Serveur d'images ISO en PXE

Table des matières

Avertissement....1
Introduction....1
Matériel et stratégie....2
Architecture du PXE dans /srv/tftp....2
Configuration du réseau....3
Configuration de GRUB....4
Autorisation du forwarding....4
Serveur DHCP....4
Serveur DNS....6
Pare-feu....7
Serveur TFTP + XINETD :....8
Utilitaires PXE....8
Serveur NFS + copie des distributions....9
Serveur HTTP....10
Menu de démarrage....11
Tests....14
Conclusion....15
Discussion sur le PXE....15

Avertissement

Ne commencez pas à « bricoler » un serveur **PXE** dans un environnement de production réel qui contiendrait déjà un serveur **DHCP**, surtout si vous ne maîtrisez pas les aspects réseaux.

Et si vous installez un PXE en milieu pro, assurez-vous des réglages BIOS et d'un mot de passe à l'installation qui vous évitera de voir un jour un utilisateur foutre en l'air vos machines !

Introduction

Fondamentalement, le PXE permet deux choses :

- Primo, déployer et réinstaller rapidement des images disques sur des parcs de machines fixes et généralement homogènes. C'est le but du projet FOG (https://fogproject.org/) en mode « graphique », largement usité en milieu professionnel.

  Comme tout outil de déploiement en masse, FOG s'appuie sur les adresses MAC statiques des PC – **comprenez qu'il est fait pour gérer des parcs dans de grosses structures, et demande une administration rigoureuse**.

- Secundo, installer rapidement des images disques dans des environnements hétérogènes. Par exemple dans des réunions informatisées (install party) gérées par des associatifs, où les particuliers viennent avec leur machine personnelle, se faire gratuitement installer une Debian, Ubuntu ou Mint.

  Dans ce cadre, gérer l'adresse MAC n'a évidemment aucun sens et est même une perte de temps. En outre, avec les machines des particuliers, on peut avoir de mauvaises surprises dans la prise en charge de PXE, et il faut parfois « jongler » pour trouver la bonne combinaison technique...

Matériel et stratégie

Ce tutoriel a été validé sur un PC Dell de récup à 150€, avec 16Go de RAM, un SSD de 500Go et un processeur i5-3570.

L'OS retenu est une **Debian** stable en installation réseau minimale, munie de deux cartes réseaux Gigabit (très important) nommées respectivement **eno1** et **enp2s0**. À vous d'adapter les scripts en conséquence !

La configuration ici proposée fonctionne en BIOS normal & UEFI.

N.B. : pour une obscure raison, le BIOS UEFI de KVM ne supporte pas le chargement NFS du noyau, ce qui explique la double configuration NFS/HTTP ici présentée.

Architecture du PXE dans /srv/tftp

Ce tutoriel s'attachera à recréer l'arborescence suivante.

```
.
├── bios
│   ├── ldlinux.c32
│   ├── libcom32.c32
│   ├── libutil.c32
│   ├── lpxelinux.0
│   ├── menu.c32
│   ├── os -> ../os
│   ├── pxelinux.0
│   └── pxelinux.cfg -> ../pxelinux.cfg/
├── efi32
│   ├── ldlinux.e32
│   ├── libcom32.c32
│   ├── libutil.c32
│   ├── menu.c32
│   ├── os -> ../os
│   ├── pxelinux.cfg -> ../pxelinux.cfg/
│   └── syslinux.efi
├── efi64
│   ├── ldlinux.e64
│   ├── libcom32.c32
│   ├── libutil.c32
│   ├── menu.c32
│   ├── os -> ../os
│   ├── pxelinux.cfg -> ../pxelinux.cfg/
│   └── syslinux.efi
├── os
│   ├── linuxmint-21.1-mate-64bit.iso
│   ├── mint
│   ├── ubuntu
│   ├── ubuntu-mate-18.04.4-desktop-i386.iso
│   └── ubuntu-mate-22.04.2-desktop-amd64.iso
└── pxelinux.cfg
    ├── default
    ├── http
    └── nfs
```

À noter le dossier ***pxelinux.cfg/*** et son fichier ***default*** qui fera ici office de menu graphique pour toutes les architectures.

