chacun, et quelques petits paramètres. Mais tout d'abord, créons un utilisateur qui aura accès aux partages. Pour cela on utilise la commande (le mot de passe vous sera demandé) :

```
smbpasswd -a utilisateur
```

Passons maintenant à la configuration, ouvrez le fichier `smb.conf`.

```
nano /etc/samba/smb.conf
```

Repérez les options suivantes, et complétez les :

```
workgroup =           //ici vous mettez votre groupe de travail
server string =       //un champ de description du serveur
netbios name =        //le nom netbios de votre machine, son nom d'hôte
security = user       //le mode de sécurité doit être user.
encrypt passwords = true //pour forcer l'encryptage md5 des mots de passe samba
invalid users = root  //ici mettez les users non autorisés à accéder aux partages.
```

Enfin, allez tout en bas du fichier. Ici vous déclarerez les partages. Voici un exemple de partage. Entre crochets, c'est le nom du partage. Le **comment** est la description que vous verrez lorsque votre souris est sur le partage. Le **path** est le chemin vers le répertoire. L'option **browseable** permet de voir ou non le partage dans le serveur, c'est-à-dire, si lors de l'accès au serveur le partage sera caché, il sera tout de même accessible par celui connaissant son nom. **Write list** est la liste des utilisateurs ayant accès en écriture sur le partage. Ici inscrivez l'utilisateur créé précédemment. Enfin, **create mask** et **directory mask** sont les droits qui seront données aux fichiers et répertoires qui seront créés par l'utilisateur.

```
[documents]
comment = documents partagés
path = /documents
browseable = yes
write list = utilisateur
create mask = 0755
directory mask = 0755
```

Si vous souhaitez que le partage soit accessible par tout le monde, il vous faut rajouter la ligne « **public = yes** ».

```
[documents]
comment = documents partagés
path = /documents
browseable = yes
public = yes
create mask = 0755
directory mask = 0755
```

Si vous voulez que le partage soit publique, mais uniquement en lecture seule, il vous faut par exemple :

```
[documents]
comment = documents partagés
path = /documents
browseable = yes
public = yes
read only = yes
create mask = 0755
directory mask = 0755
```

Si par contre, vous voulez qu'à ce partage, votre utilisateur ait le droit d'écriture, contrairement à tout le monde, rajoutez la ligne **write list**, ce qui donne :

```
[documents]
comment = documents partagés
path = /documents
browseable = yes
public = yes
read only = yes
create mask = 0755
directory mask = 0755
write list = utilisateur
```

A l'inverse, pour un partage accessible à tout le monde, et avec droit d'écriture, il faut :

```
[documents]
comment = documents partagés
path = /documents
browseable = yes
public = yes
create mask = 0755
directory mask = 0755
writable = yes
```

**Attention : N'oubliez pas de changer les droits des répertoires partagés en conséquence !**

Une fois tous vos partages configurés, sauvegardez. Rechargeons la configuration de samba pour tester :

```
/etc/init.d/samba restart
```

Maintenant, passons à l'ouverture des ports. Avec ufw, ouvrez les ports 445, 137, 138 et 139.

```
ufw allow 137
ufw allow 138
ufw allow 139
ufw allow 445
```

Maintenant, tentez un accès. Sous windows il faut utiliser le chemin `\\ip-serveur`. Sous linux, `smb://ip-serveur`. Vous devriez voir apparaître, si browseable est à yes, le partage. A l'ouverture, le login et mot de passe de l'utilisateur créé vous sera demandé si vous aviez configuré le partage pour.

Il y a des commandes très utiles pour le dépannage de Samba. Il y a par exemple les commandes :

**smbclient -d3 -L ip-serveur** qui vous permet de voir la configuration de samba d'un serveur depuis un pc linux client. (nécessite l'installation de smbclient)

**testparm** vous permettra de voir si la configuration est correcte.

**nmblookup** nom-serveur pour voir si le serveur est accessible.

**smbstatus** vous donnera les partages actuellement ouverts, avec les logins et IP en cours.

Voilà ! Le principal est dit.