Configuration du réseau

L'interface WAN sera en DHCP, l'interface LAN en 192.168.150.0/24 avec un bridge pour de futures inclusions (ajouts de machines virtuelles).

- Contenu de */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
iface lo inet loopback

# WAN
auto eno1
allow-hotplug eno1
iface eno1 inet dhcp

# LAN
auto enp2s0
allow-hotplug enp2s0
iface enp2s0 inet manual

auto tap1
allow-hotplug tap1
iface tap1 inet manual
    pre-up ip tuntap add tap1 mode tap user root

auto tap2
allow-hotplug tap2
iface tap2 inet manual
    pre-up ip tuntap add tap2 mode tap user root

auto br1
allow-hotplug br1
iface br1 inet static
    pre-up ip l add name br1 type bridge
    pre-up ip l set tap1 master br1
    pre-up ip l set tap2 master br1
    address 192.168.150.1/24
    bridge_ports enp2s0 regex tap*
    bridge_stp off       # désactiver le
protocole Spanning Tree
    bridge_waitport 0    # délai avant qu'un
port soit disponible
    bridge_fd 0          # temps en secondes
entre learning state et forwarding state
    #bridge_maxwait 3     # temps d'attente max
du démarrage
```

- Redémarrage du réseau :

```
systemctl restart networking
```

Configuration de GRUB

- Dans ***/etc/default/grub***, on vire l'option **quiet** de la ligne GRUB_CMDLINE_LINUX_DEFAULT, et on désactive l'IPv6 à la place :

```
GRUB_CMDLINE_LINUX_DEFAULT="ipv6.disable=1"
```

- Une fois le fichier sauvegardé, on remet GRUB à jour :

```
update-grub
```

Autorisation du forwarding

- Dans ***/etc/sysctl.conf***, on active le passage des trames IPv4 en décommentant la ligne :

```
net.ipv4.ip_forward=1
```

- Une fois le fichier sauvegardé, on vérifie :

```
sysctl -p
```

Serveur DHCP

- Installez le serveur :

```
apt install isc-dhcp-server
```

- Modifiez ***/etc/default/isc-dhcp-server*** :

```
...
DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
...
INTERFACESv4="br1"
```

Remplacez **/etc/dhcp/dhcpd.conf** par :

```
ddns-update-style none;
authoritative;
option magic code 208 = string;
option configfile code 209 = text;
option pathprefix code 210 = text;
option reboottime code 211 = unsigned integer
32;
# Active la gestion des architectures de
micrologiciels
option arch code 93 = unsigned integer 16;
subnet 192.168.150.0 netmask 255.255.255.0 {
    range 192.168.150.30 192.168.150.50;
    option domain-name-servers 192.168.150.1;
    option domain-name "matrix.lan";
    option routers 192.168.150.1;
    option broadcast-address 192.168.150.255;
    default-lease-time 600;
    max-lease-time 7200;
    allow bootp;
    allow booting;
    # Fichier d'amorçage à fournir en fonction
de l'architecture demandeuse
    if option arch = 00:06 {
        filename "efi32/syslinux.efi";
    } else if option arch = 00:07 {
```

```
        filename "efi64/syslinux.efi";
        #filename "efi64/grubx64.efi";
    } else if option arch = 00:09 {
        filename "efi64/syslinux.efi";
        #filename "efi64/grubx64.efi";
    } else {
        filename "bios/lpxelinux.0";
    }
    next-server 192.168.150.1;
# option 208 = d0 - non utilisée ici
# option pxelinux.magic f1:00:74:7e;
# option 209 = d1 - pour gagner du temps, on
écrase le réglage par défaut qui consiste à
rechercher un fichier de configuration nommé
avec l'adresse MAC du client, puis plusieurs
combinaisons hexa utilisant l'adresse IP du
client. On garde la dernière option : celle du
fichier default.
# option 210 = d2 - non utilisée ici
# option 211 = d3 - non utilisée ici
    option configfile "pxelinux.cfg/default";
    #option pathprefix "/srv/tftp/";
    if exists dhcp-parameter-request-list {
# exemple si on voulait envoyé toutes les
nouvelles variables
# option dhcp-parameter-request-list =
concat(option dhcp-parameter-request-
list,d0,d1,d2,d3);
# mais nous, nous voulons ici uniquement écraser
la variable 209 = d1
        option dhcp-parameter-request-list =
concat(option dhcp-parameter-request-list,d1);
    }
    # évalue si l'adresse est déjà utilisée
    ping-check = 1;
}
```

- Redémarrage du serveur DHCP :

```
systemctl restart isc-dhcp-server
```

Quelques explications…

Le paquet **isc-dhcp-server** permet de travailler avec des classes (non utilisées ici) ou des espaces de configuration (un peu l'équivalent des espaces de noms en programmation objet...).

Il permet également d'envoyer des options PXE, en même temps qu'il fournit l'adresse IP au client, à commencer par les indispensables **filename** et **next-server**.

Ici on court-circuite le fonctionnement par défaut du paquet **pxelinux**, qui va rechercher successivement dans le dossier ***pxelinux.cfg/*** un nom de fichier basé sur l'adresse MAC du client. Puis s'il ne trouve pas ce fichier, il construit plusieurs autres noms successifs à partir de l'adresse IP, et en tout dernier lieu, il utilise le fichier ***default***.

Le problème, c'est que chaque tentative dure plusieurs minutes, et que le fichier ***default*** n'est donc atteint qu'au bout d'un temps

considérable (20-30mn), incompatible avec notre but : fournir immédiatement une liste de choix de systèmes à démarrer !

D'où le fait qu'on ait gardé l'option **configfile** dans notre fichier, et qu'on ait rajouté la variable à **dhcp-parameter-request-list**, afin de dire à **pxelinux** d'arrêter sa routine par défaut, et de tout de suite sauter à notre menu ***pxelinux.cfg/default*** que l'on créera plus loin.

Serveur DNS

- Installez le paquet **bind** :

```
apt install bind9
```

- Modifiez le fichier */etc/default/named* en rajoutant l'option **-4** :

```
OPTIONS="-u bind -4"
```

- Ajoutez le fichier de zone direct ***/etc/bind/db.matrix.lan*** :

```
$TTL 3d
@    IN    SOA   pxe.matrix.lan. root.matrix.lan. (
                 20230401       ; Serial
                 604800         ; Refresh
                 86400          ; Retry
                 2419200        ; Expire
                    3600 ) ; Negative Cache TTL
     IN    NS    pxe.matrix.lan.
pxe  IN    A     192.168.150.1
```

- Ajoutez le fichier de zone inverse ***/etc/bind/db.150.168.192*** :

```
$TTL 1d
$ORIGIN  150.168.192.in-addr.arpa.
@    IN    SOA   pxe.matrix.lan. root.matrix.lan. (
                 20230401       ; Serial
                 604800         ; Refresh
                 86400          ; Retry
                 2419200        ; Expire
                 3600 )     ; Negative Cache TTL

     IN    NS    pxe.matrix.lan.
1    IN    PTR   pxe.matrix.lan.
```

- Modifiez ***/etc/bind/named.conf.local*** :

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they
are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

acl goodclients {
    192.168.150.0/24;
    localhost;
};

zone "matrix.lan" {
    type master;
```

```
    file "/etc/bind/db.matrix.lan";
    allow-update { none; };
    allow-transfer { none;};
    allow-query { goodclients ;} ;
};

zone "150.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.150.168.192";
    allow-update { none; };
    allow-transfer { none;};
    allow-query { goodclients ;} ;
};
```