## V) Configuration en serveur FTP

Installons un serveur FTP, ici proftpd. C'est un serveur FTP très simple à paramétrer.

```
apt-get install proftpd
```

Pendant l'installation, il vous sera demandé la manière de lancer proftpd, choisissez « **Indépendamment** ».

Une fois installé, passons à sa configuration, qui se déroule dans le fichier **/etc/proftpd/proftpd.conf**.

Je vous conseille de sauvegarder le fichier original pour pouvoir faire marche arrière :

```
cp /etc/proftpd/proftpd.conf /etc/proftpd/proftpd.conf-ori
nano /etc/proftpd/proftpd.conf
```

Il y a plusieurs options qu'il faut bien paramétrer, voici les principales :

```
UseFtpUsers on //permet de bannir des utilisateurs, inscrits dans le fichier ftpusers - voir
```

```

//après
ServerName nomdevotreserveur
ServerType standalone //laissez ceci par défaut
UseIPv6 Off //utilisation de l'ipv6 c'est à vous de voir - si votre FAI le supporte,
//activez-le.
AllowStoreRestart on //autoriser la reprise des téléchargements
Port 21 //port du FTP
MaxInstances 5 //nombre d'instances du FTP (30 par défaut). Changez en fonction
//de vos prévisions et de la rapidité de votre upload
User nobody //utilisateur qui lance le processus proftpd - mettez ceci pour la sécurité
Groupnogroup //groupe qui lance le processus proftpd - mettez ceci pour la sécurité
Umask 022 022 //masque de création des fichiers - ceci pour la sécurité
ServerAdmin adressemail //adresse mail de l'administrateur, utilisé sur les pages d'erreurs
RootLogin off //ne pas autoriser root à se connecter au ftp
AccessGrantMsg " -- Acces autorise pour %u --" //message de connexion réussie
AccessDenyMsg " -- Acces refuse --" //message de refus de connexion
DefaultRoot chemin //chemin vers la racine du FTP

```

Créons les utilisateurs du FTP. Il est fortement conseillé de se connecter avec un utilisateur spécialement prévu pour le FTP. Ici nous allons donc créer un utilisateur pour le FTP, qui ne possèdera pas de bash. Il faut d'abord rajouter dans la liste des shells valides, le shell /bin/false.

```
nano /etc/shells
```

Rajoutez /bin/false à la fin. Sauvegardez.

Créons l'utilisateur :

```
useradd nomdevotreutilisateur -s /bin/false
passwd nomdevotreutilisateur
```

En ce qui concerne les utilisateurs, en mettant l'option **UseFtpUsers** à « **on** », on peut bannir tous les utilisateurs présents dans le fichier **/etc/ftpusers**. Je vous conseille de l'activer, au passage cela permettra de bannir tous vos utilisateurs de votre serveur, et de n'autoriser que celui précédemment créé.

Créez le fichier et insérez-y les utilisateurs bannis :

```
touch /etc/ftpusers
nano /etc/ftpusers
```

Voici quelques exemples d'utilisateurs à bannir. Rajoutez tous les utilisateurs de votre système qui ont accès à un bash.

root	administrateur
daemon	hplip
bin	mysql
sys	messagebus
sync	postgres
games	ebox
man	bind

lp mail news uucp proxy www-data backup list irc gnats libuuid dhcp syslog	nagios vmail Debian-exim dansguardian sshd postfix dovecot dct amavis ftp clamav nobody klog
--	--

Ouvrez le port 21 pour le FTP (ou le port que vous avez choisi si vous l'avez changé, modifiez la commande en conséquence.) :

```
ufw allow 21
```

Redémarrez votre serveur FTP :

```
/etc/init.d/proftpd restart
```

## VI) Configuration en serveur Proxy Squid

En cours...

## VII) Configuration en serveur de mails

En cours...

## VIII) Configuration de Fail2ban

En cours...