- Modifiez ***/etc/hostname*** :

```
pxe
```

- Modifiez ***/etc/hosts*** :

```
127.0.0.1localhost
192.168.150.1 pxe.matrix.lan pxe

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

- Redémarrage du serveur DNS :

```
systemctl restart bind9
```

Pare-feu

- Installez **shorewall** :

```
apt install shorewall
```

- Créez ***/etc/shorewall/interfaces*** :

```
?FORMAT 2
net     NET_IF
dhcp,tcpflags,nosmurfs,routefilter,logmartians,s
ourceroute=0,physical=eno1
loc     LOC_IF
tcpflags,nosmurfs,routefilter,logmartians,physic
al=br1
```

- Créez ***/etc/shorewall/interfaces*** :

```
fw   firewall
net  ipv4
loc  ipv4
```

- Créez ***/etc/shorewall/snat*** :

```
MASQUERADE    192.168.150.0/24    NET_IF
```

- Créez ***/etc/shorewall/policy*** :

```
loc all       ACCEPT
$FW all       ACCEPT
net all       DROP      $LOG_LEVEL
# THE FOLOWING POLICY MUST BE LAST
all all       REJECT         $LOG_LEVE
```

- Créez ***/etc/shorewall/rules*** :

```
?SECTION ALL
?SECTION ESTABLISHED
?SECTION RELATED
?SECTION INVALID
?SECTION UNTRACKED
?SECTION NEW

# pour SQUID/SQUID-DEB-PROXY...
#REDIRECTloc  3128 tcp  www  -    !192.168.150.1
```

- Lancez le service :

```
systemctl start shorewall
```

Serveur TFTP + XINETD :

- Installez les paquets :

```
apt install xinetd tftp tftpd
```

- Ajoutez le fichier ***/etc/xinetd.d/tftp*** :

```
service tftp
{
    protocol        = udp
    port            = 69
    socket_type     = dgram
    wait            = yes
    user            = nobody
    server          = /usr/sbin/in.tftpd
    server_args     = /srv/tftp
    disable         = no
}
```

- Lancez le service

```
systemctl restart xinetd
```

- Testez l'accès

```
echo "coucou" > /srv/tftp/test
chown -R nobody:nogroup /srv/tftp
cd /tmp
tftp 192.168.150.1
get test
quit
cat test
```

Utilitaires PXE

- Installez les paquets :

```
apt install pxelinux syslinux syslinux-common
syslinux-efi syslinux-utils
```

- Créez le script ***/root/pxe.sh*** :

```
mkdir -P /srv/tftp
# inspiré de cf.
https://doc.ycharbi.fr/index.php/Serveur_PXE_UEF
I
```

```
mkdir -p
/srv/tftp/{bios,efi32,efi64,pxelinux.cfg,os}

# BIOS

cp
/usr/lib/syslinux/modules/bios/{ldlinux.c32,libc
om32.c32,libutil.c32,menu.c32} /srv/tftp/bios/
cp /usr/lib/PXELINUX/{pxelinux.0,lpxelinux.0]
/srv/tftp/bios/
cp /usr/lib/syslinux/memdisk /srv/tftp/bios/

cd  /srv/tftp/bios/
ln -s ../pxelinux.cfg/ .
ln -s ../os/ .

# UEFI 32 bits

cp /usr/lib/SYSLINUX.EFI/efi32/syslinux.efi
/srv/tftp/efi32/
cp
/usr/lib/syslinux/modules/efi32/{libutil.c32,lib
com32.c32,ldlinux.e32,menu.c32} /srv/tftp/efi32/

cd  /srv/tftp/efi32/
ln -s ../pxelinux.cfg/ .
ln -s ../os/ .

# UEFI 64 bits

cp /usr/lib/SYSLINUX.EFI/efi64/syslinux.efi
/srv/tftp/efi64/
cp
/usr/lib/syslinux/modules/efi64/{libutil.c32,lib
com32.c32,ldlinux.e64,menu.c32} /srv/tftp/efi64/

cd  /srv/tftp/efi64/
ln -s ../pxelinux.cfg/ .
ln -s ../os/ .

chown -R nobody:nogroup /srv/tftp
```

P.S. : pas la peine de mettre les fichiers binaires en mode exécutable.

- Exécutez le script :

```
. /root/pxe.sh
```

Serveur NFS + copie des distributions

Le serveur **NFS** va ici jouer le rôle de CD-ROM une fois le **PXE** démarré.

L'idée est de regrouper les images des OS du PXE dans */srv/tftp/os*. Modifiez donc */etc/exports* en rajoutant le partage :

```
/srv/tftp/os
192.168.150.0/24(ro,sync,no_wdelay,insecure_lock
s,insecure,no_root_squash,no_subtree_check)
```

- Téléchargez les images ISO de la **Ubuntu 18.04** et **22.04** depuis le net dans */srv/tftp/os* (à vous de chercher les URLS!).
- Copiez le contenu des ISO dans les dossiers cibles :

```
cd /srv/tftp/os
# pour la 18.04 - 32bits
mount -o loop,ro -t iso9660 ubuntu-mate-18.04.4-
desktop-i386.iso mnt/
mkdir -P /srv/tftp/os/ubuntu/1804
rsync -a mnt/ /srv/tftp/os/ubuntu/1804
umount mnt/
# pour la 22.04 - 64 bits
mount -o loop,ro -t iso9660 ubuntu-mate-22.04.2-
desktop-amd64.iso mnt/
mkdir -P /srv/tftp/os/ubuntu/2204
rsync -a mnt/ /srv/tftp/os/ubuntu/2204
umount mnt/
# on n'oublie pas de changer les droits...
chown -R nobody:nogroup /srv/tftp
```

Attention : un ingénieur fou a décidé que les nouvelles versions de la Debian/Ubuntu/Mint contiennent désormais dans leur ISO un dossier caché ***.disk*** qu'il ne faut pas oublier de copier, sinon erreur *Unable to find a live file system on the network* garantie !

Pour la Mint, le principe est le même et n'est donc pas détaillé dans ce document...

- Relancez le serveur **NFS** :

```
systemctl restart nfs-kernel-server
```

```
exportfs -av
showmount -e
```

- Vérifiez manuellement le montage **NFS** :

```
mount -t nfs
192.168.150.1:/srv/tftp/os/ubuntu/1804 /mnt
ls -l /mnt
# vous devez voir les fichiers
umount /mnt
# répétez l'opération pour la 22.04 !
```

Serveur HTTP

- Installez Apache2 et le module d'authentification tiers :

```
apt install apache2 apache2-mpm-itk
```

- Activez le module :

```
a2enmod mpm-itk
```

Explication : ce module va nous permettre de travailler avec l'utilisateur **nobody** et le groupe **nogroup**, à la place des traditionnels **www-data:www-data**.

- Créez le fichier ***/etc/apache2/sites-available/pxe.conf*** :

```
<Directory /srv/tftp/os>
    Options Indexes FollowSymLinks Multiviews
    AllowOverride none
    Require all granted
</Directory>
```

```
<VirtualHost *:8080>
    ServerAdmin contact@matrix.lan
    ServerName matrix.lan
    ServerAlias www.matrix.lan
    DocumentRoot /srv/tftp/os
    AssignUserID nobody nogroup
</VirtualHost>
```

- Modifiez ***/etc/apache2/ports.conf*** pour autoriser le serveur par défaut et le serveur en 8080 uniquement sur le LAN :

```
# If you just change the port or add more ports
here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 192.168.150.1:80
Listen 192.168.150.1:8080

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

- Activez l'hôte virtuel :

```
a2ensite pxe
systemctl restart apache2
```

Normalement, vous devez pouvoir vous connecter depuis le réseau LAN sur http://192.168.150.1:8080.

Menu de démarrage

L'idée est de créer 3 menus :

- un menu principal d'accueil
- un menu pour le mode HTTP + NFS (+ rapide / BIOS récents)
- un menu pour le mode NFS seul (+ lent / anciens BIOS)

avec possibilité de navigation et de retour au menu principal.

J'ai rajouté ici les entrées pour la Mint 21 et la Debian en netinst.

- Menu principal ***/srv/tftp/pxelinux.cfg/default*** :

```
DEFAULT menu.c32
MENU TITLE PXE DEBIAN
MENU MARGIN 0
MENU ROWS -9
MENU TABMSG
MENU TABMSGROW -3
MENU CMDLINEROW -3
MENU HELPMSGROW -4
MENU HELPMSGENDROW -1
MENU COLOR SCREEN 30;47
MENU COLOR BORDER 30;47
MENU COLOR TITLE 30;47
```

```
MENU COLOR SCROLLBAR 30;47
MENU COLOR SEL 37;40
MENU COLOR UNSEL 30;47
MENU COLOR CMDMARK 30;47
MENU COLOR CMDLINE 30;47
MENU COLOR TABMSG 37;40
MENU COLOR DISABLED 37;40
MENU COLOR HELP 37;40

LABEL 0 Local disk
menu default
localboot 0

LABEL 1 HTTP + NFS MODE (FASTER)
KERNEL menu.c32
APPEND pxelinux.cfg/http

LABEL 2 NFS ONLY MODE (SLOWER)
KERNEL menu.c32
APPEND pxelinux.cfg/nfs
```

- Menu secondaire ***/srv/tftp/pxelinux.cfg/http*** :

```
DEFAULT menu.c32
MENU TITLE PXE HTTP + NFS
MENU MARGIN 0
MENU ROWS -9
MENU TABMSG
MENU TABMSGROW -3
MENU CMDLINEROW -3
MENU HELPMSGROW -4
MENU HELPMSGENDROW -1
MENU COLOR SCREEN 30;47
MENU COLOR BORDER 30;47
MENU COLOR TITLE 30;47
MENU COLOR SCROLLBAR 30;47
MENU COLOR SEL 37;40
MENU COLOR UNSEL 30;47
MENU COLOR CMDMARK 30;47
MENU COLOR CMDLINE 30;47
MENU COLOR TABMSG 37;40
MENU COLOR DISABLED 37;40
MENU COLOR HELP 37;40

LABEL 0 Retour menu principal
KERNEL menu.c32
APPEND pxelinux.cfg/default

LABEL 1 Ubuntu 22.04 LTS
KERNEL
http://192.168.150.1:8080/ubuntu/2204/casper/vml
inuz
APPEND boot=casper ip=dhcp netboot=nfs
nfsroot=192.168.150.1:/srv/tftp/os/ubuntu/2204
debian-installer/language=fr locale=fr_FR.UTF-8
bootkbd=fr console-setup/layoutcode=fr console-
setup/variantcode=oss vt.global_cursor_default=1
--
```

```
INITRD
http://192.168.150.1:8080/ubuntu/2204/casper/ini
trd

LABEL 2 Mint 21.1 Mate
KERNEL
http://192.168.150.1:8080/mint/21/casper/vmlinuz
APPEND boot=casper ip=dhcp netboot=nfs
nfsroot=192.168.150.1:/srv/tftp/os/mint/21
debian-installer/language=fr locale=fr_FR.UTF-8
bootkbd=fr console-setup/layoutcode=fr console-
setup/variantcode=oss --
INITRD
http://192.168.150.1:8080/mint/21/casper/initrd.
lz

LABEL 3 Debian netinst
KERNEL
http://192.168.150.1:8080/debian/netinst/install
.amd/vmlinuz
APPEND boot=live netboot=nfs
nfsroot=tftp://192.168.150.1/os/debian/netinst
ro --
INITRD
http://192.168.150.1:8080/debian/netinst/install
.amd/initrd.gz

LABEL 4 Ubuntu 18.04 LTS
KERNEL
http://192.168.150.1:8080/ubuntu/1804/casper/vml
inuz
APPEND boot=casper netboot=nfs
nfsroot=192.168.150.1:/srv/tftp/os/ubuntu/1804
debian-installer/language=fr
console-setup/layoutcode=fr
console-setup/variantcode=oss --
INITRD
http://192.168.150.1:8080/ubuntu/1804/casper/ini
trd
```

- Menu secondaire ***/srv/tftp/pxelinux.cfg/nfs*** :

```
DEFAULT menu.c32
MENU TITLE PXE NFS ONLY
MENU MARGIN 0
MENU ROWS -9
MENU TABMSG
MENU TABMSGROW -3
MENU CMDLINEROW -3
MENU HELPMSGROW -4
MENU HELPMSGENDROW -1
MENU COLOR SCREEN 30;47
MENU COLOR BORDER 30;47
MENU COLOR TITLE 30;47
MENU COLOR SCROLLBAR 30;47
MENU COLOR SEL 37;40
MENU COLOR UNSEL 30;47
MENU COLOR CMDMARK 30;47
```

```
MENU COLOR CMDLINE 30;47
MENU COLOR TABMSG 37;40
MENU COLOR DISABLED 37;40
MENU COLOR HELP 37;40

LABEL 0 Retour menu principal
KERNEL menu.c32
APPEND pxelinux.cfg/default

LABEL 1 Ubuntu 22.04 LTS
KERNEL os/ubuntu/2204/casper/vmlinuz
APPEND boot=casper ip=dhcp netboot=nfs
nfsroot=192.168.150.1:/srv/tftp/os/ubuntu/2204
debian-installer/language=fr locale=fr_FR.UTF-8
bootkbd=fr console-setup/layoutcode=fr console-
setup/variantcode=oss vt.global_cursor_default=1
--
INITRD os/ubuntu/2204/casper/initrd

LABEL 2 Mint 21.1 Mate
KERNEL os/mint/21/casper/vmlinuz
APPEND boot=casper ip=dhcp netboot=nfs
nfsroot=192.168.150.1:/srv/tftp/os/mint/21
debian-installer/language=fr locale=fr_FR.UTF-8
bootkbd=fr console-setup/layoutcode=fr console-
setup/variantcode=oss --
INITRD os/mint/21/casper/initrd.lz

LABEL 3 Debian netinst
KERNEL
http://192.168.150.1:8080/debian/netinst/install
.amd/vmlinuz
APPEND boot=live netboot=nfs
nfsroot=tftp://192.168.150.1/os/debian/netinst
ro --
INITRD
http://192.168.150.1:8080/debian/netinst/install
.amd/initrd.gz

LABEL 4 Ubuntu 18.04 LTS
KERNEL os/ubuntu/1804/casper/vmlinuz
APPEND boot=casper netboot=nfs
nfsroot=192.168.150.1:/srv/tftp/os/ubuntu/1804
debian-installer/language=fr
console-setup/layoutcode=fr
console-setup/variantcode=oss --
INITRD os/ubuntu/1804/casper/initrd
```

P.S. :

- j'ai essayé d'utiliser **INCLUDE** pour simplifier les fichiers de configuration en suivant la documentation officielle, mais ça ne semble pas fonctionner sous KVM : le fichier externe est ignoré. Si vous avez une explication, merci de me la transmettre !
- on aurait pu chaîner la ligne INITRD dans les lignes APPEND avec **initrd=...** mais cette syntaxe est plus « visuelle ».
- remarquez la nouvelle option **ip=dhcp** depuis la **20.04**, indispensable au démarrage (on peut bien entendu aussi mettre

une adresse statique + les indications passerelle/DNS à la main - non abordé ici).

- si le mode HTTP est plus rapide que NFS dans la pratique, il faut comprendre que le HTTP n'est pas un système de fichier comme l'est NFS : le HTTP fonctionne en mode « fichier à la demande », et permet donc de charger le noyau et le RAMFS plus rapidement, mais pour le reste, NFS reste indispensable.
- l'autre solution est de télécharger toute l'image ISO de la distribution en HTTP côté client, mais c'est beaucoup plus lent dans la pratique et ça sature le réseau ! Il faut comprendre ici que les ISO des dernières distributions GNU/Linux pèsent désormais plusieurs Go, donc même sur un réseau Gbit optimal (soit 100Mo/s réels), c'est des dizaines de secondes d'attente pour rien... Dans la pratique le couple HTTP + NFS reste actuellement le meilleur compromis.

Tests

- Installez le paquet :

```
apt install virt-manager
```

- Les test seront faits avec les commandes de commande suivantes.

  Si PXE en mode BIOS non UEFI (fichiers ****.c32*** issus de ***/usr/lib/syslinux/modules/bios***) :

```
kvm --name vm1 -m 4096 -device AC97 -device
intel-hda -device hda-duplex -smp cpus=2 -net
tap,ifname=tap1,script=no,downscript=no -net nic
```

  Si PXE en mode BIOS UEFI (fichiers ****.c32*** issus de ***/usr/lib/syslinux/modules/efi64***) :

```
kvm --name vm1 -m 4096 -device AC97 -device
intel-hda -device hda-duplex -smp cpus=2 -net
tap,ifname=tap1,script=no,downscript=no -net nic
-bios OVMF.fd
```

P.S. : on remarquera l'option **-net nic** qui force le démarrage réseau, et l'option **-bios OVMF.fd** (qui suppose l'installation préalable du paquet **ovmf** pour émuler un BIOS UEFI sous KVM)

Sur une machine physique :

- N'oubliez pas de désactiver le **Secure Boot** si BIOS UEFI.
- En général, la touche **F12** permet d'appeler le boot réseau si vous testez depuis une machine physique.

Conclusion

Malgré sa complexité de mise en place, la solution ici présentée est assez facile à maintenir sous Debian, et **a surtout le mérite d'être « propre » côté dossiers**.

Elle offre une certaine souplesse à l'utilisation, en pouvant choisir à tout moment de basculer entre le mode UEFI et NON UEFI.

Enfin, elle évite de copier les distributions 2x, une pour NFS et l'autre pour HTTP...

Bref, elle fait son boulot, et c'était bien le but de ce tutoriel.

Discussion sur le PXE

Ma critique principale sur le PXE est que c'est le serveur DHCP qui indique au client PXE quel serveur et quel fichier de démarrage minimal il doit utiliser.

Il n'est donc pas possible d'isoler tout le mécanisme dans une machine virtuelle qu'il suffirait de copier/coller d'un S.I. à un autre.

Plus globalement, on sent bien que le PXE n'est pas encore « uniformisé », certains constructeurs continuant d'exiger des options à part non standardisées.

Et si les anciens BIOS sont condamnés à disparaître dans les 20 ans à venir - ce qui simplifiera les futures configurations - les BIOS UEFI ne sont pas un gage et une promesse de simplicité non plus !

On l'a bien vu avec le Secure Boot, qui permet à certains éditeurs d'utiliser leurs mises à jour pour réécrire l'ordre de démarrage dans le BIOS, à l'insu des usagers, bien évidemment...

On peut ensuite rêver demain de n'avoir plus que des terminaux d'accès au cloud tout puissant, le PXE disparaissant avec les administrateurs.

Ce n'est pas impossible en soi, mais cela signifierait que plus personne n'aurait la main sur ses données, et n'aurait accessoirement plus aucun moyen de les protéger, et donc de se protéger...

La facilité immédiate n'est jamais synonyme de liberté.

Ce sera la conclusion finale de ce tutoriel – merci de l'avoir suivi !